Ike Starter Set

Table of Contents

How to Read This Guide

The Identicon

Every component in the knowledge base has a formal public identity. This guide makes that identity visible as an **identicon**: a small square image computed deterministically from the component's own public identifier — a visual hash of identity, and nothing else. Its purpose is recognition: the same component draws exactly the same identicon everywhere it appears — in this guide's HTML and PDF, in print, and onscreen in applications such as Komet, all of which compute it from the same identity data — so you can recognize a component at a glance, and see that two references name the same thing, before reading a word. Because it is derived from identity alone, an identicon carries no meaning about what the component is; the name, the sigil, and the status marks carry that.

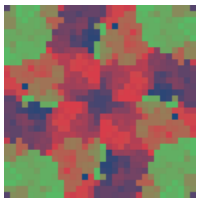

Figure 1. The identicon of English Language, computed from its public identifier — the same image wherever this component appears

Koncept Badges

Every formally identified term in this guide renders as a **Koncept Badge**: one leading mark — a kind sigil for the marked kinds, or a Koncept's logical-status marks, both described below — then the identicon, then the component's name in small caps. A sigil always immediately precedes an identicon — it marks what kind of thing the identicon identifies, and never stands alone in a badge; status marks follow the same rule. Kind sigils and status marks never co-occur, so a badge's lead always says exactly one thing. A badge is not styling; it is a reference to one exact component in the knowledge base, and it links to that koncept's entry in the Koncept Glossary at the end of this book, where its definition, axioms, and identifiers live. The prose is knowledge-base-native too: each chapter's narrative is itself stored in the knowledge base and rendered here, so the same content backs this guide and every other surface that reads the set.

Figure 2. Anatomy of a Koncept Badge — Description Pattern's badge, dissected

A STAMP badge follows the same anatomy with provenance in place of a name — shown here for this set's own inception STAMP, declared in the ledger source as Active · Inception · IKE Community · IkeFoundation module · Primordial path. Inception is the platform's own named instant for time before any release — every version in this pre-release set carries it, and it renders as the word rather than a date on every surface.

Figure 3. Anatomy of a STAMP badge — the set's inception stamp, dissected

The Component Kinds

Everything in the knowledge base is one of a small set of component kinds, and a badge is honest about which kind it names. The scheme is *Koncept bare, everything else marked*: a Koncept badge carries no kind sigil, and every other kind announces itself with a sigil ahead of the identicon. Bare of a kind mark, a Koncept's badge instead leads with its logical status — the marks of the following section — when it has a stated definition.

Table 1. The component kinds, their badge sigils, and a concrete example of each

Kind	Badge	What it is
Koncept	no kind sigil — leads with its status marks (next section) e.g. ⊑ English Language	A **Koncept** — with a K — is the formal construct in the knowledge base, deliberately distinct from a concept in the mind. A concept is the thought a person holds; a Koncept is the identified, versioned construct that carries descriptions (its names) and a logical definition (its meaning) — formally, the referent of the semiotic triangle: the thing the badge's name symbolizes and a reader's concept refers to. Most badges in this guide are Koncepts.

Kind	Badge	What it is
Pattern	P violet P e.g. P DESCRIPTION PATTERN	The schema for a family of semantics: what a semantic of this pattern means as a whole, and the ordered fields it carries — each with its own meaning, purpose, and data type. Every pattern in this set has its own shape table (see below).
Description	D amber D e.g. D UNINITIALIZED COMPONENT (SOLOR)	Human-readable text bound to a concept — itself a semantic, of P DESCRIPTION PATTERN or a dialect pattern. Every name rendered in this guide, including the text inside each badge, is a description in the knowledge base.
Semantic	S green S e.g. S GRETEL	A structured assertion attached to any component, instantiating exactly one pattern: its fields hold the values, in the pattern's declared order. The *e.g.* rows in the shape tables are read from real semantics.
STAMP	grey pentagon e.g. Active · Inception · IKE Community	The provenance every version of every component carries: status, time, author, module, and path. A STAMP badge leads with the pentagon sigil — a point-up five-dimensional radar, one reading per STAMP dimension — then the STAMP's own identicon, which tells one STAMP from another at a glance, then the compact provenance text (status, time, author) in place of a name: a STAMP is identified like every component, but what it says is provenance.

The concept and pattern examples are live badges — click either to visit its glossary entry. The description, semantic, and STAMP examples are *specimen* badges drawn from the set's real content — its earliest-authored description, the author roster's first entry, and the set's inception provenance. A specimen renders in full — sigil, identicon, name — because a sigil never stands alone; it is unlinked only because its referent is not a named term in this guide's glossary. The badge language is the same on every surface that renders individual semantics and versions: Komet itself, chat and mail integrations, and this guide. A red question-mark sigil (**?**) marks a reference whose kind could not be determined.

Koncept Status Marks

Bare of a kind sigil, a Koncept badge has leading marks of its own: the copula of its stated logical definition, straight out of EL++, with a fork appended when its parentage branches. The marks are data, not decoration — each badge shows the connective its concept's own definition uses — and they are always visible, on every surface, never an artifact of hovering. (Komet's navigator carries the same facts today as a hover tooltip and a row-end indicator; the badge marks make them legible at a glance, everywhere a badge renders.)

Table 2. The Koncept status marks and a live example of each

Mark	Reading	Example
⊑ (grey)	*Primitive* — the definition states necessary conditions only: what is true of every instance, but not enough to classify one. The overwhelmingly common case.	e.g. ⊑ ENGLISH LANGUAGE
⊑∨ (fork in blue)	Primitive, with more than one stated parent — the taxonomy branches upward here. The fork appends to whichever copula the concept carries.	e.g. ⊑∨ ARRAY DEFAULT
≡ (green)	*Sufficiently defined* — the stated definition is enough to classify an instance; the concept is equivalent to its definition.	This set authors none yet: sufficient definitions are the exception in real terminologies, not the rule (see ⊑ SUFFICIENT SET).
⊤ (dark amber)	The taxonomy root — a stated definition with nothing above it.	e.g. ⊤ INTEGRATED KNOWLEDGE MANAGEMENT

A Koncept with no stated logical definition carries no status mark at all — a truly bare badge is itself informative.

Pattern-Shape Tables

Each pattern's chapter renders its shape as a numbered table — for example Description Pattern shape — organized as one three-row group per **Model Feature**: first the pattern's referenced component, then each declared field, 1-indexed, in declared order.

- An italic label row names the feature: *Referenced component* (the component — concept, semantic, pattern, or STAMP — this pattern's semantics attach to) or *Field N*. The referenced component is never numbered "Field 0" because it is not part of the fields array: it is a chronicle-level property of the semantic, structurally separate from the field values.
- A content row names the feature's **Meaning** — the concept saying what a value of this feature is — its **Purpose** — the concept saying why the pattern carries it — and its **Data type**, drawn from the closed set of data-type concepts the platform actually resolves. The referenced component has no data type, so that cell is an em dash.
- An indented *e.g.* row shows the feature's value on one real semantic already in the knowledge base — deterministically the earliest-authored one, so the example is stable across regenerations — rendered as a badge when the value is itself an identified component, and as plain text otherwise. A pattern with no semantics yet simply has no e.g. rows.
- An indented *default* row shows the field's value from the pattern's default value semantic (see the Default Values chapter), rendered the same way. Only a pattern with an authored default carries these rows.

The Glossary and the List of Patterns

Every koncept referenced anywhere in this guide has an entry in the comprehensive, alphabetical Koncept Glossary that closes the book — definition, axioms, identifiers, and taxonomy links to parents and children. The List of Patterns closing the first chapter indexes every pattern's glossary entry, the same book convention as a List of Tables or a List of Figures. Occasional indented concept trees show a live taxonomy subtree of the knowledge base, each node a badge.

1. What This Set Provides

The IkeFoundation knowledge set declares its content as an append-only ledger, rooted at ⊑ IkeFoundation root; this guide and its comprehensive glossary are generated from the same materialized store as the released change set.

The chapters that follow describe how this set's own concepts support the platform's core mechanisms — language and dialect, STAMP versioning, logical and navigational coordinates, EL++, the field-level meta-schema, and the base component/chronicle model itself — each naming the actual koncepts involved. The Koncept Glossary at the end lists every koncept in the set, alphabetically, for lookup.

Every pattern in the set is defined by a referenced-component meaning and purpose, plus a declared, ordered list of fields, each with its own meaning, purpose, and data type — a pattern's own semantics are exactly the set of components carrying that pattern's public id. The list below indexes all of them — a List of Patterns, the same book convention as a List of Tables or a List of Figures, styled like the table of contents above — each entry linking to that pattern's own full shape, rendered as a table in the Koncept Glossary.

List of Patterns

- 35. Version control path pattern

2. Language Concepts

No curated narrative for `DescriptionPattern`.

Description Pattern shape shows the pattern's full shape: each Model Feature — the referenced component, then each field in declared order — with its meaning, purpose, and data type, and a live example value drawn from a real semantic already in the set.

Table 3. Description Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Description semantic	⊑∨ Description Attachment	—
e.g. ⊑ *Uninitialized Component*		
Field 1		
⊑∨ Language for description	⊑∨ Language	⊑ Component data type
e.g. ⊑ *English Language*		
Field 2		
⊑∨ Text for description	⊑∨ Description	⊑ String data type
e.g. Uninitialized Component (SOLOR)		
Field 3		
⊑∨ Description case significance	⊑∨ Case Sensitivity Rule	⊑ Component data type
e.g. ⊑ *Description not case sensitive*		
Field 4		
⊑∨ Description type	⊑∨ Description Role	⊑ Component data type
e.g. ⊑ *Fully qualified name description type*		

The dialect patterns that layer acceptability on top of a description are shown in US Dialect Pattern shape and GB Dialect Pattern shape.

Table 4. US Dialect Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Description acceptability	⊑∨ Description semantic	—
e.g. Uninitialized Component (SOLOR)		
Field 1		
⊑∨ United States of America English dialect	⊑∨ Description acceptability	⊑ Component data typ
e.g. ⊑ *Preferred*		

Table 5. GB Dialect Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Description acceptability	⊑∨ Description semantic	—
Field 1		
⊑∨ Great Britain English dialect	⊑∨ Description acceptability	⊑ Component data type

3. STAMP Concepts

Every version of every component carries a STAMP: a (status, author, module, path, time) tuple recording who committed it, in what state, in which module, on which path, and when. Four of those five dimensions are themselves concepts in this set.

3.1. Status

No curated narrative for `StatusValue`.

3.2. Author

No curated narrative for `Author`.

3.3. Module

No curated narrative for `Module`.

Module origins pattern shape shows the module-origins pattern's shape.

Table 6. Module origins pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Originated Module	⊑∨ Module Lineage	—
Field 1		
⊑∨ Module origins	⊑∨ Origin Module Set	⊑ Component Id set data type

3.4. Path

No curated narrative for `Path`.

Path origins pattern shape and Version control path pattern shape show the two path patterns' shapes.

Table 7. Path origins pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Originated Path	⊑∨ Path Lineage	—
e.g. ⊑ *Master path*		
Field 1		
⊑∨ Path concept	⊑∨ Branch Source	⊑ Component data type
e.g. ⊑ *Development path*		
Field 2		
⊑∨ Path origins	⊑∨ Branch Point	⊑ Instant literal

Meaning	Purpose	Data type
e.g. Latest		

Table 8. Version control path pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ PATH	⊑∨ SET MEMBERSHIP	—
e.g. ⊑ *MASTER PATH*		

3.5. Edit Coordinate

No curated narrative for `AuthorForEditCoordinate`.

Put together, a `StampCoordinate` is exactly these four dimensions plus a time: the allowed states, the module preference and exclusion, and a position (path plus time). `StampCalculator` walks a component's version history and returns whichever version is both on an allowed path (at or before the coordinate's time) and in an allowed state — "latest state," computed the same way every time the same coordinate is used.

As a pattern, P STAMP PATTERN itself carries a referenced-component meaning of ⊑∨ VERSION PROPERTIES and a purpose of ⊑∨ COMMIT PROVENANCE — why the pattern exists: to record who committed a version, in what state, module, path, and when. Five fields each carry their own meaning and purpose concept: ⊑∨ STATUS VALUE (meaning) for ⊑∨ STATUS FOR VERSION (purpose), ⊑∨ TIME (meaning) for ⊑∨ TIME FOR VERSION (purpose), and ⊑∨ AUTHOR, ⊑∨ MODULE, and ⊑∨ PATH (meaning) for ⊑∨ AUTHOR FOR VERSION, ⊑∨ MODULE FOR VERSION, and ⊑∨ PATH FOR VERSION (purpose) respectively — the same four koncepts named above, plus time, now restated here by their own exact field-meaning identity. That shape, feature by feature, is STAMP pattern shape.

Table 9. STAMP pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ VERSION PROPERTIES	⊑∨ COMMIT PROVENANCE	—

Meaning	Purpose	Data type
Field 1		
⊑∨ STATUS VALUE	⊑∨ STATUS FOR VERSION	⊑ COMPONENT DATA TYPE
Field 2		
⊑∨ TIME	⊑∨ TIME FOR VERSION	⊑ LONG
Field 3		
⊑∨ AUTHOR	⊑∨ AUTHOR FOR VERSION	⊑ COMPONENT DATA TYPE
Field 4		
⊑∨ MODULE	⊑∨ MODULE FOR VERSION	⊑ COMPONENT DATA TYPE
Field 5		
⊑∨ PATH	⊑∨ PATH FOR VERSION	⊑ COMPONENT DATA TYPE

4. Logic Coordinates

No curated narrative for `LogicCoordinateProperties`.

Solor Concepts Pattern shape and SOLOR concept assemblage shape show the membership patterns named here — the fresh Solor Concepts Pattern and the dormant legacy SOLOR concept assemblage it supersedes.

Table 10. Solor Concepts Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONCEPT PATTERN FOR LOGIC COORDINATE	⊑∨ SET MEMBERSHIP	—

Table 11. SOLOR concept assemblage shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		

Meaning	Purpose	Data type
⊑∨ Concept pattern for logic coordinate	⊑∨ Membership semantic	—

5. Navigation Coordinates

No curated narrative for `NavigationVertex`.

Stated Navigation Pattern shape and Inferred Navigation Pattern shape show the two navigation patterns' shapes.

Table 12. Stated Navigation Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Is-a	⊑∨ Taxonomy Navigation Cache	—
Field 1		
⊑∨ Relationship destination	⊑∨ Is-a	⊑ Component Id set data type
Field 2		
⊑∨ Relationship origin	⊑∨ Is-a	⊑ Component Id set data type

Table 13. Inferred Navigation Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Is-a	⊑∨ Taxonomy Navigation Cache	—
Field 1		
⊑∨ Relationship destination	⊑∨ Is-a	⊑ Component Id set data type
Field 2		
⊑∨ Relationship origin	⊑∨ Is-a	⊑ Component Id set data type

6. EL++ Concepts

No curated narrative for `Axioms`.

6.1. Roles and Role Operators

No curated narrative for `Role`.

6.2. Inclusion, Necessary, and Sufficient Sets

No curated narrative for `InclusionSet`.

6.3. Concrete Value Operators

No curated narrative for `ConcreteValueOperator`.

6.4. Logical Definitions and Grouping

No curated narrative for `LogicalDefinition`.

EL++ Stated Axioms Pattern shape, EL++ Inferred Axioms Pattern shape, and OWL Axiom Syntax Pattern shape show the three axiom patterns' shapes.

Table 14. EL++ Stated Axioms Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STATED DEFINITION	⊑∨ LOGICAL DEFINITION	—
e.g. ⊑ *UNINITIALIZED COMPONENT*		
Field 1		
⊑∨ EL++ STATED TERMINOLOGICAL AXIOMS	⊑∨ LOGICAL DEFINITION	⊑ DITREE DATA TYPE
e.g. 4-vertex tree		

Table 15. EL++ Inferred Axioms Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		

Meaning	Purpose	Data type
⊑∨ Inferred Definition	⊑∨ Logical Definition	—
Field 1		
⊑∨ EL++ Inferred terminological axioms	⊑∨ Logical Definition	⊑ DiTree data type

Table 16. OWL Axiom Syntax Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Axiomatized Component	⊑∨ Axiom Expression	—
Field 1		
⊑∨ Axiom Syntax	⊑∨ Express axiom syntax	⊑ String data type

7. Semantic Field Model

This chapter describes the field-level meta-schema: what kind of value a pattern's field holds, how that value renders in Komet's UI, what its value can mean, and how a field's legal values can be constrained.

7.1. Field Categories

No curated narrative for `FieldCategories`.

7.2. Display Fields

No curated narrative for `DisplayFields`.

7.3. Field Value Terminology

No curated narrative for `Meaning`.

7.4. Constraining Fields

No curated narrative for `ConstrainedPattern`.

⊑∨ Author
⊑ SnoRocket classifier
⊑ IKE Community

⊑ TINKAR STARTER DATA AUTHOR
⊑ GRETEL
⊑ KOMET USER

This section is the shape-level introduction. The Field Constraints chapter, next, describes how the starter set supports constraints in practice — authoring and retrieval, conjunctive composition, each shape's legal-set derivation, the member match relations, and coherence with default values — and renders the two constraint patterns' shapes as Taxonomy Field Constraint Pattern shape and Value-set Field Constraint Pattern shape.

8. Field Constraints

Where the Semantic Field Model chapter's Constraining Fields section introduces the constraint apparatus — two patterns, one per parameter shape, both attaching to the pattern whose field they constrain — this chapter describes how the starter set supports constraints in practice: what authoring one looks like, how several constraints on one field compose, how each shape derives its legal set, and how constraints and default values cohere.

A constraint is first-class content of the pattern it constrains. Unlike a default value — which needs the Defaults and templates module as its category boundary (see the Default Values chapter) — a constraint needs no machinery at all: no special module, no exclusion from any read path, no dedicated accessor. Each constraint semantic attaches to the constrained pattern as its referenced component, and reading a pattern's constraints is retrieval by enumeration: walk the semantics attached to the pattern that are of the two constraint patterns. Identity is declared descriptively, so the ledger reads as prose — the STAMP author constraint below is declared as "Taxonomy Field Constraint: STAMP pattern Author field kind-of Author", the rule's own name saying exactly what it asserts.

Constraint semantics compose conjunctively. Each semantic is one conjunct, and a value is legal only when every conjunct on its field holds — no override, no precedence, no disjunction to reason about. A field may carry constraints of both shapes at once, a taxonomy constraint and a value-set constraint intersecting: legal then means inside the taxonomy region **and** matching an enumerated member.

The pattern axis is the parameter shape: one constraint pattern per parameter tuple, never one per datatype, and never a union carried by sentinel values. A kind or relation field earns its place as a genuine parameter only when every field of the tuple stays meaningful for every one of its values — the four taxonomy kinds share one identical (field, kind, anchor) tuple, so one pattern carries all four; value-set membership carries a different tuple entirely, so it is a pattern of its own — a shape, never a fifth kind. Every field of every constraint semantic is meaningful: no ⊑ BLANK CONCEPT sentinel anywhere in this story.

8.1. The Taxonomy Field Constraint

No curated narrative for `TaxonomyFieldConstraintPattern`.

Taxonomy Field Constraint Pattern shape shows the pattern's shape.

Table 17. Taxonomy Field Constraint Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Constrained Pattern	⊑∨ Field Value Restriction	—
e.g. P Description Pattern		
Field 1		
⊑∨ Constrained field	⊑∨ Constraint Scope	⊑ Component data type
e.g. ⊑∨ *Language for description*		
Field 2		
⊑∨ Constraint kind	⊑∨ Constraint Rule	⊑ Component data type
e.g. ⊑ *Kind-of field constraint*		
Field 3		
⊑∨ Constraint anchor concept	⊑∨ Taxonomy Reference Point	⊑ Component data type
e.g. ⊑∨ *Language*		

The starter set carries five taxonomy constraints on real, long-standing patterns — the four STAMP dimensions and the Description Pattern's language field (STAMP pattern shape and Description Pattern shape show the constrained patterns' own shapes):

- P STAMP pattern 's author, module, and path fields are each kind-of constrained to their own dimension's taxonomy — anchors ⊑∨ Author , ⊑∨ Module , and ⊑∨ Path — the anchor itself a legal value, and every new author, module, or path minted beneath it a legal value from the moment it exists.

- P STAMP PATTERN 's status field is immediate-child constrained to ⊑∨ STATUS VALUE : STAMP status is a closed, one-level value list — exactly ⊑ ACTIVE STATE , ⊑ INACTIVE STATE , ⊑ PRIMORDIAL STATE , ⊑ CANCELED STATE , and ⊑ WITHDRAWN STATE — and the constraint says so as data.
- P DESCRIPTION PATTERN 's language field (meaning LANGUAGE CONCEPT NID FOR DESCRIPTION) is kind-of constrained to ⊑∨ LANGUAGE : any language concept, at any depth, is a legal description language.

8.2. The Value-set Field Constraint

No curated narrative for `ValueSetFieldConstraintPattern`.

Value-set Field Constraint Pattern shape shows the pattern's shape.

Table 18. Value-set Field Constraint Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONSTRAINED PATTERN	⊑∨ FIELD VALUE RESTRICTION	—
e.g. P *PREFERRED REVIEWER PATTERN*		
Field 1		
⊑∨ CONSTRAINED FIELD	⊑∨ CONSTRAINT SCOPE	⊑ COMPONENT DATA TYPE
e.g. ⊑∨ *PREFERRED REVIEWER*		
Field 2		
⊑∨ VALUE-SET PATTERN	⊑∨ LEGAL VALUE SOURCE	⊑ COMPONENT DATA TYPE
e.g. P *STARTER SET AUTHOR ROSTER PATTERN*		
Field 3		
⊑∨ VALUE-SET FIELD	⊑∨ VALUE DISAMBIGUATION	⊑ COMPONENT DATA TYPE
e.g. ⊑∨ *ROSTER AUTHOR*		
Field 4		
⊑∨ MEMBER MATCH RELATION	⊑∨ MATCH RULE	⊑ COMPONENT DATA TYPE

Meaning	Purpose	Data type
e.g. ⊑ *Equal match relation*		

The set's worked value-set constraint is the one the Default Values chapter's worked default already satisfies: P Preferred Reviewer Pattern 's single field, ⊑∨ Preferred reviewer , is constrained to membership in the starter set author roster. The constraint's four fields read: constrained field = ⊑∨ Preferred reviewer , value-set pattern = P Starter Set Author Roster Pattern , value-set field = ⊑∨ Roster author , member match relation = ⊑ Equal match relation (Preferred Reviewer Pattern shape and Starter Set Author Roster Pattern shape show both patterns' shapes).

The roster is why the ⊑∨ Value-set field pointer exists. Each roster entry is a semantic with two fields — ⊑∨ Roster author , the member, and ⊑∨ Roster order , a display order — and only the first is what a candidate value is matched against: ⊑∨ Roster order is a characteristic of the entry, not the member. The pointer names the member-bearing Model Feature by its meaning concept — a declared field's meaning, or, for a membership pattern whose members are the referenced components themselves, the source pattern's referenced-component meaning — so any pattern whose active semantics enumerate things can serve as a value set, with no value-set-specific machinery on the source side.

8.3. The Member Match Relation

No curated narrative for `MemberMatchRelation`.

One relation is admitted today:

No curated narrative for `EqualMatchRelation`.

8.4. The Apparatus Constrains Itself

The apparatus's own discriminator fields are exactly the closed one-level value lists the immediate-child kind exists for, and the set constrains them accordingly — as ordinary content, through the same pattern any other constraint uses:

- "Taxonomy Field Constraint: Taxonomy Field Constraint Pattern Constraint kind field immediate-child of Taxonomy field constraint kind" — constrained field = ⊑∨ Constraint kind , kind = ⊑ Immediate child field constraint , anchor = ⊑ Taxonomy field constraint kind : the kind field of every taxonomy constraint holds one of the four kinds, and nothing else.
- "Taxonomy Field Constraint: Value-set Field Constraint Pattern Member match relation field immediate-child of Member match relation" — constrained field = ⊑∨ Member match relation , kind = ⊑ Immediate child field constraint , anchor =

⊑∨ MEMBER MATCH RELATION again: the relation field of every value-set constraint holds an admitted relation, and nothing else — the same concept serving as the field's meaning and the constraint's anchor without conflict, the two roles read from different fields entirely.

Nothing marks these two semantics as special: they are taxonomy constraints like the STAMP pattern's four, retrieved the same way, composed the same way. Growing either closed list is authoring: mint the child concept, and the existing constraint admits it without being touched — for a relation, the bijection gate additionally requires its evaluator to ship in the same change.

8.5. Constraints and Defaults

Constraints and defaults meet at one coherence rule: a default value must satisfy its field's constraints — a fresh semantic pre-populated from defaults should be legal where a curator has finished choosing, and loudly illegal where they have not.

P PREFERRED REVIEWER PATTERN shows the finished side: its authored default, ⊑ GRETEL, is the roster's own first entry, so the default satisfies the value-set constraint on the very field it fills — a real, constraint-satisfying value, visible as the *default* row of Preferred Reviewer Pattern shape (the Default Values chapter walks the full round trip).

The loud defaults show the unfinished side, deliberately. ⊑ UNINITIALIZED COMPONENT belongs to no domain taxonomy region and no curated value set, and `NaN` matches nothing under ⊑ EQUAL MATCH RELATION, itself included. So wherever a constrained field still carries its loud default, constraint checking fails it until someone chooses a real value — the same machinery that validates finished data announces unfinished data, the loud-defaults principle of the Default Values chapter restated from the checking side.

9. Default Values

An editor opening a fresh semantic needs a starting value for every field — the store rejects null field values outright. This chapter describes how the starter set supports default values with no defaults-specific machinery in the model at all: a default is ordinary working-path content — a complete semantic of the target pattern — kept out of domain reads by two structural dimensions the model already has, its referenced component and its module.

9.1. The Default Value Semantic

No curated narrative for `DefaultValueConcept`.

9.2. The Defaults and Templates Module

No curated narrative for `DefaultsAndTemplatesModule`.

9.3. Templates

No curated narrative for `TemplateConcept`.

9.4. Loud Defaults for Every Data Type

What value should a default actually carry? The governing principle is **loud defaults**: a default must work as an initial value while reading as obviously needing revision — never Java's silent zero-values, which disappear into real data instead of announcing themselves. The starter set demonstrates the principle at full breadth with one pattern carrying a loud default for every data type a pattern field can declare.

No curated narrative for `DataTypeDefaultsPattern`.

Data Type Defaults Pattern shape shows the pattern's full shape; the indented *default* row under each field is read live from the pattern's own default value semantic, through the same computed identity any editor would resolve.

Table 19. Data Type Defaults Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Defaulted Data Types	⊑∨ Data Type Default Provision	—
Field 1		
⊑∨ String default	⊑∨ Data type default exemplar	⊑ String data type
default UNINITIALIZED		
Field 2		
⊑∨ Component default	⊑∨ Data type default exemplar	⊑ Component data type
default ⊑ *Uninitialized Component*		
Field 3		
⊑∨ ComponentIdSet default	⊑∨ Data type default exemplar	⊑ Component Id set data type
default ⊑ *Uninitialized Component*		
Field 4		
⊑∨ ComponentIdList default	⊑∨ Data type default exemplar	⊑ Component Id list data type

Meaning	Purpose	Data type
default ⊑ *Uninitialized Component*		
Field 5		
⊑∨ DiTree default	⊑∨ Data type default exemplar	⊑ DiTree data type
default 1-vertex tree		
Field 6		
⊑∨ DiGraph default	⊑∨ Data type default exemplar	⊑ DiGraph data type
default 2-vertex cycle		
Field 7		
⊑∨ Concept default	⊑∨ Data type default exemplar	⊑ Concept data type
default ⊑ *Uninitialized Component*		
Field 8		
⊑∨ Semantic default	⊑∨ Data type default exemplar	⊑ Semantic data type
default Uninitialized Component (SOLOR)		
Field 9		
⊑∨ Integer default	⊑∨ Data type default exemplar	⊑ Integer data type
default 777777777		
Field 10		
⊑∨ Float default	⊑∨ Data type default exemplar	⊑ Float data type
default NaN		
Field 11		
⊑∨ Boolean default	⊑∨ Data type default exemplar	⊑ Boolean data type
default false		
Field 12		

Meaning	Purpose	Data type
⊑∨ ByteArray default	⊑∨ Data type default exemplar	⊑ Byte array data type
default UNINITIALIZED		
Field 13		
⊑∨ Array default	⊑∨ Data type default exemplar	⊑ Array data type
default [UNINITIALIZED]		
Field 14		
⊑∨ Instant default	⊑∨ Data type default exemplar	⊑ Instant literal
default Pre-inception		
Field 15		
⊑∨ Long default	⊑∨ Data type default exemplar	⊑ Long
default 77777777777777777		
Field 16		
⊑∨ Decimal default	⊑∨ Data type default exemplar	⊑ Decimal data type
default 777777777.777		

The sixteen decided defaults group into a small number of strategies:

- **The UNINITIALIZED string forms.** String is the literal `UNINITIALIZED`; ByteArray is the UTF-8 bytes of `UNINITIALIZED` — the loud String default carried in byte form; Array is an array of exactly one element, the String `UNINITIALIZED` — present and non-empty, since an empty array would be a silent zero-value.
- **The loud placeholder component.** Component and Concept default to ⊑ Uninitialized Component, the base set's own loud placeholder for a value still owed — deliberately not ⊑ Blank Concept: Blank means a deliberate not-applicable, a quiet, legitimate state, while Uninitialized means a value still owed. ComponentIdSet and ComponentIdList are the singleton set and list holding ⊑ Uninitialized Component — present and non-empty, their one member itself the loud placeholder.

- **Native non-values.** Types whose value space offers a true non-value use it — a non-value is louder still, because it cannot be mistaken for data at all. Float is `NaN`, which also propagates through arithmetic, so any computation over an unrevised default stays loudly not-a-number; Instant is the pre-inception instant (historically 'premundane'), the time type's own before-all-recorded-time marker, chosen over a sentinel date because a true non-value cannot be mistaken for a real timestamp.
- **Stretched-sevens sentinels.** Numeric types without a native non-value use a planted repdigit: Integer is `777777777` (nine sevens), Long is `77777777777777777` (eighteen sevens), and Decimal is `777777777.777` — nine sevens, a decimal point, three more sevens, the decimal point placed deliberately to demonstrate the type's distinguishing property: a scaled decimal, not an integer. A repdigit signature reads as deliberately planted and cannot plausibly occur as natural data, where `MIN_VALUE` would read as an overflow bug.
- **Structural values that prove their type.** DiGraph is a deliberate simple cycle — two vertices, both ⊑ Uninitialized Component, with edges in both directions: a cycle can never be a tree, so the value proves the field truly holds a graph and not a tree. DiTree is the smallest well-formed tree — a single vertex, the loud placeholder at its root.
- **A real, resolvable semantic.** The Semantic field's default is the fully qualified name description semantic of ⊑ Uninitialized Component — a real, resolvable semantic whose subject is itself the loud placeholder, so the reference is valid yet plainly unrevised.
- **The flagged compromise.** Boolean is `false`: a two-valued type has no loud member, so no Boolean default can announce itself as unrevised; `false` is offered as the initial value with that limitation stated rather than hidden.

9.5. A Worked Default: Preferred Reviewer

P Preferred Reviewer Pattern carries the set's other default value semantic — the minimal end-to-end demonstration the sixteen-type exemplar generalizes. Its single field, ⊑∨ Preferred reviewer, is value-set constrained to the P Starter Set Author Roster Pattern roster (see the Field Constraints chapter), and its authored default is ⊑ Gretel, that roster's first entry — a real, constraint-satisfying default value, not a placeholder. The full round trip is visible in this guide itself: the default is authored as ordinary ledger content in the Defaults and templates module, `StampCalculator.getDefault(pattern)` resolves it by computed identity, and the same value renders as the *default* row of Preferred Reviewer Pattern shape.

10. Tinkar Base Model

This chapter describes the component/chronicle/pattern-of-patterns meta-model every other chapter's own terminology ultimately descends from.

10.1. Components and Chronicles

No curated narrative for `TinkarModelConcept`.

The base-model shape patterns' own tables follow, in the order the narrative names them: the chronicle and version shapes for components, concepts, patterns, semantics, and STAMPs, then the base-model membership patterns.

Table 20. Component Chronology Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Component field	⊑∨ Chronicle Identity and History	—
Field 1		
⊑∨ Public ID field	⊑∨ Uniquely identify knowledge graph components	⊑ Component D
Field 2		
⊑∨ Component versions field	⊑∨ Component versions set	⊑ Component I

Table 21. Component Version Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Component versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type

Table 22. Concept field pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Concept field	⊑∨ Chronicle Identity and History	—
Field 1		

Meaning	Purpose	Data type
⊑∨ PUBLIC ID FIELD	⊑∨ UNIQUELY IDENTIFY KNOWLEDGE GRAPH COMPONENTS	⊑ COMPONENT DATA
Field 2		
⊑∨ CONCEPT VERSIONS FIELD	⊑∨ CONCEPT VERSIONS SET	⊑ COMPONENT ID SE

Table 23. Concept Version Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONCEPT VERSIONS FIELD	⊑∨ VERSION SNAPSHOT	—
Field 1		
⊑∨ STAMP FIELD	⊑∨ VERSION PROVENANCE	⊑ COMPONENT DATA TYPE

Table 24. Pattern Chronology Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ PATTERN FIELD	⊑∨ CHRONICLE IDENTITY AND HISTORY	—
Field 1		
⊑∨ PUBLIC ID FIELD	⊑∨ UNIQUELY IDENTIFY KNOWLEDGE GRAPH COMPONENTS	⊑ COMPONENT DATA
Field 2		
⊑∨ PATTERN VERSIONS FIELD	⊑∨ PATTERN VERSIONS SET	⊑ COMPONENT ID SE

Table 25. Pattern Version Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ PATTERN VERSIONS FIELD	⊑∨ VERSION SNAPSHOT	—

Meaning	Purpose	Data type
Field 1		
⊑∨ STAMP FIELD	⊑∨ VERSION PROVENANCE	⊑ COMPONENT DATA TYPE
Field 2		
⊑∨ PATTERN MEANING FIELD	⊑∨ MEANING DECLARATION	⊑ COMPONENT DATA TYPE
Field 3		
⊑∨ PATTERN PURPOSE FIELD	⊑∨ PURPOSE DECLARATION	⊑ COMPONENT DATA TYPE
Field 4		
⊑∨ FIELD DEFINITION FIELD	⊑∨ FIELD DEFINITIONS SET	⊑ COMPONENT ID SET DATA TYPE

Table 26. Semantic Chronology Pattern shape

Meaning	Purpose	Data
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ SEMANTIC FIELD	⊑∨ CHRONICLE IDENTITY AND HISTORY	—
Field 1		
⊑∨ PUBLIC ID FIELD	⊑∨ UNIQUELY IDENTIFY KNOWLEDGE GRAPH COMPONENTS	⊑ C
Field 2		
⊑∨ SEMANTIC PATTERN FIELD	⊑∨ PATTERN MEMBERSHIP	⊑ C
Field 3		
⊑∨ SEMANTIC REFERENCED COMPONENT FIELD	⊑∨ ATTACHMENT TARGET	⊑ C
Field 4		
⊑∨ SEMANTIC VERSIONS SET	⊑∨ VERSION HISTORY	⊑ C

Table 27. Semantic version field pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ SEMANTIC VERSIONS FIELD	⊑∨ VERSION SNAPSHOT	—
Field 1		
⊑∨ STAMP FIELD	⊑∨ VERSION PROVENANCE	⊑ COMPONENT DATA TYPE
Field 2		
⊑∨ SEMANTIC FIELD FIELD	⊑∨ SEMANTIC FIELD FIELDS SET	⊑ COMPONENT ID SET DATA TYPE

Table 28. STAMP Chronology Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STAMP FIELD	⊑∨ CHRONICLE IDENTITY AND HISTORY	—
Field 1		
⊑∨ PUBLIC ID FIELD	⊑∨ UNIQUELY IDENTIFY KNOWLEDGE GRAPH COMPONENTS	⊑ COMPONENT DATA
Field 2		
⊑∨ STAMP VERSIONS FIELD	⊑∨ STAMP VERSIONS SET	⊑ COMPONENT ID SET

Table 29. STAMP version field pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STAMP VERSIONS FIELD	⊑∨ VERSION SNAPSHOT	—
Field 1		
⊑∨ STAMP FIELD	⊑∨ VERSION PROVENANCE	⊑ COMPONENT DATA TYPE
Field 2		

Meaning	Purpose	Data type
⊑∨ STATUS FIELD	⊑∨ STATUS FOR VERSION	⊑ COMPONENT DATA TYPE
Field 3		
⊑∨ TIME FIELD	⊑∨ TIME FOR VERSION	⊑ STRING DATA TYPE
Field 4		
⊑∨ AUTHOR FIELD	⊑∨ AUTHOR FOR VERSION	⊑ COMPONENT DATA TYPE
Field 5		
⊑∨ MODULE FIELD	⊑∨ MODULE FOR VERSION	⊑ COMPONENT DATA TYPE
Field 6		
⊑∨ PATH FIELD	⊑∨ PATH FOR VERSION	⊑ COMPONENT DATA TYPE

Table 30. Tinkar base model component pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STARTER DATA AUTHORING	⊑∨ SET MEMBERSHIP	—
e.g. ⊑ *UNINITIALIZED COMPONENT*		

Table 31. Komet base model component pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STARTER DATA AUTHORING	⊑∨ SET MEMBERSHIP	—
e.g. P *KOMET BASE MODEL COMPONENT PATTERN*		

10.2. Base Model Primitives

No curated narrative for `Object`.

11. Value Constraints and Reference Ranges

No curated narrative for `ValueConstraint`.

Value Constraint Pattern shape shows the pattern's shape.

Table 32. Value Constraint Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑ ∨ VALUE CONSTRAINT	⊑ ∨ NUMERIC VALUE RESTRICTION	—
Field 1		
⊑ ∨ VALUE CONSTRAINT SOURCE	⊑ ∨ REFERENCE RANGE AUTHORITY	⊑ CONCEPT DATA TYPE
Field 2		
⊑ ∨ MINIMUM VALUE OPERATOR	⊑ ∨ CONCRETE VALUE OPERATOR	⊑ CONCEPT DATA TYPE
Field 3		
⊑ ∨ REFERENCE RANGE MINIMUM	⊑ ∨ REFERENCE RANGE	⊑ FLOAT DATA TYPE
Field 4		
⊑ ∨ MAXIMUM VALUE OPERATOR	⊑ ∨ CONCRETE VALUE OPERATOR	⊑ COMPONENT DATA TYPE
Field 5		
⊑ ∨ REFERENCE RANGE MAXIMUM	⊑ ∨ REFERENCE RANGE	⊑ FLOAT DATA TYPE
Field 6		
⊑ ∨ EXAMPLE UCUM UNITS	⊑ ∨ UNIT EXAMPLE	⊑ STRING DATA TYPE

12. Authoring Actions and Legacy Constraints

No curated narrative for `ActionProperties`.

13. Feature-Based and Intrinsic Roles

No curated narrative for `DescriptionType`.

14. Annotations, Correlations and Editor Focus

No curated narrative for `AnnotationType`.

Comment pattern shape, Starter Set Author Roster Pattern shape, and Preferred Reviewer Pattern shape show these three patterns' shapes.

Table 33. Comment pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Subject of Commentary	⊑∨ Editorial Clarification	—
Field 1		
⊑∨ Comment	⊑∨ Editorial Clarification	⊑ String data type

Table 34. Starter Set Author Roster Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Starter set author roster	⊑∨ Roster Membership	—
e.g. P *Starter Set Author Roster Pattern*		
Field 1		
⊑∨ Roster author	⊑∨ Roster Entry	⊑ Component data type
e.g. ⊑ *Gretel*		
Field 2		
⊑∨ Roster order	⊑∨ Display Sequence	⊑ Long
e.g. 1		

Table 35. Preferred Reviewer Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Preferred reviewer assignment	⊑∨ Review Routing	—
Field 1		
⊑∨ Preferred reviewer	⊑∨ Assigned Reviewer	⊑ Component data type
default ⊑ *Gretel*		

15. Identifiers

No curated narrative for `IdentifierSource`.

Identifier Pattern shape shows the identifier pattern's shape.

Table 36. Identifier Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Identified Component	⊑∨ External Identity Mapping	—
e.g. ⊑ *Uninitialized Component*		
Field 1		
⊑∨ Identifier Source	⊑∨ Identifier Authority	⊑ Component data type
e.g. ⊑ *Universally unique identifier*		
Field 2		
⊑∨ Identifier Value	⊑∨ Identifier Text	⊑ String data type
e.g. 55f74246-0a25-57ac-9473-a788d08fb656		

16. Coordinate Properties: Language and Path

No curated narrative for `LanguageCoordinateProperties`.

17. Tree Amalgams

No curated narrative for `TreeAmalgamProperties`.

18. Clinical Phenomena and Domain Content

No curated narrative for `Phenomenon`.

19. Object Properties and Relationships

No curated narrative for `ObjectProperties`.

20. Provenance and Root

No curated narrative for `IntegratedKnowledgeManagement`.

21. Koncept Glossary

⊑ **Acceptable**

Specifies that a description is acceptable, but not preferred within a language or dialect.

```
⊑ Description acceptability
```

Parents: ⊑ ∨ DESCRIPTION ACCEPTABILITY

Since: Inception

id: `Acceptable` | UUID: `12b9e103-060e-3256-9982-18c1191af60e`

⊑ **Action properties**

The retired ISAAC action-rule vocabulary: the parameters a rule-driven authoring action took. Retained for identity compatibility.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Children: ⊑ CONDITIONAL TRIGGERS ⊑ CONCEPT CONSTRAINTS ⊑ ROLE TYPE TO ADD ⊑ CONCEPT TO FIND

Since: Inception

id: `ActionProperties` | UUID: `80ba281c-7d47-57cf-8100-82b69bce998b`

⊑ Active state

Concept used to represent a status for components that are active.

```
⊑ Status value
```

Parents: ⊑∨ STATUS VALUE

Since: Inception

id: `ActiveState` | UUID: `09f12001-0e4f-51e2-9852-44862a4a0db4`

⊑ Allowed states for stamp coordinate

Predefined list of values for STAMP coordinate

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `AllowedStatesForStampCoordinate` | UUID: `23f69f6f-a502-5876-a835-2b1b4d5ce91e`

⊑ And

An operator that typically is employed to combine two conditions

```
⊑ Connective operator
```

Parents: ⊑ CONNECTIVE OPERATOR

Since: Inception

id: `And` | UUID: `fa113d51-07d2-587c-8930-0bce207d506d`

⊑ Annotation property set

The OWL annotation-property grouping: properties that annotate content without logical consequence, carried for OWL interchange.

```
⊑ Logical expression model
```

Parents: ⊑ LOGICAL EXPRESSION MODEL

Since: Inception

id: `AnnotationPropertySet` | UUID: `cb9e33de-f82c-495d-89fa-69afecbcd47d`

⊑ Annotation type

Metadata about program elements, and annotation types define the structure of these annotations

⊑ `Legacy model concepts`

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `AnnotationType` | UUID: `3fe77951-58c9-51b3-8e7e-65edcf7ace0a`

⊑ Anonymous concept

Concepts or entities that do not have a specific, named identity, (defined on-the-fly without a dedicated name)

⊑ `Concept type`

Parents: ⊑ CONCEPT TYPE

Since: Inception

id: `AnonymousConcept` | UUID: `f8f936d4-3ac7-5629-9f65-9452608056a1`

⊑ Any component

A general-purpose container to represent any component with generic data structure. Modifiable based on the specific requirements and characteristics of the components.

⊑ `Chronicle and version model`

Parents: ⊑ CHRONICLE AND VERSION MODEL

Since: Inception

id: `AnyComponent` | UUID: `927da7ac-3403-5ccc-b07b-88f60cc3a5f8`

⊑ Array

The retired dynamic-column type for ordered element sequences, superseded by Array data type.

```
⊑ Dynamic column data types
```

Parents: ⊑ DYNAMIC COLUMN DATA TYPES

Since: Inception

id: `Array` | UUID: `318622e6-dd7a-5651-851d-2d5c2af85767`

⊑ Array data type

A lexical set of semantically related elements/items

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `ArrayDataType` | UUID: `b168ad04-f814-5036-b886-fd4913de88c8`

⊑∨ Array default

The Array field's loud default: an array of exactly one element, the String "UNINITIALIZED" — present and non-empty (an empty array would be a silent zero-value), its single element the loud String default.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `ArrayDefault` | UUID: `daae3fab-6803-5c61-84ef-9d1d1f17c97b`

⊑∨ Assigned Reviewer

Why a preferred reviewer value is recorded: to name which author is the preferred reviewer.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑ EDITORIAL MODEL ⊑ PURPOSE

Since: Inception

id: `AssignedReviewer` | UUID: `b25df610-7523-5db8-8954-97878fd4e5e1`

⊑ Atom

A logical axiom that participates in a set's expression rather than framing one: connectives (and, or), concept references, disjoint-with assertions, property sequence implications, and the typed atoms (role, interval role, feature). Mirrors LogicalAxiom.Atom in the sealed hierarchy.

```
⊑ Logical axiom
```

Parents: ⊑ LOGICAL AXIOM

Children: ⊑ CONNECTIVE OPERATOR ⊑ CONCEPT REFERENCE ⊑ TYPED ATOM ⊑ PROPERTY SEQUENCE IMPLICATION ⊑ DISJOINT WITH

Since: Inception

id: `Atom` | UUID: `e9d6a6d4-b509-5852-9ef7-8b4d2e637461`

⊑ ∨ Attachment Target

Why a Semantic Chronology Pattern's Semantic referenced component field is recorded: to identify what component this semantic is attached to.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ CHRONICLE AND VERSION MODEL ⊑ PURPOSE

Since: Inception

id: `AttachmentTarget` | UUID: `1faad323-8fdd-51e3-8209-c1f904d478e0`

⊑ ∨ Author

The concept identifying who committed a version — the author dimension of a STAMP, and the root of the value space an author field's concept is drawn from; distinct from Author for version, which names why the field is recorded.

```
⊑ STAMP dimensions ⊓ Meaning
```

Parents: ⊑ STAMP DIMENSIONS ⊑ MEANING

Children: ⊑ SNOROCKET CLASSIFIER ⊑ IKE COMMUNITY ⊑ TINKAR STARTER DATA AUTHOR ⊑ GRETEL ⊑ KOMET USER

Since: Inception

id: `Author` | UUID: `f7495b58-6630-3499-a44e-2052b5fcf06c`

⊑∨ Author field

A concept field whose value is the author dimension of a STAMP — who committed the version.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `AuthorField` | UUID: `a9210ad6-cc48-47df-86e5-2192d56704a6`

⊑ Author for edit coordinate

Individual or entity who made a particular edit or revision in a document (authoring a specific location or point in the codebase where an edit was made)

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `AuthorForEditCoordinate` | UUID: `337e93ba-531b-59a4-8153-57dca00e58d2`

⊑ Author for stamp coordinate

In individual or an entity responsible for defining or updating the values associated with the STAMP coordinate

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `AuthorForStampCoordinate` | UUID: `4fda23b8-b016-5d2a-97d5-7ff779d60701`

⊑∨ Author for version

Individual or entity who made a specific set of changes or modifications to a codebase/ terminology resulting in the creation of a new version or revision

```
⊑ Version Properties ⊓ Purpose
```

Parents: ⊑∨ VERSION PROPERTIES ⊑ PURPOSE

Since: Inception

id: `AuthorForVersion` | UUID: `4eb9de0d-7486-5f18-a9b4-82e3432f4103`

⊑∨ Axiom Expression

Why an OWL Axiom Syntax Pattern semantic exists: to express a component's definitional axioms as OWL functional-syntax text.

```
⊑ Logical expression model ⊓ Purpose
```

Parents: ⊑ Logical expression model ⊑ Purpose

Since: Inception

id: `AxiomExpression` | UUID: `8a080408-6f2a-560e-ab75-65477a0075d8`

⊑ Axiom focus

A statement or proposition that is assumed to be true without requiring proof, it serves as a foundation principles on which a system or theory is built. Focus refers to the central point of attention or concentration on a specific concept/axioms

```
⊑ Component type focus
```

Parents: ⊑ Component type focus

Since: Inception

id: `AxiomFocus` | UUID: `9c6fbddd-58bd-5881-b926-c813bbff849b`

⊑ Axiom origin

Groups the two origins a definition can have: stated, as an editor authored it, or inferred, as a reasoner derived it.

```
⊑ Axioms
```

Parents: ⊑ Axioms

Children: ⊑ Inferred premise type ⊑ Stated premise type

Since: Inception

id: `AxiomOrigin` | UUID: `b868bd83-5cd4-5d84-9cf7-b08674fbc79b`

⊑∨ Axiom Syntax

Groups the syntaxes a definition's axioms can be serialized in for exchange, such as OWL functional syntax.

```
⊑ Axioms ⊓ Meaning
```

Parents: ⊑ Axioms ⊑ Meaning

Children: ⊑∨ Express axiom syntax

Since: Inception

id: `AxiomSyntax` | UUID: `8da1c508-c2a2-4899-b26d-87f8b98a7558`

⊑∨ Axiomatized Component

What an OWL Axiom Syntax Pattern semantic's referenced component is: the component whose definitional axioms the semantic expresses — distinct from Axiom syntax, which names the syntax text the field holds.

```
⊑ Logical expression model ⊓ Meaning
```

Parents: ⊑ Logical expression model ⊑ Meaning

Since: Inception

id: `AxiomatizedComponent` | UUID: `94325002-4bf4-5236-ada7-0109d9682e08`

⊑ Axioms

Groups the logical-axiom meta-schema concepts: the terminology for expressing a concept's formal, machine-processable meaning — EL++ terminological axioms, interval-set axioms, and their syntax and origin.

```
⊑ Tinkar Model concept
```

Parents: ⊑ Tinkar Model concept

Children: ⊑ EL++ terminological axioms ⊑ Axiom origin ⊑ Interval Set Axioms ⊑∨ Axiom Syntax ⊑ Temporal Axiom

Since: Inception

id: `Axioms` | UUID: `20746b91-521a-45a6-89da-0fe32384a67f`

⊑ Blank Concept

The editor's empty-value marker: what an editing surface writes into a component-valued field a person has deliberately cleared, and what it tests for when asking whether that field is empty. An authored, persisted value -- distinct from Uninitialized Component, the platform's null-object for a reference that was never set at all. Neither asserts anything about the world; a claim that no value applies would need a concept this set does not yet carry.

```
⊑ Tinkar Model concept
```

Parents: ⊑ Tinkar Model concept

Since: Inception

id: `BlankConcept` | UUID: `cd23d88d-2fcd-4007-8829-97e37bf336aa`

⊑ Boolean

The retired dynamic-column type for boolean values, superseded by Boolean data type.

```
⊑ Dynamic column data types
```

Parents: ⊑ Dynamic column data types

Since: Inception

id: `Boolean` | UUID: `08f2fb74-980d-5157-b92c-4ff1eac6a506`

⊑ Boolean data type

The data type for fields holding a two-valued truth value: true or false.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `BooleanDataType` | UUID: `d6b9e2cc-31c6-5e80-91b7-7537690aae32`

⊑ ∨ Boolean default

The Boolean field's default: false — a flagged compromise. A two-valued type has no loud member, so no Boolean default can announce itself as unrevised; false is offered as the initial value with that limitation stated rather than hidden.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `BooleanDefault` | UUID: `3125e470-edb7-5498-b00b-6f53e45e09b3`

⊑ Boolean field

The field kind for boolean-valued fields: a pattern field holding true or false.

⊑ `Field definition data type field`

Parents: ⊑ Field definition data type field

Since: Inception

id: `BooleanField` | UUID: `4229683e-8772-4936-abd5-edc5a180f4d1`

⊑ Boolean literal

A literal whose value is true or false, as compared in a concrete-domain axiom.

⊑ `Literal value`

Parents: ⊑ Literal value

Since: Inception

id: `BooleanLiteral` | UUID: `49f41695-66a7-5471-846d-21c168f54c19`

⊑ Boolean reference

Reference(a pointer) to a Boolean object

⊑ `Query clauses`

Parents: ⊑ Query clauses

Since: Inception

id: `BooleanReference` | UUID: `de49d207-a26e-5f8a-b905-953a4dd13c21`

⊑ Boolean substitution

The process of replacing or substituting boolean values or expression in a logical context

⊑ `Field substitution`

Parents: ⊑  Field substitution

Since: Inception

id: `BooleanSubstitution` | UUID: `03559f9d-f1e4-5485-894b-4d457f145d54`

⊑∨  Branch Point

Why the origin instant is recorded: to fix the moment this path branched from its origin — the point up to which the origin path's versions are visible from this path.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑  Provenance model ⊑  Purpose

Since: Inception

id: `BranchPoint` | UUID: `80c6f90e-a85e-5be5-a447-380efdb0e37c`

⊑∨  Branch Source

Why the origin path value is recorded: to say which path this path branched from. The record's subject is the referenced component — the originated path — never this field.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑  Provenance model ⊑  Purpose

Since: Inception

id: `BranchSource` | UUID: `3f29da5f-a31f-5917-899e-e991032ca2e2`

⊑  Byte array

The retired dynamic-column type for raw byte sequences, superseded by Byte array data type.

```
⊑ Dynamic column data types
```

Parents: ⊑  Dynamic column data types

Since: Inception

id: `ByteArray` | UUID: `9a84fecf-708d-5de4-9c5f-e17973229e0f`

⊑  Byte array data type

The data type for fields holding a raw sequence of bytes, uninterpreted by the model -- an encoded document, an image, or any opaque payload.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `ByteArrayDataType` | UUID: `dbdd8df2-aec3-596b-88fc-7b83b5594a45`

⊑∨ ByteArray default

The ByteArray field's loud default: the UTF-8 bytes of "UNINITIALIZED" — the loud String default carried in byte form, never a silent empty array.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `ByteArrayDefault` | UUID: `4d6448a1-a5a7-5ed8-a10d-0cf1312cd413`

⊑ Canceled state

Concept used to represent a status for components that are canceled

```
⊑ Status value
```

Parents: ⊑∨ Status value

Since: Inception

id: `CanceledState` | UUID: `b42c1948-7645-5da8-a888-de6ec020ab98`

⊑ Case insensitive evaluation

Evaluates values regardless of the case

```
⊑ Text comparison measure semantic
```

Parents: ⊑ Text comparison measure semantic

Since: Inception

id: `CaseInsensitiveEvaluation` | UUID: `74bbdaff-f061-5807-b334-3c88ac3e9421`

⊑ Case sensitive evaluation

The text-comparison measure that distinguishes upper from lower case: two strings match only if their case agrees.

```
⊑ Text comparison measure semantic
```

Parents: ⊑ TEXT COMPARISON MEASURE SEMANTIC

Since: Inception

id: `CaseSensitiveEvaluation` | UUID: `a95e5dbc-a179-57f9-9cdd-6de8c026396d`

⊑∨ Case Sensitivity Rule

Why a description's case significance value is recorded: to say whether the description's text is case-sensitive.

```
⊑ Description model ⊓ Purpose
```

Parents: ⊑ DESCRIPTION MODEL ⊑ PURPOSE

Since: Inception

id: `CaseSensitivityRule` | UUID: `7c02e59b-7759-5cc2-a8aa-cc29dbf569f9`

⊑ Case significance concept nid for description

A field label which captures the case significance for a given concept description

```
⊑ Description version properties
```

Parents: ⊑ DESCRIPTION VERSION PROPERTIES

Since: Inception

id: `CaseSignificanceConceptNidForDescription` | UUID: `57271621-3f3c-58dd-8148-2674bc11b7e5`

⊑ Chinese language

The Chinese language, as a value for a description's language field: a language coordinate that names it selects Chinese-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `ChineseLanguage` | UUID: `aacbc859-e9a0-5e01-b6a9-9a255a47b0c9`

⊑ Chronicle and version model

The meta-schema of chronicles and their versions: the field kinds every chronicle shape records (public identity, the versions collection), the field kinds every version shape records (STAMP, the pattern-declaration and semantic-value slots), and the meaning/purpose concepts those shapes declare. This subsystem describes how knowledge is carried; what any of it is about belongs to the domain content itself.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Purpose Declaration ⊑ Any component ⊑ Semantic properties ⊑∨ Version Properties ⊑∨ Set membership ⊑ Uninitialized Component ⊑∨ Version Snapshot ⊑ Chronicle properties ⊑∨ Meaning Declaration ⊑∨ Pattern Membership ⊑∨ Attachment Target ⊑∨ Chronicle Identity and History ⊑∨ Version History

Since: Inception

id: `ChronicleAndVersionModel` | UUID: `863dfaec-a508-5c9f-a51f-d8691ab38e8b`

⊑∨ Chronicle Identity and History

Why a chronicle-shape pattern's referenced component carries this semantic: to record that component's own identity (public id) alongside its complete version history.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ Chronicle and version model ⊑ Purpose

Since: Inception

id: `ChronicleIdentityAndHistory` | UUID: `d00b0f8b-05ea-58c5-be24-b6ea54f1de86`

⊑ Chronicle properties

Attributes or characteristic associated with a historical record or an account of events (metadata, timestamps)

```
⊑ Chronicle and version model
```

Parents: ⊑ Chronicle and version model

Children: ⊑ UUID LIST FOR COMPONENT ⊑ PRIMORDIAL UUID FOR CHRONICLE ⊑ VERSION LIST FOR CHRONICLE ⊑ SEMANTIC LIST FOR CHRONICLE

Since: Inception

id: `ChronicleProperties` | UUID: `2ba2ef47-30af-57ec-9073-38693f020d7e`

⊑ Classifier for logic coordinate

The logic-coordinate dimension naming which reasoner owns the inferred content a view resolves; inferred semantics are stamped with the classifier as their author.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `ClassifierForLogicCoordinate` | UUID: `4b90e89d-2a0e-5ca3-8ae5-7498d148a9d2`

⊑∨ Comment

A filed label to capture free text information which may be necessary to add or change (concepts, relationships, semantics, etc)

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `Comment` | UUID: `147832d4-b9b8-5062-8891-19f9c4e4760a`

P Comment pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ SUBJECT OF COMMENTARY	⊑∨ EDITORIAL CLARIFICATION	—
Field 1		
⊑∨ COMMENT	⊑∨ EDITORIAL CLARIFICATION	⊑ STRING DATA TYPE

Since: Inception

id: `CommentPattern` | UUID: `3734fb0a-4c14-5831-9a61-4743af609e7a`

⊑∨ Commit Provenance

Why a STAMP pattern semantic exists: to record who committed a version, in what state, module, path, and when.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑ Provenance model ⊑ Purpose

Since: Inception

id: `CommitProvenance` | UUID: `3e03a324-4a64-55ec-915c-cf56f97beef5`

P Component Chronology Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Component field	⊑∨ Chronicle Identity and History	—
Field 1		
⊑∨ Public ID field	⊑∨ Uniquely identify knowledge graph components	⊑ Componen
Field 2		
⊑∨ Component versions field	⊑∨ Component versions set	⊑ Componen

Since: Inception

id: `ComponentChronologyPattern` | UUID: `c48db76d-5eb0-4ff5-84d0-5c3c4ec77767`

⊑ Component data type

A field that identifies any component -- concept, semantic, pattern, or STAMP -- by reference, not restricted to concepts.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `ComponentDataType` | UUID: `fb00d132-fcc3-5cbf-881d-4bcc4b4c91b3`

⊑∨ Component default

The Component field's loud default: Uninitialized Component (SOLOR), the base set's own loud placeholder for a component not yet chosen. Not Blank Concept — Blank means a deliberate not-applicable, while Uninitialized means a value still owed.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `ComponentDefault` | UUID: `153b82c3-6210-5833-a124-19e773b21d09`

⊑∨ Component field

The field kind whose value references a component of any kind — concept, pattern, semantic, or STAMP. The parent of every reference-valued field kind, and the Component Chronology Pattern's referenced-component meaning: the component whose identity and history the chronicle records.

```
⊑ Field categories ⊓ Meaning
```

Parents: ⊑ FIELD CATEGORIES ⊑ MEANING

Children: ⊑∨ CONCEPT FIELD ⊑∨ SEMANTIC REFERENCED COMPONENT FIELD ⊑∨ STAMP FIELD ⊑∨ SEMANTIC FIELD ⊑∨ PATTERN FIELD

Since: Inception

id: `ComponentField` | UUID: `8bd36a0c-d05d-46b7-a79a-d11477705cc1`

⊑ Component for semantic

The field naming the component a semantic is attached to, in the semantic-properties vocabulary. Duplicates Referenced component for semantic and the live Semantic referenced component field kind, which is what pattern declarations actually use; retained for identity compatibility.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `ComponentForSemantic` | UUID: `0bc32c16-698e-5719-8bd5-efa099c7d782`

⊑ Component Id list data type

An ordered list of component ids.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `ComponentIdListDataType` | UUID: `e553d3f1-63e1-4292-a3a9-af646fe44292`

⊑ Component Id set data type

An unordered set of component ids.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `ComponentIdSetDataType` | UUID: `e283af51-2e8f-44fa-9bf1-89a99a7c7631`

⊑ Component semantic

The semantic type whose field value is a component reference.

```
⊑ Semantic type
```

Parents: ⊑ Semantic type

Since: Inception

id: `ComponentSemantic` | UUID: `127e7274-0b88-5519-a9db-85d4b9ce6a4a`

⊑ Component type focus

Groups the component kinds a rule or action can be scoped to: concepts, descriptions, or axioms.

```
⊑ Tinkar Model concept
```

Parents: ⊑ Tinkar Model concept

Children: ⊑ Concept focus ⊑ Description focus ⊑ Axiom focus

Since: Inception

id: `ComponentTypeFocus` | UUID: `f1f179d0-26af-5123-9b29-9fc6cd01dd29`

P Component Version Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Component versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type

Since: Inception

id: `ComponentVersionPattern` | UUID: `a38b7d2d-8fa5-4206-9185-a1af9f81be2c`

⊑∨ Component versions field

The field kind holding a chronicle's version collection — one entry per committed version of the component. Every chronicle-shape pattern declares it beside the public id, and the Component Version Pattern carries it as its own referenced-component meaning.

```
⊑ Field categories ⊓ Meaning
```

Parents: ⊑ Field categories ⊑ Meaning

Children: ⊑∨ Concept versions field ⊑∨ Pattern versions field
⊑∨ STAMP versions field ⊑∨ Semantic versions field

Since: Inception

id: `ComponentVersionsField` | UUID: `1a852426-422a-48db-a618-c906ac4c8e6c`

⊑∨ Component versions set

Why a chronicle's versions field is recorded: to hold every version the component has ever had — the complete history a chronicle exists to keep alongside its identity.

```
⊑ Field categories ⊓ Purpose
```

Parents: ⊑ Field categories ⊑ Purpose

Children: ⊑∨ Semantic versions set ⊑∨ STAMP versions set ⊑∨ Pattern versions set ⊑∨ Concept versions set

Since: Inception

id: `ComponentVersionsSet` | UUID: `54d670f1-234d-485a-a354-e1fa7eea1bf2`

⊑∨ ComponentIdList default

The ComponentIdList field's loud default: the singleton list holding Uninitialized Component — present and non-empty (an empty list would be a silent zero-value), its one entry itself the loud placeholder.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `ComponentIdListDefault` | UUID: `9b2ebb85-a651-5fa7-8636-9239fe354f3e`

⊑∨ ComponentIdSet default

The ComponentIdSet field's loud default: the singleton set holding Uninitialized Component — present and non-empty (an empty set would be a silent zero-value), its one member itself the loud placeholder.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `ComponentIdSetDefault` | UUID: `28cb73e5-68d2-5fa3-8b9b-bc2142ecbffe`

⊑ Concept constraints

The retired action parameter holding the filters a rule-driven action applied when selecting which concepts to act on.

```
⊑ Action properties
```

Parents: ⊑ Action properties

Since: Inception

id: `ConceptConstraints` | UUID: `081273cd-92dd-593c-9d9b-63d33838e70b`

⊑ Concept data type

Field for the human readable description for the given concept

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `ConceptDataType` | UUID: `ac8f1f54-c7c6-5fc7-b1a8-ebb04b918557`

⊑ ∨ Concept default

The Concept field's loud default: Uninitialized Component (SOLOR) — the loud placeholder for a concept not yet chosen, obviously needing revision wherever it appears.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `ConceptDefault` | UUID: `eb902228-e21c-57e2-a66c-b7db1235c9e5`

⊑ Concept details tree table

The Komet surface presenting a concept's details as a tree table: descriptions, axioms, and attached semantics in one expandable view.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Since: Inception

id: `ConceptDetailsTreeTable` | UUID: `1655edd8-7b73-52c5-98b0-263d1ab3a90b`

⊑ ∨ Concept field

A component field narrowed to concepts: its value references a concept specifically. The parent of the STAMP-dimension field kinds and of a pattern's meaning- and purpose-declaration field kinds, and the Concept Chronology Pattern's referenced-component meaning.

```
⊑ Component field ⊓ Meaning
```

Parents: ⊑∨ Component field ⊑ Meaning

Children: ⊑∨ Value-set field ⊑∨ Path field ⊑ Field definition purpose field
⊑∨ Constraint anchor concept ⊑∨ Status field ⊑∨ Constrained field
⊑∨ Roster author ⊑∨ Author field ⊑∨ Constraint kind
⊑∨ Pattern meaning field ⊑ Field definition meaning field ⊑∨ Module field
⊑∨ Preferred reviewer ⊑∨ Pattern purpose field ⊑∨ Value-set pattern

Since: Inception

id: `ConceptField` | UUID: `ebe2aa74-f100-41b2-8d75-2d8f06ce5e4e`

P Concept field pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Concept field	⊑∨ Chronicle Identity and History	—
Field 1		
⊑∨ Public ID field	⊑∨ Uniquely identify knowledge graph components	⊑ Component d
Field 2		
⊑∨ Concept versions field	⊑∨ Concept versions set	⊑ Component I

Since: Inception

id: `ConceptFieldPattern` | UUID: `3e510cb9-1666-4676-9334-d288a56bf155`

⊑ Concept focus

The component-type focus selecting concepts: an action or rule scoped to concept components.

```
⊑ Component type focus
```

Parents: ⊑ Component type focus

Since: Inception

id: `ConceptFocus` | UUID: `dca9854d-9e4c-5e8a-8b30-6c1af6901bb8`

⊑∨ Concept pattern for logic coordinate

The pattern whose active semantics enumerate the SOLOR concepts a Logic Coordinate reasons over.

```
⊑ ImmutableCoordinate Properties ⊓ Meaning
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES ⊑ MEANING

Since: Inception

id: `ConceptPatternForLogicCoordinate` | UUID: `16486419-5d1c-574f-bde6-21910ad66f44`

⊑ Concept reference

A field to capture a reference to validate concept

```
⊑ Atom
```

Parents: ⊑ ATOM

Since: Inception

id: `ConceptReference` | UUID: `e89148c7-4fe2-52f8-abb9-6a53605d20cb`

⊑ Concept semantic

The semantic type whose field value is a concept reference: a semantic that points at one concept.

```
⊑ Semantic type
```

Parents: ⊑ SEMANTIC TYPE

Since: Inception

id: `ConceptSemantic` | UUID: `fbf054fb-ceaf-5ab8-b946-bbcc4835ce07`

⊑ Concept substitution

A field substitution whose substituted value is a concept reference; part of the retired ISAAC field-substitution family.

```
⊑ Field substitution
```

Parents: ⊑ FIELD SUBSTITUTION

Since: Inception

id: `ConceptSubstitution` | UUID: `23483990-b738-5f43-bc03-befd43928a37`

⊑ Concept to find

Find concept (if searching on Komet shows us the results 'details and further information?)

```
⊑ Action properties
```

Parents: ⊑ ACTION PROPERTIES

Since: Inception

id: `ConceptToFind` | UUID: `91687b29-3bbb-540b-9de6-91246c75afd0`

⊑ Concept type

A field that captures a defined concept label

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Children: ⊑ SEMANTIC FIELD CONCEPTS ⊑ ANONYMOUS CONCEPT ⊑ ∨ PATH CONCEPT

Since: Inception

id: `ConceptType` | UUID: `106f3ba1-63b8-5596-8dbe-524fa2e89fc0`

⊑ Concept version

A filed that captures the version of the terminology that it came from

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Since: Inception

id: `ConceptVersion` | UUID: `c202f992-3f4b-5f30-9b32-e376f68367d1`

P Concept Version Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		

Meaning	Purpose	Data type
⊑∨ Concept versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type

Since: Inception

id: `ConceptVersionPattern` | UUID: `7943a5f1-538b-4fda-8acb-019e0bec125b`

⊑∨ Concept versions field

The component versions field narrowed to concept chronicles: its value collects a concept's own versions. The Concept Version Pattern's referenced-component meaning.

```
⊑ Component versions field ⊓ Meaning
```

Parents: ⊑∨ Component versions field ⊑ Meaning

Since: Inception

id: `ConceptVersionsField` | UUID: `3a08b5f1-f17e-4db5-8cf9-c6540f26f241`

⊑∨ Concept versions set

Why a concept chronicle's versions field is recorded: to hold every version the concept has ever had — the concept-chronicle specialization of Component versions set.

```
⊑ Component versions set ⊓ Purpose
```

Parents: ⊑∨ Component versions set ⊑ Purpose

Since: Inception

id: `ConceptVersionsSet` | UUID: `806c7f9f-52f9-4b53-9758-122899b28a76`

⊑∨ Concrete value operator

Groups the comparison operators a concrete-domain axiom uses to relate a measured quantity to a literal bound: equal, less than, greater than, and their inclusive forms.

```
⊑ Tinkar Model concept ⊓ Purpose
```

Parents: ⊑ Tinkar Model concept ⊑ Purpose

Children: ⊑ Less than or equal to ⊑ Equal to ⊑ Greater than or equal to ⊑ Greater than ⊑ Less than ⊑∨ Maximum Value Operator ⊑∨ Minimum Value Operator

Since: Inception

id: `ConcreteValueOperator` | UUID: `843b0b55-8785-5544-93f6-581da9cf1ff3`

⊑ Conditional triggers

Conditional triggers based on actions, reasoner

```
⊑ Action properties
```

Parents: ⊑ Action properties

Since: Inception

id: `ConditionalTriggers` | UUID: `a3e4ac54-db82-5345-8713-7e0da98bbb0a`

⊑ Connective operator

A field that captures what the operator is (logical connective)

```
⊑ Atom
```

Parents: ⊑ Atom

Children: ⊑ And ⊑ Or

Since: Inception

id: `ConnectiveOperator` | UUID: `3fdcaadc-d972-58e9-84f1-b3a39903b076`

⊑∨ Constrained field

The field-meaning concept of the pattern field this constraint governs.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ Concept field ⊑ Meaning

Since: Inception

id: `ConstrainedField` | UUID: `45756d94-080b-5a6b-a751-12af64cc7000`

⊑∨ Constrained Pattern

The referenced-component role a field constraint semantic's attachment target plays: the pattern that has one of its own fields constrained — one attachment role, shared by the Taxonomy Field Constraint Pattern and the Value-set Field Constraint Pattern, and distinct from the constraint mechanisms themselves.

```
⊑ Constraint model ⊓ Meaning
```

Parents: ⊑ CONSTRAINT MODEL ⊑ MEANING

Since: Inception

id: `ConstrainedPattern` | UUID: `548d102c-d84a-5135-a45a-b7d35d9718cd`

⊑∨ Constraint anchor concept

The concept a kind-of, descendant, leaf-descendant, or immediate-child constraint is relative to.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `ConstraintAnchorConcept` | UUID: `e1782109-1a60-5bb5-a7e4-9742e06d62c2`

⊑∨ Constraint kind

Which Taxonomy field constraint kind this constraint semantic expresses.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `ConstraintKind` | UUID: `4b623f76-1a22-5bdd-9e1a-b1e3c1cbd829`

⊑ Constraint model

The meta-schema of field constraints (IKE-Network/ike-issues#890): the constraint-declaration roles (constrained pattern and field, rule, scope, anchor), the taxonomy constraint kinds, the value-set apparatus with its member match relations, and the restriction concepts constraint patterns declare as meanings and purposes. Constraints are first-class pattern content — declared, versioned, and exchanged like any other knowledge.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Member match relation ⊑ Taxonomy field constraint kind ⊑∨ Numeric Value Restriction ⊑∨ Constraint Scope ⊑∨ Constrained Pattern ⊑∨ Taxonomy Reference Point ⊑∨ Match Rule ⊑∨ Value Disambiguation ⊑∨ Unit Example ⊑∨ Constraint Rule ⊑∨ Legal Value Source ⊑∨ Reference Range Authority ⊑∨ Field Value Restriction

Since: Inception

id: `ConstraintModel` | UUID: `8147fe74-0e97-5568-b121-244a68902afd`

⊑∨ Constraint Rule

Why the Constraint kind value is recorded: to say which taxonomy-relative rule — kind-of, descendant, leaf-descendant, or immediate-child — applies.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `ConstraintRule` | UUID: `179e8846-380f-54cc-8eef-27c1820e5e9e`

⊑∨ Constraint Scope

Why the Constrained field value is recorded: to say which field of the constrained pattern this rule applies to.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `ConstraintScope` | UUID: `13c213d9-b5f2-5b02-ab9f-452724ead9f0`

⊑ Correlation expression

The subject side of a correlation: the expression being checked for correspondence against a reference expression.

```
⊑ Correlation properties
```

Parents: ⊑ Correlation properties

Since: Inception

id: `CorrelationExpression` | UUID: `1711815f-99a1-5d1a-8f1e-75dc7bf41928`

⊑ Correlation properties

Characteristics or measures that describe the relationship between two or more variables

⊑ `Legacy model concepts`

Parents: ⊑ LEGACY MODEL CONCEPTS

Children: ⊑ CORRELATION REFERENCE EXPRESSION ⊑ CORRELATION EXPRESSION

Since: Inception

id: `CorrelationProperties` | UUID: `8f682e00-3d9e-5506-bd19-b2169d6c8752`

⊑ Correlation reference expression

The reference side of a correlation: the expression an authored expression is compared against when checking whether two expressions correspond.

⊑ `Correlation properties`

Parents: ⊑ CORRELATION PROPERTIES

Since: Inception

id: `CorrelationReferenceExpression` | UUID: `acb73d95-7c96-590c-9f24-23da54f95ff2`

⊑ Creative Commons BY license

Creative Commons (CC) licenses are a set of public copyright licenses that enable the free distribution of an otherwise copyrighted work

⊑ `Provenance model`

Parents: ⊑ PROVENANCE MODEL

Since: Inception

id: `CreativeCommonsBYLicense` | UUID: `3415a972-7850-57cd-aa86-a572ca1c2ceb`

⊑ Czech dialect

The Czech dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in Czech usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Since: Inception

id: `CzechDialect` | UUID: `6979e268-0b59-542f-bac0-313c5ddf6a2e`

⊑ Czech language

The Czech language, as a value for a description's language field: a language coordinate that names it selects Czech-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `CzechLanguage` | UUID: `33aa2d26-0541-557c-b796-904cbf245101`

⊑ Danish language

The Danish language, as a value for a description's language field: a language coordinate that names it selects Danish-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `DanishLanguage` | UUID: `987681fb-f3ef-595d-90e2-067baf2bc71f`

⊑ Data Concept

Groups concepts that represent data values and defaults — the data-carrying side of the model, as distinct from Tinkar Model concept's structural, self-descriptive side.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑ Default Data Concept

Since: Inception

id: `DataConcept` | UUID: `ae7069d1-67fa-4470-a56f-0d24a8fcea83`

⊑ Data property set

The logical set grouping data-property axioms under a definition root. Mirrors LogicalAxiom.LogicalSet.DataPropertySet.

```
⊑ Logical set
```

Parents: ⊑ Logical set

Since: Inception

id: `DataPropertySet` | UUID: `6b8ed642-de72-4aee-953d-42e5db92c0ab`

⊑ Data Property Set Axioms

A baseline grouping for data-property-set axioms, retained for identity compatibility: the platform's set meaning is Data property set.

```
⊑ Logical set
```

Parents: ⊑ Logical set

Since: Inception

id: `DataPropertySetAxioms` | UUID: `1402d311-0b4b-4014-81d2-e715c6696346`

⊑ ∨ Data type default exemplar

Why every Data Type Defaults Pattern field value is recorded: to carry its data type's loud default — a value that works as an initial value but reads as obviously needing revision, never Java's silent zero-value. One purpose serves all sixteen fields because every field exists for this identical reason; distinct purposes would be manufactured variety.

```
⊑ Defaults and templates model ⊓ Purpose
```

Parents: ⊑ Defaults and templates model ⊑ Purpose

Since: Inception

id: `DataTypeDefaultExemplar` | UUID: `ffbec350-7b66-5efd-a2ba-b972a1ae3870`

⊑∨ Data Type Default Provision

Why a Data Type Defaults Pattern semantic exists: to provide, in one semantic, a loud default value for every data type a pattern field can declare — and, replayed, to exercise every field serialization path the store supports end to end.

```
⊑ Defaults and templates model ⊓ Purpose
```

Parents: ⊑ Defaults and templates model ⊑ Purpose

Since: Inception

id: `DataTypeDefaultProvision` | UUID: `96b4fd68-4378-545d-9d4e-926283161a31`

P Data Type Defaults Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Defaulted Data Types	⊑∨ Data Type Default Provision	—
Field 1		
⊑∨ String default	⊑∨ Data type default exemplar	⊑ String data type
default UNINITIALIZED		
Field 2		
⊑∨ Component default	⊑∨ Data type default exemplar	⊑ Component data type
default ⊑ Uninitialized Component		
Field 3		
⊑∨ ComponentIdSet default	⊑∨ Data type default exemplar	⊑ Component Id set data type
default ⊑ Uninitialized Component		
Field 4		
⊑∨ ComponentIdList default	⊑∨ Data type default exemplar	⊑ Component Id list data type
default ⊑ Uninitialized Component		

Meaning	Purpose	Data type
Field 5		
⊑∨ DiTree default	⊑∨ Data type default exemplar	⊑ DiTree data type
default 1-vertex tree		
Field 6		
⊑∨ DiGraph default	⊑∨ Data type default exemplar	⊑ DiGraph data type
default 2-vertex cycle		
Field 7		
⊑∨ Concept default	⊑∨ Data type default exemplar	⊑ Concept data type
default ⊑ *Uninitialized Component*		
Field 8		
⊑∨ Semantic default	⊑∨ Data type default exemplar	⊑ Semantic data type
default Uninitialized Component (SOLOR)		
Field 9		
⊑∨ Integer default	⊑∨ Data type default exemplar	⊑ Integer data type
default 777777777		
Field 10		
⊑∨ Float default	⊑∨ Data type default exemplar	⊑ Float data type
default NaN		
Field 11		
⊑∨ Boolean default	⊑∨ Data type default exemplar	⊑ Boolean data type
default false		
Field 12		
⊑∨ ByteArray default	⊑∨ Data type default exemplar	⊑ Byte array data type

Meaning	Purpose	Data type
default UNINITIALIZED		
Field 13		
⊑∨ Array default	⊑∨ Data type default exemplar	⊑ Array data type
default [UNINITIALIZED]		
Field 14		
⊑∨ Instant default	⊑∨ Data type default exemplar	⊑ Instant literal
default Pre-inception		
Field 15		
⊑∨ Long default	⊑∨ Data type default exemplar	⊑ Long
default 777777777777777777		
Field 16		
⊑∨ Decimal default	⊑∨ Data type default exemplar	⊑ Decimal data type
default 777777777.777		

Since: Inception

id: `DataTypeDefaultsPattern` | UUID: `e604e510-ffd6-5f04-ab6e-50842dedd43a`

⊑ Decimal

The retired dynamic-column type for high-precision decimal values, superseded by Decimal data type.

⊑ `Dynamic column data types`

Parents: ⊑ Dynamic column data types

Since: Inception

id: `Decimal` | UUID: `dccb0476-3b63-3d48-b5a2-85bd0ad2fa61`

⊑ Decimal data type

Represents values as high-precision decimal values.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `DecimalDataType` | UUID: `b413fe94-4ada-4aee-96f9-22be19699d40`

⊑∨ Decimal default

The Decimal field's loud default: the stretched-sevens sentinel 777777777.777 — nine sevens, a decimal point, three more sevens. The decimal point placement deliberately demonstrates the type's distinguishing property (a scaled decimal, not an integer) — the same move as the DiGraph cycle proving graph-ness. Never Java's silent zero.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `DecimalDefault` | UUID: `3eadc612-bc06-51c8-8a55-323d57c95d0b`

⊑ Default Data Concept

The baseline's placeholder anchor for default data, retained for identity compatibility with upstream stores; its placeholder seeds are deliberately not adopted here.

```
⊑ Data Concept
```

Parents: ⊑ Data Concept

Since: Inception

id: `DefaultDataConcept` | UUID: `4a32d2ad-baca-42b5-a432-4c4ae6431668`

⊑ Default module for edit coordinate

The edit-coordinate dimension naming the module a new edit is stamped with, unless the editor chooses another from the options.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `DefaultModuleForEditCoordinate` | UUID: `e83d322c-e275-5392-a5db-1de5fe98acb5`

⊑ Default value concept

The attachment point for default values: a default value semantic of a pattern is a complete semantic of that pattern whose referenced component is this concept and whose fields carry the default values an editor offers for that pattern's fields. Attachment here is a support declaration only — never a domain assertion about this concept or any other component — and this concept never joins a domain member set.

```
⊑ Defaults and templates model
```

Parents: ⊑ Defaults and templates model

Since: Inception

id: `DefaultValueConcept` | UUID: `b3af49c5-9e8a-5b94-94d4-b1e2b72e1c2f`

⊑∨ Defaulted Data Types

What a Data Type Defaults Pattern semantic means: the sixteen data types ConceptToDataType recognizes for pattern fields, each exemplified by one field carrying that type's loud default.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `DefaultedDataTypes` | UUID: `a7b1ea9b-1297-566a-a175-e9b3096175c7`

⊑ Defaults and templates model

The meta-schema of authoring support content (IKE-Network/ike-issues#885): the attachment points default value and template semantics reference, the per-data-type default concepts, and the exemplar/provision concepts the Data Type Defaults apparatus declares. Support content itself lives in the Defaults and templates module; these are the foundation concepts that describe it.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Defaulted Data Types ⊑∨ Long default ⊑∨ Concept default ⊑∨ Boolean default ⊑∨ Array default ⊑∨ Semantic default ⊑∨ DiGraph default ⊑∨ ByteArray default ⊑∨ ComponentIdList default ⊑∨ String default ⊑∨ Integer default ⊑∨ ComponentIdSet default ⊑∨ Decimal default ⊑∨ Float default ⊑∨ DiTree default ⊑∨ Component default ⊑∨ Instant default ⊑ Default value concept ⊑ Template concept ⊑∨ Data Type Default Provision ⊑∨ Data type default exemplar

Since: Inception

id: `DefaultsAndTemplatesModel` | UUID: `78ea9f07-028e-5e7a-8d9c-993947a663dc`

⊑ Defaults and templates module

The module scoping every default value and template semantic. The boundary is bidirectional and live-and-die: every version of a defaults or template chronology lives in this module, and this module holds only defaults and template content — a violation in either direction is an error to be withdrawn, never adopted as precedent. As the export dimension, module crossed with path selects a user's preferences for exchange: include this module to carry defaults and templates alongside domain content, exclude it to leave them behind.

```
⊑ Module
```

Parents: ⊑∨ Module

Since: Inception

id: `DefaultsAndTemplatesModule` | UUID: `8d013392-1db6-58cd-a1cb-74a1e603399a`

⊑ Definition description type

Semantic value describing the description type for the description pattern is a definition

```
⊑ Description type
```

Parents: ⊑∨ Description type

Since: Inception

id: `DefinitionDescriptionType` | UUID: `700546a3-09c7-3fc2-9eb9-53d318659a09`

⊑ Definition root

The root vertex of a definition graph: every stated or inferred definition begins here, and its children are the definition's logical sets. Mirrors LogicalAxiom.DefinitionRoot.

```
⊑ Logical axiom
```

Parents: ⊑ LOGICAL AXIOM

Since: Inception

id: `DefinitionRoot` | UUID: `e7271c01-6ed4-5240-963f-34d1f24153b0`

⊑ Descendant field constraint

Legal values are every descendant of the anchor concept, not including the anchor concept itself.

```
⊑ Taxonomy field constraint kind
```

Parents: ⊑ TAXONOMY FIELD CONSTRAINT KIND

Since: Inception

id: `DescendantFieldConstraint` | UUID: `f503b7bc-84ff-5fab-b5a1-384b831894fa`

⊑∨ Description

Human-readable text attached to a component: the words a person reads where the machinery holds an identity. Every rendering surface resolves one, choosing by language, dialect, and description type.

```
⊑ Tinkar Model concept ⊓ Purpose
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ PURPOSE

Since: Inception

id: `Description` | UUID: `87118daf-d28c-55fb-8657-cd6bc8425600`

⊑∨ Description acceptability

Whether a given human readable text for a concept is permissible

```
⊑ Tinkar Model concept ⊓ Meaning ⊓ Purpose
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING ⊑ PURPOSE

Children: ⊑ PREFERRED ⊑ ACCEPTABLE

Since: Inception

id: `DescriptionAcceptability` | UUID: `96b61063-0d29-5aea-9652-3f5f328aadc3`

⊑∨ Description Attachment

Why a Description Pattern semantic exists: to attach a human-readable name or definition to a concept.

```
⊑ Description model ⊓ Purpose
```

Parents: ⊑ DESCRIPTION MODEL ⊑ PURPOSE

Since: Inception

id: `DescriptionAttachment` | UUID: `179bfdd3-4e9c-55ae-86b9-47214fcfcc4a`

⊑ Description case sensitive

Assumes the description is dependent on capitalization

```
⊑ Description case significance
```

Parents: ⊑∨ DESCRIPTION CASE SIGNIFICANCE

Since: Inception

id: `DescriptionCaseSensitive` | UUID: `0def37bc-7e1b-384b-a6a3-3e3ceee9c52e`

⊑∨ Description case significance

Specifies how to handle the description text in terms of case sensitivity

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Children: ⊑ NOT APPLICABLE ⊑ DESCRIPTION INITIAL CHARACTER CASE SENSITIVE ⊑ DESCRIPTION CASE SENSITIVE ⊑ DESCRIPTION NOT CASE SENSITIVE

Since: Inception

id: `DescriptionCaseSignificance` | UUID: `c3dde9ea-b144-5f49-845a-20cc7d305250`

⊑ Description core type

Used to mark non-snomed descriptions as one of the core snomed types

```
⊑ Description type
```

Parents: ⊑∨ DESCRIPTION TYPE

Since: Inception

id: `DescriptionCoreType` | UUID: `351955ff-30f4-5806-a0a5-5dda79756377`

⊑ Description dialect pair

The pairing of a dialect with a description: the two-field shape a dialect semantic records, saying how acceptable one description is in one dialect. Its two children name the sides of that pair.

```
⊑ Description version properties
```

Parents: ⊑ DESCRIPTION VERSION PROPERTIES

Children: ⊑ DESCRIPTION FOR DIALECT/DESCRIPTION PAIR
⊑ DIALECT FOR DIALECT/DESCRIPTION PAIR

Since: Inception

id: `DescriptionDialectPair` | UUID: `a27bbbf8-57b5-590c-8650-1247f6f963eb`

⊑ Description focus

The component-type focus selecting descriptions: an action or rule scoped to description components.

```
⊑ Component type focus
```

Parents: ⊑ COMPONENT TYPE FOCUS

Since: Inception

id: `DescriptionFocus` | UUID: `6edf734d-7f57-5430-9164-6ee0824fd94b`

⊑ Description for dialect/description pair

The description side of a dialect-and-description pair: which description the paired acceptability judgment is about.

```
⊑ Description dialect pair
```

Parents: ⊑ DESCRIPTION DIALECT PAIR

Since: Inception

id: `DescriptionForDialectDescriptionPair` | UUID: `1137767a-ad8b-5bc5-9842-a1f9b09d1ecc`

⊑ Description initial character case sensitive

Value which designates initial character as sensitive for a given description

```
⊑ Description case significance
```

Parents: ⊑∨ DESCRIPTION CASE SIGNIFICANCE

Since: Inception

id: `DescriptionInitialCharacterCaseSensitive` | UUID: `17915e0d-ed38-3488-a35c-cda966db306a`

⊑ Description list for concept

The field holding every description attached to a concept, so a rendering surface can choose among them by language, dialect, and type.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Since: Inception

id: `DescriptionListForConcept` | UUID: `ab3e8771-7c7c-5e57-8acf-147b16da36e2`

⊑ Description logic profile for logic coordinate

The logic-coordinate dimension naming which description-logic profile the reasoner assumes: the coordinate slot whose value is a profile such as EL++ logic profile, as distinct from the profile family itself.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `DescriptionLogicProfileForLogicCoordinate` | UUID: `aef80e34-b2dd-5dca-a989-3e0ee2699be3`

⊑ Description model

The meta-schema of human-language rendering: how a component's descriptions attach, which roles a description plays (fully qualified name, regular name, definition), and the rules — case sensitivity, dialect acceptability — a rendering surface applies when choosing what to show.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑∨ Description Attachment ⊑∨ Description Role ⊑∨ Case Sensitivity Rule

Since: Inception

id: `DescriptionModel` | UUID: `21a76cff-6404-5164-95be-d4be98778c57`

⊑ Description not case sensitive

Value which designate character as not sensitive for a given description

```
⊑ Description case significance
```

Parents: ⊑∨ Description case significance

Since: Inception

id: `DescriptionNotCaseSensitive` | UUID: `ecea41a2-f596-3d98-99d1-771b667e55b8`

P Description Pattern

Contains all metadata and human readable text that describes the concept

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Description semantic	⊑∨ Description Attachment	—
e.g. ⊑ Uninitialized Component		
Field 1		
⊑∨ Language for description	⊑∨ Language	⊑ Component data type
e.g. ⊑ English Language		
Field 2		
⊑∨ Text for description	⊑∨ Description	⊑ String data type
e.g. Uninitialized Component (SOLOR)		
Field 3		

Meaning	Purpose	Data type
⊑∨ Description case significance	⊑∨ Case Sensitivity Rule	⊑ Component data type
e.g. ⊑ *Description not case sensitive*		
Field 4		
⊑∨ Description type	⊑∨ Description Role	⊑ Component data type
e.g. ⊑ *Fully qualified name description type*		

Since: Inception

id: `DescriptionPattern` | UUID: `a4de0039-2625-5842-8a4c-d1ce6aebf021`

⊑∨ Description Role

Why a description's type value is recorded: to say which of the three description roles — fully qualified name, regular name, or definition — this description plays.

```
⊑ Description model ⊓ Purpose
```

Parents: ⊑ Description model ⊑ Purpose

Since: Inception

id: `DescriptionRole` | UUID: `d0ac5edc-5467-5048-8934-787812063d0d`

⊑∨ Description semantic

Purpose and meaning for the description pattern and dialect patterns

```
⊑ Tinkar Model concept ⊓ Meaning ⊓ Purpose
```

Parents: ⊑ Tinkar Model concept ⊑ Meaning ⊑ Purpose

Since: Inception

id: `DescriptionSemantic` | UUID: `81487d5f-6115-51e2-a3b3-93d783888eb8`

⊑∨ Description type

The role a description plays for its component: fully qualified name, regular name, or definition. A rendering surface picks by type before picking by dialect.

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Children: ⊑ DESCRIPTION CORE TYPE ⊑ DEFINITION DESCRIPTION TYPE ⊑ FULLY QUALIFIED NAME DESCRIPTION TYPE ⊑ INVERSE NAME ⊑ REGULAR NAME DESCRIPTION TYPE ⊑ EXTENDED DESCRIPTION TYPE

Since: Inception

id: `DescriptionType` | UUID: `ad0c19e8-2ccc-59c1-8b7e-c56c03aca8eb`

⊑ Description type for description

Why a description version records its type: to say which role the description plays for its component -- fully qualified name, regular name, or definition -- so a rendering surface can pick by role.

```
⊑ Description version properties
```

Parents: ⊑ DESCRIPTION VERSION PROPERTIES

Since: Inception

id: `DescriptionTypeForDescription` | UUID: `a00c5ad7-5b8a-5592-a28c-64057dd3caab`

⊑ Description type preference list for language coordinate

The language-coordinate dimension holding description types in preference order: which type a surface renders first when a component carries several.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `DescriptionTypePreferenceListForLanguageCoordinate` | UUID: `44c7eab6-fdb8-5427-9d7a-52ab63f7a6f9`

⊑ Description version properties

Combination of terms that might be used in a specific context or domain

```
⊑ Version Properties
```

Parents: ⊑ ∨ VERSION PROPERTIES

Children: ⊑ DESCRIPTION TYPE FOR DESCRIPTION ⊑∨ LANGUAGE FOR DESCRIPTION ⊑ CASE SIGNIFICANCE CONCEPT NID FOR DESCRIPTION ⊑ DESCRIPTION DIALECT PAIR

Since: Inception

id: `DescriptionVersionProperties` | UUID: `732aad24-4add-59d6-bbc9-840a8b9dc754`

⊑ Description-logic profile

The family of description-logic profiles: a profile fixes which logical operators definitions may use; this set's profile is EL++.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Children: ⊑ EL++ LOGIC PROFILE

Since: Inception

id: `DescriptionLogicProfile` | UUID: `14eadb10-fbd0-5999-af37-05728a8503af`

⊑ Destination module for edit coordinate

The edit-coordinate dimension naming the module a moved or promoted version lands in -- the target of a transfer, as distinct from the default module new edits are stamped with and the options an editor may choose among.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `DestinationModuleForEditCoordinate` | UUID: `349cfd1d-10fd-5f8d-a0a5-d5ef0932b4da`

⊑ Development module

Predefines or standard module within a system or application that is specifically designed to support the development phase of a project

```
⊑ Module
```

Parents: ⊑∨ MODULE

Since: Inception

id: `DevelopmentModule` | UUID: `529a7069-bd33-59e6-b2ce-537fa874360a`

⊑ Development path

A path that specifies that the components are currently under development

```
⊑ Path
```

Parents: ⊑∨ PATH

Since: Inception

id: `DevelopmentPath` | UUID: `1f200ca6-960e-11e5-8994-feff819cdc9f`

⊑ Dialect

A regional or professional variety of a language, differing in which terms are preferred rather than in the language itself. Dialect semantics record whether a description is preferred or acceptable in one, and a language coordinate consults them in order.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Children: ⊑ RUSSIAN DIALECT ⊑ POLISH DIALECT ⊑ IRISH DIALECT ⊑ ENGLISH DIALECT ⊑ FRENCH DIALECT ⊑ CZECH DIALECT ⊑ KOREAN DIALECT

Since: Inception

id: `Dialect` | UUID: `b9c34574-c9ac-503b-aa24-456a0ec949a2`

⊑ Dialect for dialect/description pair

The dialect side of a dialect-and-description pair: which dialect the paired acceptability judgment applies in.

```
⊑ Description dialect pair
```

Parents: ⊑ DESCRIPTION DIALECT PAIR

Since: Inception

id: `DialectForDialectDescriptionPair` | UUID: `850bc47d-5235-5bce-99f4-c41f8a163d69`

⊑ Dialect pattern preference list for language coordinate

The preference order among dialect patterns a Language Coordinate consults when resolving a description's acceptability.

```
⊑ Language coordinate properties
```

Parents: ⊑ LANGUAGE COORDINATE PROPERTIES

Since: Inception

id: `DialectPatternPreferenceListForLanguageCoordinate` | UUID: `c060ffbf-e95f-5960-b296-8a3255c820ac`

⊑ DiGraph data type

A field that holds a directed graph whose edges are ordered pairs of vertices. Each edge can be followed from one vertex to another vertex.

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `DiGraphDataType` | UUID: `60113dfe-2bad-11eb-adc1-0242ac120002`

⊑ ∨ DiGraph default

The DiGraph field's loud default: a simple cycle — two vertices, both Uninitialized Component, with edges in both directions. The cycle is deliberate: a cycle can never be a tree, so the value proves the field truly holds a graph and not a tree.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `DiGraphDefault` | UUID: `87cb6ced-a7e4-5e96-9a20-c1ef56d68e75`

⊑ Digraph for logic coordinate

A value which describes a immutable coordinate property

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `DigraphForLogicCoordinate` | UUID: `1cdacc80-0dea-580f-a77b-8a6b273eb673`

⊑ Directed graph

A graph whose edges each run from one vertex to another, so processing follows the direction: the representational data structure navigation, lineage, and expression content is carried in. A coordinate can configure a view of a directed graph -- which graphs to include, under what STAMP, with what combining rules -- but a coordinate is configuration, never the structure itself.

```
⊑ Graph
```

Parents: ⊑ Graph

Since: Inception

id: `DirectedGraph` | UUID: `47a787a7-bdce-528d-bfcc-fde1add8d599`

⊑ Disjoint with

The atom asserting that two concepts share no instances: nothing can be both. Mirrors LogicalAxiom.Atom.DisjointWithAxiom in the sealed hierarchy.

```
⊑ Atom
```

Parents: ⊑ Atom

Since: Inception

id: `DisjointWith` | UUID: `f8433993-9a2d-5377-b564-80a45c7b7824`

⊑ Display Fields

Groups the data types a pattern's field definition can declare: what kind of value a field holds, and therefore which editor and renderer a surface uses for it.

```
⊑ Tinkar Model concept
```

Parents: ⊑ Tinkar Model concept

Children: ⊑ Component Id set data type ⊑ DiTree data type ⊑ Vertex display field ⊑ Image display field ⊑ Component data type ⊑ Concept data type ⊑ Component Id list data type ⊑ Byte array data type ⊑ Double display field ⊑ Integer data type ⊑ DiGraph data type ⊑ Float data type ⊑ Boolean data type ⊑ Logical expression display field ⊑ String data type ⊑ UUID display field ⊑ Decimal data type ⊑ Array data type ⊑ Semantic data type

Since: Inception

id: `DisplayFields` | UUID: `4e627b9c-cecb-5563-82fc-cb0ee25113b1`

⊑∨ Display Sequence

Why a roster order value is recorded: to say where this entry falls in the roster's own display order.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑ Editorial model ⊑ Purpose

Since: Inception

id: `DisplaySequence` | UUID: `f40fa97d-81b3-5eb0-9060-190579961b24`

⊑ DiTree data type

A field that holds a directed tree graph obtained from an undirected tree by replacing each undirected edge with two directed edges of opposite direction.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `DiTreeDataType` | UUID: `32f64fc6-5371-11eb-ae93-0242ac130002`

⊑∨ DiTree default

The DiTree field's loud default: a single-vertex tree whose vertex meaning is Uninitialized Component — the smallest well-formed tree, carrying the loud placeholder at its root.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `DiTreeDefault` | UUID: `48afa790-115b-5bf4-a134-a58d5615dfca`

⊑ Double

The retired dynamic-column type for double-precision values, superseded by Double display field.

```
⊑ Dynamic column data types
```

Parents: ⊑ Dynamic column data types

Since: Inception

id: `Double` | UUID: `7172e6ac-a05a-5a34-8275-aef430b18207`

⊑ Double display field

The data type for fields holding a double-precision floating-point number.

⊑ Display Fields

Parents: ⊑ Display Fields

Since: Inception

id: `DoubleDisplayField` | UUID: `85ff6e8f-9151-5428-a5f0-e07844b69260`

⊑ Dutch language

The Dutch language, as a value for a description's language field: a language coordinate that names it selects Dutch-language descriptions for rendering.

⊑ Language

Parents: ⊑ ∨ Language

Since: Inception

id: `DutchLanguage` | UUID: `21d11bd1-3dab-5034-9625-81b9ae2bd8e7`

⊑ Dynamic column data types

The retired ISAAC dynamic-semantic column type vocabulary, superseded by the Display Fields data types that field definitions actually declare. Retained for identity compatibility; its members duplicate the live vocabulary one for one.

⊑ Legacy model concepts

Parents: ⊑ Legacy model concepts

Children: ⊑ Decimal ⊑ Signed integer ⊑ UUID data type ⊑ Byte array ⊑ Double ⊑ Array ⊑ Float ⊑ Long ⊑ Boolean

Since: Inception

id: `DynamicColumnDataTypes` | UUID: `61da7e50-f606-5ba0-a0df-83fd524951e7`

⊑ Dynamic referenced component restriction

An ISAAC-era restriction on a dynamic semantic's referenced component type, retired to Legacy: patterns declare their referenced-component meaning directly.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `DynamicReferencedComponentRestriction` | UUID: `0d94ceeb-e24f-5f1a-84b2-1ac35f671db5`

⊑ Edge

A directed connection from one vertex of a graph to another. In the platform's carriers an edge is adjacency — recorded on the vertexes it connects, carrying no properties of its own; what an edge means is given by the meanings of the vertexes it joins.

```
⊑ Graph model
```

Parents: ⊑ GRAPH MODEL

Since: Inception

id: `Edge` | UUID: `9b3a5142-c1db-54a6-b130-0b7290602535`

⊑ ∨ Editorial Clarification

Why a free-text comment is captured: to explain an authoring decision or provide context a formal field cannot, for a future reviewer or maintainer.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑ EDITORIAL MODEL ⊑ PURPOSE

Since: Inception

id: `EditorialClarification` | UUID: `2cdee396-1e5a-5053-bc5b-4db5c0300930`

⊑ Editorial model

The meta-schema of editorial workflow: commentary and its subjects, review routing and reviewer assignment, and the author-roster apparatus. These concepts describe how people work on the knowledge base — as distinct from what the knowledge base asserts.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑∨ KNOWLEDGE-BASE DOCUMENTATION ⊑∨ NARRATIVE PROSE ELEMENT ⊑∨ ROSTER ORDER ⊑∨ STARTER SET AUTHOR ROSTER ⊑∨ COMMENT ⊑∨ PROSE TEXT ⊑ KOMET USER LIST ⊑∨ PREFERRED REVIEWER ASSIGNMENT ⊑∨ ROSTER MEMBERSHIP ⊑∨ ASSIGNED REVIEWER ⊑∨ REVIEW ROUTING ⊑∨ DISPLAY SEQUENCE ⊑∨ SUBJECT OF COMMENTARY ⊑∨ ROSTER ENTRY ⊑∨ EDITORIAL CLARIFICATION

Since: Inception

id: `EditorialModel` | UUID: `6dec60bc-67e5-5af7-8228-bb5161feffee`

⊑ El profile set operator

Groups the set operators the EL profile admits in definitions: the sufficient and necessary set operators.

```
⊑ Meaning
```

Parents: ⊑ MEANING

Since: Inception

id: `ElProfileSetOperator` | UUID: `2352b7a2-11fd-5a68-8ece-fcb3b36570da`

⊑ EL++ ditree

The directed graph that results from classifying a set of EL++ axioms

```
⊑ Tree
```

Parents: ⊑ TREE

Since: Inception

id: `ELDitree` | UUID: `ee04d7db-3407-568f-9b93-7b1f9f5bb0fc`

P EL++ Inferred Axioms Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ INFERRED DEFINITION	⊑∨ LOGICAL DEFINITION	—

Meaning	Purpose	Data type
Field 1		
⊑∨ EL++ Inferred terminological axioms	⊑∨ Logical Definition	⊑ DiTree data type

Since: Inception

id: `ELInferredAxiomsPattern` | UUID: `9f011812-15c9-5b1b-85f8-bb262bc1b2a2`

⊑ El++ Inferred Concept Definition

The inferred definition of a concept in the EL++ profile: the logical expression a reasoner derives from the stated definitions.

```
⊑ Logical Definition
```

Parents: ⊑∨ Logical Definition

Since: Inception

id: `ElInferredConceptDefinition` | UUID: `b2897aa0-a697-5bf2-9156-7a437c6a5057`

⊑∨ EL++ Inferred terminological axioms

The inferred side of a concept's EL++ terminological axioms: the definitional axioms a classifier concluded follow from what was stated — computed, never hand-written. The EL++ Inferred Axioms Pattern's one field carries them as a directed tree.

```
⊑ EL++ terminological axioms ⊓ Meaning
```

Parents: ⊑ EL++ terminological axioms ⊑ Meaning

Since: Inception

id: `ELInferredTerminologicalAxioms` | UUID: `b6d3be7d-1d7f-5c44-a425-5357f878c212`

⊑ EL++ logic profile

The EL++ description-logic profile: the expressivity this set's definitions stay within, chosen because classification remains tractable at terminology scale.

```
⊑ Description-logic profile
```

Parents: ⊑ Description-logic profile

Since: Inception

id: `ELLogicProfile` | UUID: `1f201e12-960e-11e5-8994-feff819cdc9f`

P EL++ Stated Axioms Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STATED DEFINITION	⊑∨ LOGICAL DEFINITION	—
e.g. ⊑ *UNINITIALIZED COMPONENT*		
Field 1		
⊑∨ EL++ STATED TERMINOLOGICAL AXIOMS	⊑∨ LOGICAL DEFINITION	⊑ DITREE DATA TYPE
e.g. 4-vertex tree		

Since: Inception

id: `ELStatedAxiomsPattern` | UUID: `e813eb92-7d07-5035-8d43-e81249f5b36e`

⊑ EL++ Stated Concept Definition

The stated definition of a concept in the EL++ profile: the logical expression an editor asserts, before classification.

```
⊑ Logical Definition
```

Parents: ⊑∨ LOGICAL DEFINITION

Since: Inception

id: `ELStatedConceptDefinition` | UUID: `0c464a4a-a0bc-53ef-9c01-ef5a049f2656`

⊑∨ EL++ Stated terminological axioms

The stated side of a concept's EL++ terminological axioms: the definitional axioms an author actually wrote — necessary and sufficient sets, role restrictions — as distinct from what a classifier later infers. The EL++ Stated Axioms Pattern's one field carries them as a directed tree.

```
⊑ EL++ terminological axioms ⊓ Meaning
```

Parents: ⊑ EL++ TERMINOLOGICAL AXIOMS ⊑ MEANING

Since: Inception

id: `ELStatedTerminologicalAxioms` | UUID: `1412bd09-eb0c-5107-9756-10c1c417d385`

⊑ EL++ terminological axioms

The set of relationships or axioms has defined by the EL++ Logic

```
⊑ Axioms
```

Parents: ⊑ AXIOMS

Children: ⊑∨ EL++ INFERRED TERMINOLOGICAL AXIOMS
⊑∨ EL++ STATED TERMINOLOGICAL AXIOMS

Since: Inception

id: `ELTerminologicalAxioms` | UUID: `b3ec50c4-e8cf-4720-b192-31374705f3b7`

⊑ English Dialect

Specifies the dialect of the English language

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Children: ⊑∨ GREAT BRITAIN ENGLISH DIALECT
⊑∨ UNITED STATES OF AMERICA ENGLISH DIALECT

Since: Inception

id: `EnglishDialect` | UUID: `c0836284-f631-3c86-8cfc-56a67814efab`

⊑ English Language

The English language, as a value for a description's language field: a language coordinate that names it selects English-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `EnglishLanguage` | UUID: `02018e5a-46ba-5297-92f1-6931b9f98a12`

⊑ Equal match relation

A value matches a member when they are equal by the member datatype's natural equality: entity references by identity, scalars by value, byte arrays by content, and Decimal by numeric equality (compareTo == 0 — an identical-representation rule would be its own relation). Float compares by IEEE semantics, so NaN matches nothing — keeping the loud-defaults coherence: an unrevised NaN default fails its constraint until revised. Graph types are outside this relation's operand conformance — graph enumerations await the isomorphic relations.

```
⊑ Member match relation
```

Parents: ⊑∨ MEMBER MATCH RELATION

Since: Inception

id: `EqualMatchRelation` | UUID: `bf47171e-a556-5b87-9e53-23252d12d3f1`

⊑ Equal to

The concrete-domain operator asserting that a value matches the stated one exactly.

```
⊑ Concrete value operator
```

Parents: ⊑∨ CONCRETE VALUE OPERATOR

Since: Inception

id: `EqualTo` | UUID: `5c9b5844-1434-5111-83d5-cb7cb0be12d9`

⊑ Exact

Source and target are semantic or exact lexical match

```
⊑ Grouping
```

Parents: ⊑ GROUPING

Since: Inception

id: `Exact` | UUID: `8aa6421d-4966-5230-ae5f-aca96ee9c2c1`

⊑∨ Example UCUM Units

The Unified Code for Units of Measure (UCUM) is a code system intended to include all units of measures being contemporarily used in international science, engineering, and business. (www.unitsofmeasure.org) This field contains example units of measures for this term expressed as UCUM units.

```
⊑ Phenomenon ⊓ Meaning
```

Parents: ⊑ PHENOMENON ⊑ MEANING

Since: Inception

id: `ExampleUCUMUnits` | UUID: `80cd4978-314d-46e3-bc13-9980280ae955`

⊑ Existential restriction

Existential restrictions describe objects that participate in at least one relationship along a specified property to objects of a specified class.

```
⊑ Role operator
```

Parents: ⊑ ROLE OPERATOR

Since: Inception

id: `ExistentialRestriction` | UUID: `91e9080f-78f6-5d23-891d-f5b6e77995c8`

⊑∨ Express axiom syntax

Why an axiom-syntax semantic exists: to carry a component's definitional axioms as serialized syntax text, for exchange with systems that read that syntax.

```
⊑ Axiom Syntax ⊓ Purpose
```

Parents: ⊑∨ AXIOM SYNTAX ⊑ PURPOSE

Since: Inception

id: `ExpressAxiomSyntax` | UUID: `db55557c-e9ee-4504-aae3-df695b6d6c57`

⊑ Extended description type

Used to store non-snomed description types when other terminologies are imported

```
⊑ Description type
```

Parents: ⊑∨ DESCRIPTION TYPE

Since: Inception

id: `ExtendedDescriptionType` | UUID: `5a2e7786-3e41-11dc-8314-0800200c9a66`

⊑ Extended relationship type

Used to store non-snomed relationship types when other terminologies are imported- especially when a relationship is mapped onto a snomed relationship type (such as isa)

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `ExtendedRelationshipType` | UUID: `d41d928f-8a97-55c1-aa6c-a289b413fbfd`

⊑∨ External Identity Mapping

Why an Identifier Pattern semantic exists: to record an alternate identifier a component carries in an external terminology's own identifier scheme.

```
⊑ Identifier model ⊓ Purpose
```

Parents: ⊑ IDENTIFIER MODEL ⊑ PURPOSE

Since: Inception

id: `ExternalIdentityMapping` | UUID: `7aba1c11-4a23-57a1-9512-f1df89165789`

⊑ Feature

The typed atom asserting a concrete-domain comparison: a feature type, a concrete value operator, and a literal, such as a strength of exactly 500 milligrams. Mirrors LogicalAxiom.Atom.TypedAtom.Feature in the sealed hierarchy.

```
⊑ Typed atom
```

Parents: ⊑ TYPED ATOM

Since: Inception

id: `Feature` | UUID: `5e76a88e-794a-5fdd-8eb2-4a9e4b1386b6`

⊑ Feature Role Type

The role type marking a role as feature-valued: the bridge between role machinery and concrete-domain features.

```
⊑ Role type
```

Parents: ⊑ ROLE TYPE

Since: Inception

id: `FeatureRoleType` | UUID: `acb8d47e-adac-491d-bc60-78e94cacd312`

⊑ Feature Type

The type vocabulary for feature atoms: names which measurable characteristic a feature axiom compares, the concrete-domain counterpart of a role type.

```
⊑ Logical expression model
```

Parents: ⊑ LOGICAL EXPRESSION MODEL

Since: Inception

id: `FeatureType` | UUID: `c9120d8b-1acc-5267-9f33-fa716abdb69d`

⊑ Field categories

Groups the field-level meta-schema concepts: the terminology describing what kind of value — a component, a field definition, a semantic field, or a set of these — a semantic's field can hold.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Children: ⊑∨ SEMANTIC FIELD FIELD ⊑∨ SEMANTIC FIELD FIELDS SET ⊑ FIELD VALUE FIELD ⊑∨ COMPONENT VERSIONS FIELD ⊑∨ COMPONENT FIELD ⊑∨ FIELD DEFINITIONS SET ⊑∨ FIELD DEFINITION FIELD ⊑∨ COMPONENT VERSIONS SET ⊑ FIELD DEFINITION DATA TYPE FIELD

Since: Inception

id: `FieldCategories` | UUID: `ed230c7c-20f9-470d-8566-5057f92748a5`

⊑ Field definition data type field

The field kind naming a field definition's data type slot: which data type a pattern declares for one of its fields.

```
⊑ Field categories
```

Parents: ⊑ FIELD CATEGORIES

Children: ⊑ STRING FIELD ⊑ INSTANT FIELD ⊑ INTEGER FIELD ⊑ BOOLEAN FIELD

Since: Inception

id: `FieldDefinitionDataTypeField` | UUID: `02273b53-fce7-4cbe-921d-2cff67e81ad5`

⊑∨ Field definition field

The field kind naming a pattern version's field-definition slot: each value is one field definition — the meaning, purpose, and data type the pattern declares for one of its fields.

```
⊑ Field categories ⊓ Meaning
```

Parents: ⊑ FIELD CATEGORIES ⊑ MEANING

Since: Inception

id: `FieldDefinitionField` | UUID: `14171f07-e74f-409a-b555-06b478818f76`

⊑ Field definition meaning field

The field kind naming a field definition's meaning slot: the concept saying what the declared field is.

```
⊑ Concept field
```

Parents: ⊑∨ CONCEPT FIELD

Since: Inception

id: `FieldDefinitionMeaningField` | UUID: `74dffbed-0bef-44a4-8ad6-8cff84fe47ae`

⊑ Field definition purpose field

The field kind naming a field definition's purpose slot: the concept saying why the declared field is recorded.

```
⊑ Concept field
```

Parents: ⊑∨ CONCEPT FIELD

Since: Inception

id: `FieldDefinitionPurposeField` | UUID: `93239959-50e6-4645-b5fc-6d47da92e666`

⊑∨ Field definitions set

Why a pattern version's field-definition field is recorded: to hold the pattern's complete, ordered field definitions — the declared shape every semantic of the pattern must fit.

```
⊑ Field categories ⊓ Purpose
```

Parents: ⊑ FIELD CATEGORIES ⊑ PURPOSE

Since: Inception

id: `FieldDefinitionsSet` | UUID: `975de83e-ab99-4a9e-9051-4cbf310a2123`

⊑ Field substitution

Replacing a placeholder variable in a field with a specific value

```
⊑ Meaning
```

Parents: ⊑ MEANING

Children: ⊑ INSTANT SUBSTITUTION ⊑ FLOAT SUBSTITUTION ⊑ CONCEPT SUBSTITUTION ⊑ BOOLEAN SUBSTITUTION

Since: Inception

id: `FieldSubstitution` | UUID: `8fdce1aa-ca82-5abc-8cfa-230c14688abc`

⊑ Field value field

The field kind for one field value a semantic version actually carries, as distinct from the field definition its pattern declares.

```
⊑ Field categories
```

Parents: ⊑ FIELD CATEGORIES

Since: Inception

id: `FieldValueField` | UUID: `7e4a96fc-0522-4d74-a7d1-ca74be3bc236`

⊑∨ Field Value Restriction

Why a field constraint semantic exists: to restrict which values are legal for one of the constrained pattern's own fields.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ CONSTRAINT MODEL ⊑ PURPOSE

Since: Inception

id: `FieldValueRestriction` | UUID: `c7e33912-1f1c-5821-9c88-4b0a4504c028`

⊑ Float

The retired dynamic-column type for floating-point values, superseded by Float data type.

```
⊑ Dynamic column data types
```

Parents: ⊑ DYNAMIC COLUMN DATA TYPES

Since: Inception

id: `Float` | UUID: `fb591801-7b37-525d-980d-98a1c63ceee0`

⊑ Float data type

Represents values as high-precision fractional values.

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `FloatDataType` | UUID: `6efe7087-3e3c-5b45-8109-90d7652b1506`

⊑∨ Float default

The Float field's loud default: NaN, the type's native non-value — chosen over a stretched-sevens sentinel because a true non-value is louder still: it cannot be mistaken for data at all, and it propagates through arithmetic, so any computation over an unrevised default stays loudly not-a-number.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `FloatDefault` | UUID: `8ce26a3b-46fc-563b-853d-ccb5ee9183e7`

⊑ Float literal

Numbers with decimal point or an exponential part

```
⊑ Literal value
```

Parents: ⊑ LITERAL VALUE

Since: Inception

id: `FloatLiteral` | UUID: `da754dd9-9961-5819-91f5-8245d49850b4`

⊑ Float substitution

A field substitution whose substituted value is a float; part of the retired ISAAC field-substitution family.

```
⊑ Field substitution
```

Parents: ⊑ FIELD SUBSTITUTION

Since: Inception

id: `FloatSubstitution` | UUID: `cf18fe25-bd21-586e-9da4-da6cb335fd12`

⊑ French dialect

The French dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in French usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Since: Inception

id: `FrenchDialect` | UUID: `75d00a0d-8e46-5e42-ad34-3e46269b28a3`

⊑ French Language

The French language, as a value for a description's language field: a language coordinate that names it selects French-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `FrenchLanguage` | UUID: `8b23e636-a0bd-30fb-b8e2-1f77eaa3a87e`

⊑ Fully qualified name description type

Fully qualified name is a description that uniquely identifies and differentiates it from other concepts with similar descriptions

```
⊑ Description type
```

Parents: ⊑∨ DESCRIPTION TYPE

Since: Inception

id: `FullyQualifiedNameDescriptionType` | UUID: `00791270-77c9-32b6-b34f-d932569bd2bf`

P GB Dialect Pattern

Particular form of language specific form of English language, particular to Great Britain.

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ DESCRIPTION ACCEPTABILITY	⊑∨ DESCRIPTION SEMANTIC	—
Field 1		
⊑∨ GREAT BRITAIN ENGLISH DIALECT	⊑∨ DESCRIPTION ACCEPTABILITY	⊑ COMPONENT DATA TYPE

Since: Inception

id: `GBDialectPattern` | UUID: `561f817a-130e-5e56-984d-910e9991558c`

⊑ German Language

The German language, as a value for a description's language field: a language coordinate that names it selects German-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `GermanLanguage` | UUID: `5f144b18-76a8-5c7e-8480-55a5030d707f`

⊑ Graph

A set of vertexes and the edges that connect them — the representation logical expressions, navigation structures, and lineage records are all carried in. A graph as carried here is directed: each edge runs from one vertex to another, and processing follows that direction.

```
⊑ Graph model
```

Parents: ⊑ Graph model

Children: ⊑ Directed graph ⊑ Tree

Since: Inception

id: `Graph` | UUID: `da454dbd-ed6e-55cf-af5a-0d51b40d7640`

⊑ Graph model

The meta-schema of graph representation and processing: vertexes that carry a meaning and properties, edges that connect them, graphs assembled from both, and trees as the single-parent special case. Logical expressions, navigation structures, and version-control lineage are all carried as graphs; this subsystem names the representation those carriers share.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑ Vertex meaning ⊑ Graph ⊑ Vertex ⊑ Edge ⊑ Vertex property

Since: Inception

id: `GraphModel` | UUID: `c99b70a6-e563-5c8a-b3d9-5f599d0ee9bd`

⊑∨ Great Britain English dialect

Great Britain: English Language reference set

```
⊑ English Dialect ⊓ Meaning
```

Parents: ⊑ English Dialect ⊑ Meaning

Since: Inception

id: `GreatBritainEnglishDialect` | UUID: `eb9a5e42-3cba-356d-b623-3ed472e20b30`

⊑ Greater than

The concrete-domain operator asserting that a value is strictly above the stated bound.

```
⊑ Concrete value operator
```

Parents: ⊑∨ Concrete value operator

Since: Inception

id: `GreaterThan` | UUID: `65af466b-360c-58b2-8b7d-2854150029a8`

⊑ Greater than or equal to

The concrete-domain operator asserting that a value is at least the stated bound.

```
⊑ Concrete value operator
```

Parents: ⊑∨ CONCRETE VALUE OPERATOR

Since: Inception

id: `GreaterThanOrEqualTo` | UUID: `c1baba19-e918-5d2c-8fa4-b0ad93e03186`

⊑ Gretel

The default editing identity a fresh Komet installation offers, so a user can author before establishing their own author concept.

```
⊑ Author
```

Parents: ⊑∨ AUTHOR

Since: Inception

id: `Gretel` | UUID: `1c0023ed-559e-3311-9e55-bd4bd9e5628f`

⊑ Grouping

Groups the correlation match markers: whether an expression correlation is exact or partial.

```
⊑ Tinkar Model concept
```

Parents: ⊑ TINKAR MODEL CONCEPT

Children: ⊑ PARTIAL ⊑ EXACT

Since: Inception

id: `Grouping` | UUID: `8d76ead7-6c75-5d25-84d4-ca76d928f8a6`

⊑ Has Active Ingredient

A domain role type from the medication model: relates a product to an ingredient active in it. Held as a SNOMED-heritage example; migrates to a domain starter set when one exists.

```
⊑ Role type
```

Parents: ⊑ ROLE TYPE

Since: Inception

id: `HasActiveIngredient` | UUID: `65bf3b7f-c854-36b5-81c3-4915461020a8`

⊑ Has Dose Form

A domain role type from the medication model: relates a product to its dose form. Held as a SNOMED-heritage example; migrates to a domain starter set when one exists.

```
⊑ Role type
```

Parents: ⊑ ROLE TYPE

Since: Inception

id: `HasDoseForm` | UUID: `072e7737-e22e-36b5-89d2-4815f0529c63`

⊑ Health concept

The domain anchor for health-related content under Phenomenon. Held as a SNOMED-heritage example; domain content grows in domain starter sets.

```
⊑ Phenomenon
```

Parents: ⊑ PHENOMENON

Since: Inception

id: `HealthConcept` | UUID: `a892950a-0847-300c-b477-4e3cbb945225`

⊑ ∨ Identified Component

What an Identifier Pattern semantic's referenced component is: the component an external identifier names — the carrier of an alternate identity in another identifier scheme, distinct from Identifier source, which names the field value saying where that identifier came from (IKE-Network/ike-issues#891).

```
⊑ Identifier model ⊓ Meaning
```

Parents: ⊑ IDENTIFIER MODEL ⊑ MEANING

Since: Inception

id: `IdentifiedComponent` | UUID: `ab2c34b1-7194-5245-ae7a-fde89830442b`

⊑∨ Identifier Authority

Why an identifier source value is recorded: to say which identifier-issuing authority the identifier came from.

```
⊑ Identifier model ⊓ Purpose
```

Parents: ⊑ Identifier model ⊑ Purpose

Since: Inception

id: `IdentifierAuthority` | UUID: `120121c5-5b15-5ffb-9e8a-6cf39d2a2acd`

⊑ Identifier model

The meta-schema of identity interchange: how a component carries identifiers minted by external authorities, which authority stands behind each identifier, and how an external identity maps onto a knowledge-base component. Distinct from public identity itself (a chronicle-model concern): this subsystem is about the identifiers other systems know a component by.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Identifier Authority ⊑∨ Identifier Text
⊑∨ External Identity Mapping ⊑ NID ⊑∨ Identified Component

Since: Inception

id: `IdentifierModel` | UUID: `ca825cab-4ea6-5ef2-824c-3ca40a98bf72`

P Identifier Pattern

An identifier pattern is used to identity a concept which contains the identifier source and the actual value.

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Identified Component	⊑∨ External Identity Mapping	—
e.g. ⊑ Uninitialized Component		
Field 1		

Meaning	Purpose	Data type
⊑∨ Identifier Source	⊑∨ Identifier Authority	⊑ Component data type
e.g. ⊑ *Universally unique identifier*		
Field 2		
⊑∨ Identifier Value	⊑∨ Identifier Text	⊑ String data type
e.g. 55f74246-0a25-57ac-9473-a788d08fb656		

Since: Inception

id: `IdentifierPattern` | UUID: `5d60e14b-c410-5172-9559-3c4253278ae2`

⊑∨ Identifier Source

An identifier used to label the identity of a unique component.

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ Tinkar Model concept ⊑ Meaning

Children: ⊑ Universally unique identifier

Since: Inception

id: `IdentifierSource` | UUID: `5a87935c-d654-548f-82a2-0c06e3801162`

⊑∨ Identifier Text

Why an identifier value is recorded: to hold the identifier's own literal text, once a source has been chosen.

```
⊑ Identifier model ⊓ Purpose
```

Parents: ⊑ Identifier model ⊑ Purpose

Since: Inception

id: `IdentifierText` | UUID: `a3f37520-ca20-5bf0-bb48-ce4892160233`

⊑∨ Identifier Value

The literal text of an identifier, once an issuing source has been named -- the value side of an identifier semantic.

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `IdentifierValue` | UUID: `b32dd26b-c3fc-487e-987e-16ace71a0d0f`

⊑ IKE Community

Community authorship: the IKE Network itself, attributed as author for content synthesized by tooling on the Network's behalf (e.g. the identity-exact starter-set ingest, IKE-Network/ike-issues#872) rather than by an individual editor or an ingested upstream source.

```
⊑ Author
```

Parents: ⊑ ∨ AUTHOR

Since: Inception

id: `IKECommunity` | UUID: `a03bedb4-f505-5153-a213-7581b651c929`

⊑ IkeFoundation module

The module scoping every stamp of the IkeFoundation content set; the export dimension for this knowledge.

```
⊑ Module
```

Parents: ⊑ ∨ MODULE

Since: Inception

id: `IkeFoundationModule` | UUID: `61615e8f-173a-5875-937e-51f1fa7bbadb`

⊑ IkeFoundation root

Root concept of the Ike starter set.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Since: Inception

id: `IkeFoundationRoot` | UUID: `391b61a5-f53c-513b-acfd-bfa0100db7d8`

⊑ Image display field

The data type for fields holding an image, encoded as the bytes of a renderable image format.

⊑ `Display Fields`

Parents: ⊑ Display Fields

Since: Inception

id: `ImageDisplayField` | UUID: `cd9ea037-0af9-586b-9369-7bc044cdb8f7`

⊑ Immediate child field constraint

Legal values are the anchor concept's direct children only, not deeper descendants.

⊑ `Taxonomy field constraint kind`

Parents: ⊑ Taxonomy field constraint kind

Since: Inception

id: `ImmediateChildFieldConstraint` | UUID: `97f24f21-0210-5a00-98cd-2efb76d1f4cc`

⊑ ImmutableCoordinate Properties

Groups the dimension concepts every coordinate kind is assembled from -- stamp, language, logic, path, edit, and navigation -- named individually so a view's configuration is itself describable knowledge. Coordinates are immutable: a changed dimension yields a new coordinate, never a mutated one.

⊑ `View coordinate model`

Parents: ⊑ View coordinate model

Children: ⊑ Module options for edit coordinate ⊑ Root for logic coordinate
⊑ Stated pattern for logic coordinate ⊑ Digraph for logic coordinate
⊑ Description logic profile for logic coordinate ⊑ Author for edit coordinate
⊑ Vertex sort ⊑ Path for path coordinate
⊑ Language nid for language coordinate ⊑ Inferred pattern for logic coordinate
⊑ ∨ Concept pattern for logic coordinate ⊑ Destination module for edit coordinate
⊑ Module preference list for stamp coordinate ⊑ Vertex state set
⊑ Modules for stamp coordinate ⊑ Path options for edit coordinate
⊑ Module exclusion set for stamp coordinate ⊑ Author for stamp coordinate
⊑ Description type preference list for language coordinate
⊑ Language specification for language coordinate

⊑ Allowed states for stamp coordinate
⊑ Module preference list for language coordinate
⊑ Module preference order for stamp coordinate ⊑ Path origins for stamp path
⊑ Classifier for logic coordinate ⊑ Default module for edit coordinate
⊑ Position on path

Since: Inception

id: `ImmutableCoordinateProperties` | UUID: `ab41a788-8a83-5452-8dc0-2d8375e0bfe6`

⊑ Implication set

A baseline grouping for implication axioms, retained for identity compatibility: property-sequence implications are the atoms it grouped.

```
⊑ Logical set
```

Parents: ⊑ Logical set

Since: Inception

id: `ImplicationSet` | UUID: `ee467a5b-9292-4e0a-a165-3b1a359a8c98`

⊑ Inactive state

Concept used to represent a status for components that are no longer active

```
⊑ Status value
```

Parents: ⊑ ∨ Status value

Since: Inception

id: `InactiveState` | UUID: `03004053-c23e-5206-8514-fb551dd328f4`

⊑ Include Lower Bound

The field saying whether an interval includes its lower bound: closed when included, open when not.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `IncludeLowerBound` | UUID: `2300a210-d722-48af-8c36-118a3f980312`

⊑ Include Upper Bound

The field saying whether an interval includes its upper bound: closed when included, open when not.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `IncludeUpperBound` | UUID: `990b7e1d-3dcc-4c6e-a068-e30400607d50`

⊑ Inclusion set

A set of relationships that indicate something is has an inclusion. Not necessarily or sufficient but inclusive.

```
⊑ Logical set
```

Parents: ⊑ Logical set

Since: Inception

id: `InclusionSet` | UUID: `def77c09-e1eb-40f2-931a-e7cf2ce0e597`

⊑ ∨ Inferred Definition

The relationships/axioms of a concept that have been inferred

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ Tinkar Model concept ⊑ Meaning

Since: Inception

id: `InferredDefinition` | UUID: `b1abf4dc-9838-4b46-ac55-10c4f92ba10b`

⊑ Inferred navigation

Navigation computed by the reasoner from inferred definitions: the classified parent and child adjacency navigators render.

```
⊑ Navigation
```

Parents: ⊑ Navigation

Since: Inception

id: `InferredNavigation` | UUID: `4bc6c333-7fc9-52f1-942d-f8decba19dc2`

P Inferred Navigation Pattern

A pattern specifying the origins and destinations for concepts based on the inferred terminological axioms.

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Is-a	⊑∨ Taxonomy Navigation Cache	—
Field 1		
⊑∨ Relationship destination	⊑∨ Is-a	⊑ Component Id set data type
Field 2		
⊑∨ Relationship origin	⊑∨ Is-a	⊑ Component Id set data type

Since: Inception

id: `InferredNavigationPattern` | UUID: `a53cc42d-c07e-5934-96b3-2ede3264474e`

⊑ Inferred pattern for logic coordinate

The pattern the classifier's derived axioms are written to, for a Logic Coordinate.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `InferredPatternForLogicCoordinate` | UUID: `9ecf4d76-4346-5e5d-8316-bdff48a5c154`

⊑ Inferred premise type

The axiom view following the application of the reasoner

```
⊑ Axiom origin
```

Parents: ⊑ Axiom origin

Since: Inception

id: `InferredPremiseType` | UUID: `a4c6bf72-8fb6-11db-b606-0800200c9a66`

⊑∨ Instant default

The Instant field's loud default: the pre-inception instant (historically 'premundane'), the time type's native non-value — the platform's own before-all-recorded-time marker, chosen over a sentinel date because a true non-value cannot be mistaken for a real timestamp.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `InstantDefault` | UUID: `4f18ebf5-f890-571c-a2e1-1432efc95988`

⊑ Instant field

The field kind for instant-valued fields: a pattern field holding a point in time.

```
⊑ Field definition data type field
```

Parents: ⊑ Field definition data type field

Children: ⊑∨ Time field

Since: Inception

id: `InstantField` | UUID: `e9bde1bc-aa72-430a-afe1-aa8aec8833b4`

⊑ Instant literal

May refer to a specific point in time which is often represented by a date or time value

```
⊑ Literal value
```

Parents: ⊑ Literal value

Since: Inception

id: `InstantLiteral` | UUID: `1fbf42e2-42b7-591f-b7fd-ba5de659529e`

⊑ Instant substitution

A field substitution whose substituted value is an instant; part of the retired ISAAC field-substitution family.

```
⊑ Field substitution
```

Parents: ⊑  Field substitution

Since: Inception

id: `InstantSubstitution` | UUID: `56599345-31c5-5817-9d36-57dd3a626b3a`

⊑  Integer data type

Data type that represents some range of mathematical integers

```
⊑ Display Fields
```

Parents: ⊑  Display Fields

Since: Inception

id: `IntegerDataType` | UUID: `ff59c300-9c4e-5e77-a35d-6a133eb3440f`

⊑ ∨  Integer default

The Integer field's loud default: the stretched-sevens sentinel 777777777 (nine sevens) — a repdigit signature that reads as deliberately planted and cannot plausibly occur as natural data, where Integer.MIN_VALUE would read as an overflow bug. Never Java's silent zero.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑  Defaults and templates model ⊑  Meaning

Since: Inception

id: `IntegerDefault` | UUID: `11f902b5-7c46-5157-9cfd-289d3019a559`

⊑  Integer field

The field kind for integer-valued fields: a pattern field holding a whole number.

```
⊑ Field definition data type field
```

Parents: ⊑  Field definition data type field

Since: Inception

id: `IntegerField` | UUID: `db249d1f-ea2e-4608-ae13-166ed20ca825`

⊤  Integrated Knowledge Management

Terminologies that are represented in a harmonized manner

Children: ⊑  Model concept ⊑  Phenomenon

Since: Inception

id: `IntegratedKnowledgeManagement` | UUID: `7c21b6c5-cf11-5af9-893b-743f004c97f5`

⊑ Interval Lower Bound

The field holding an interval's lower bound value.

⊑ `Interval Set Axioms`

Parents: ⊑ INTERVAL SET AXIOMS

Since: Inception

id: `IntervalLowerBound` | UUID: `52b3e38a-fccb-4779-aa61-4e87abd56419`

⊑ Interval property set

The logical set grouping interval-property axioms under a definition root. Mirrors LogicalAxiom.LogicalSet.IntervalPropertySet in the sealed hierarchy.

⊑ `Logical set`

Parents: ⊑ LOGICAL SET

Since: Inception

id: `IntervalPropertySet` | UUID: `9afc988a-3724-4754-8b74-651426472b19`

⊑ Interval role

The typed atom asserting a role bounded to an interval: a role whose filler is constrained by interval bounds. Mirrors LogicalAxiom.Atom.TypedAtom.IntervalRole in the sealed hierarchy.

⊑ `Typed atom`

Parents: ⊑ TYPED ATOM

Since: Inception

id: `IntervalRole` | UUID: `ed9d3506-65ad-48ea-bd01-95474fecdbc4`

⊑ Interval Role Type

The role-type vocabulary for interval roles: names which relation an interval role asserts.

⊑ `Role type`

Parents: ⊑ Role type

Since: Inception

id: `IntervalRoleType` | UUID: `6fa58611-af37-402e-a0c2-6ee1d6068651`

⊑ Interval Set Axioms

Groups the interval-axiom vocabulary: the bound values, openness markers, and unit of measure an interval-valued assertion carries.

```
⊑ Axioms
```

Parents: ⊑ Axioms

Children: ⊑ Unit of Measure ⊑ Lower Bound Open ⊑ Interval Type ⊑ Upper Bound Open ⊑ Include Upper Bound ⊑ Interval Upper Bound ⊑ Interval Lower Bound ⊑ Include Lower Bound

Since: Inception

id: `IntervalSetAxioms` | UUID: `b253e725-d7cd-46e3-bc3a-5db8b3ffbd52`

⊑ Interval Type

The concept naming which kind of interval an interval assertion carries.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `IntervalType` | UUID: `ba3191ee-a260-41a6-99fd-74a22fdc937e`

⊑ Interval Upper Bound

The field holding an interval's upper bound value.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `IntervalUpperBound` | UUID: `6565f774-ff6c-4882-832f-31ddc462adf7`

⊑ Intrinsic role

An ISAAC-era marker for roles intrinsic to a concept, retired to Legacy.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `IntrinsicRole` | UUID: `a2d37d2d-ac49-589f-ba36-ac9b8808b00b`

⊑ Inverse name

This is the extended description type that maye be attached to a description within a concept that defines as Association refex to signify that the referenced description is the inverse of the association name

```
⊑ Description type
```

Parents: ⊑ ∨ DESCRIPTION TYPE

Since: Inception

id: `InverseName` | UUID: `c342d18a-ec1c-5583-bfe3-59e6324ae189`

⊑ Inverse tree list

The legacy tree-amalgam field holding a navigation tree's child-to-parent adjacency, paired with Tree list.

```
⊑ Tree amalgam properties
```

Parents: ⊑ TREE AMALGAM PROPERTIES

Since: Inception

id: `InverseTreeList` | UUID: `45167fb6-e01d-53a8-8be3-768aae18729d`

⊑ Irish dialect

The Irish dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in Irish usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Since: Inception

id: `IrishDialect` | UUID: `c0f77638-6274-5b40-b832-ac1cba7ec515`

⊑ Irish language

The Irish language, as a value for a description's language field: a language coordinate that names it selects Irish-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `IrishLanguage` | UUID: `58e82fc4-1492-5cf8-8997-43800360bbd6`

⊑∨ Is-a

The subtype relation: everything the child names is also what the parent names. The relation taxonomy navigation is computed from, and the only axiom shape the ingested foundation uses.

```
⊑ Tinkar Model concept ⊓ Meaning ⊓ Purpose
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING ⊑ PURPOSE

Since: Inception

id: `IsA` | UUID: `c93a30b9-ba77-3adb-a9b8-4589c9f8fb25`

⊑ Is-a inferred navigation

Designates the parent child relationship following the application of the reasoner

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `IsAInferredNavigation` | UUID: `b620768f-1479-5afa-a027-5a9ae6caf0d5`

⊑ Is-a stated navigation

Designates the parent child relationship as authored

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `IsAStatedNavigation` | UUID: `d555dde9-c97e-5480-819a-7472eda3dbfa`

⊑ Italian Language

The Italian language, as a value for a description's language field: a language coordinate that names it selects Italian-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `ItalianLanguage` | UUID: `bdd59458-381a-5818-8577-60525f11ac6c`

⊑ Kind-of field constraint

Legal values are the anchor concept itself plus every one of its descendants (self included) — Tinkar's own "kind of" relation.

```
⊑ Taxonomy field constraint kind
```

Parents: ⊑ TAXONOMY FIELD CONSTRAINT KIND

Since: Inception

id: `KindOfFieldConstraint` | UUID: `127975b1-8ca7-50f0-aea5-bd69c47682fd`

⊑ ∨ Knowledge-Base Documentation

Why a prose element is captured as knowledge-base content: to carry the guide's curated long-form manuscript as replayable KB data with resolvable k: tokens, rather than prose living only in an external file.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑ EDITORIAL MODEL ⊑ PURPOSE

Since: Inception

id: `KnowledgeBaseDocumentation` | UUID: `0812cf20-fb53-57a6-bd8d-277968cc35f8`

P Komet base model component pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Starter Data Authoring	⊑∨ Set membership	—
e.g. P Komet base model component pattern		

Since: Inception

id: `KometBaseModelComponentPattern` | UUID: `bbbbf1fe-00f0-55e0-a19c-6300dbaab9b2`

⊑ Komet issue

Komet being the 'annotation type' - specified type

```
⊑ Legacy model concepts
```

Parents: ⊑ Legacy model concepts

Since: Inception

id: `KometIssue` | UUID: `e1dd7bf2-224d-53a5-a5fb-7b25b05d17a6`

⊑ KOMET module

The module for content the Komet application itself defines, such as its window and layout terminology.

```
⊑ Module
```

Parents: ⊑∨ Module

Since: Inception

id: `KOMETModule` | UUID: `34a6dae3-e5e9-50db-a9ee-69c1067911d8`

⊑ KOMET user

Authorized to author, edit and/or view in Komet

```
⊑ Author
```

Parents: ⊑∨ AUTHOR

Since: Inception

id: `KOMETUser` | UUID: `61c1a544-2acf-58cd-8cc0-9ac581d4227e`

⊑ KOMET user list

The roster of author identities a Komet installation offers at sign-in.

```
⊑ Editorial model
```

Parents: ⊑ EDITORIAL MODEL

Since: Inception

id: `KOMETUserList` | UUID: `5e77558d-97d0-52b6-adf0-d54beb97b3a6`

⊑ Korean dialect

The Korean dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in Korean usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Children: ⊑ STANDARD KOREAN DIALECT

Since: Inception

id: `KoreanDialect` | UUID: `6fb2eb9c-fb9e-5959-802c-fb17bcba3079`

⊑ Korean Language

The Korean language, as a value for a description's language field: a language coordinate that names it selects Korean-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `KoreanLanguage` | UUID: `1464f995-d658-5e9d-86e0-8308a6fa57eb`

⊑∨ Language

Specifies the language of the description text.

```
⊑ Tinkar Model concept ⊓ Purpose
```

Parents: ⊑ Tinkar Model concept ⊑ Purpose

Children: ⊑ English Language ⊑ Russian language ⊑ Polish Language
⊑ Czech language ⊑ Chinese language ⊑ Lithuanian language
⊑ Italian Language ⊑ Swedish language ⊑ German Language
⊑ Danish language ⊑ French Language ⊑ Spanish language ⊑ Dutch language
⊑ Korean Language ⊑ Irish language

Since: Inception

id: `Language` | UUID: `f56fa231-10f9-5e7f-a86d-a1d61b5b56e3`

⊑ Language coordinate name

The field naming a stored language coordinate, so a surface can present saved coordinates by name.

```
⊑ Language coordinate properties
```

Parents: ⊑ Language coordinate properties

Since: Inception

id: `LanguageCoordinateName` | UUID: `42dff20f-5ed2-559a-91ad-91d44a573c63`

⊑ Language coordinate properties

Groups the language-coordinate dimensions: which language, which dialect order, which description types in which preference order, and which modules a rendering surface consults when choosing text to show.

```
⊑ View coordinate model
```

Parents: ⊑ View coordinate model

Children: ⊑ Dialect pattern preference list for language coordinate
⊑ Language coordinate name

Since: Inception

id: `LanguageCoordinateProperties` | UUID: `ea1a52f7-0305-5487-8766-e846330f167a`

⊑ ∨ Language for description

Captures the language code for a description

```
⊑ Description version properties ⊓ Meaning
```

Parents: ⊑ DESCRIPTION VERSION PROPERTIES ⊑ MEANING

Since: Inception

id: `LanguageForDescription` | UUID: `cd56cceb-8507-5ae5-a928-16079fe6f832`

⊑ Language nid for language coordinate

The language-coordinate dimension naming the language whose descriptions the coordinate selects.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `LanguageNidForLanguageCoordinate` | UUID: `38e0c7b8-1e33-56a2-9eb2-ee20c4960684`

⊑ Language specification for language coordinate

The language-coordinate dimension carrying the coordinate's full language specification as one value: language plus dialect and description-type preferences.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `LanguageSpecificationForLanguageCoordinate` | UUID: `b0ad4d77-e1bc-5fd1-922e-5fad675e9bfd`

⊑ Laterality

A domain role type from the anatomy model: distinguishes sided body structures, left from right. Held as a SNOMED-heritage example; migrates to a domain starter set when one exists.

```
⊑ Role type
```

Parents: ⊑ ROLE TYPE

Since: Inception

id: `Laterality` | UUID: `26ca4590-bbe5-327c-a40a-ba56dc86996b`

⊑ Leaf descendant field constraint

Legal values are the anchor concept's leaf descendants only — descendants with no children of their own — excluding grouping concepts that exist only to organize other concepts.

```
⊑ Taxonomy field constraint kind
```

Parents: ⊑ TAXONOMY FIELD CONSTRAINT KIND

Since: Inception

id: `LeafDescendantFieldConstraint` | UUID: `88c5f3a3-2932-5e45-ab7e-3b0bdd9a5233`

⊑ Legacy model concepts

A visible deprecation signal for model machinery, not a functional restriction: content reparented here is dormant or superseded and a candidate for removal in a near-future revision. Being here says nothing about whether the content is still resolvable or referenced elsewhere — it may well be — only that it is no longer this set's preferred terminology going forward. The scope is deliberate: machinery deprecates in place, where its history stays browsable; domain content that does not belong to this set migrates to its home starter set instead, and is never marked legacy here.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑ TREE AMALGAM PROPERTIES ⊑ SANDBOX COMPONENT
⊑ REFERENCED COMPONENT FOR SEMANTIC ⊑ COMPONENT FOR SEMANTIC
⊑ ANNOTATION TYPE ⊑ PROPERTY PATTERN IMPLICATION ⊑ EXTENDED RELATIONSHIP TYPE
⊑ IS-A STATED NAVIGATION ⊑ ACTION PROPERTIES ⊑ UNMODELED ROLE CONCEPT
⊑ CORRELATION PROPERTIES ⊑ DYNAMIC REFERENCED COMPONENT RESTRICTION
⊑ KOMET ISSUE ⊑ OBJECT ⊑ DYNAMIC COLUMN DATA TYPES ⊑ OBJECT PROPERTIES
⊑ IS-A INFERRED NAVIGATION ⊑ INTRINSIC ROLE

Since: Inception

id: `LegacyModelConcepts` | UUID: `7fd61405-6776-52e6-9e4e-3d21dfdf28ad`

⊑ V Legal Value Source

Why the Value-set pattern value is recorded: to name the pattern whose active semantics enumerate the legal values for a value-set constraint.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `LegalValueSource` | UUID: `97153d1a-91d8-5a53-8cb4-69c003d41526`

⊑ Less than

The concrete-domain operator asserting that a value is strictly below the stated bound.

```
⊑ Concrete value operator
```

Parents: ⊑∨ Concrete value operator

Since: Inception

id: `LessThan` | UUID: `6f96e8cf-5568-5e49-8a90-aa6c65125ee9`

⊑ Less than or equal to

The concrete-domain operator asserting that a value is at most the stated bound.

```
⊑ Concrete value operator
```

Parents: ⊑∨ Concrete value operator

Since: Inception

id: `LessThanOrEqualTo` | UUID: `6dfacbd5-8344-5794-9fda-bec95b2aa6c9`

⊑ Literal value

Groups the literal kinds a concrete-domain axiom can compare against: a fixed value written into the expression itself, rather than a reference to a component.

```
⊑ Meaning
```

Parents: ⊑ Meaning

Children: ⊑ Instant literal ⊑ Float literal ⊑ Boolean literal

Since: Inception

id: `LiteralValue` | UUID: `208a40a7-e615-5efa-9de0-2e2a5a8488b7`

⊑ Lithuanian language

The Lithuanian language, as a value for a description's language field: a language coordinate that names it selects Lithuanian-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨ LANGUAGE

Since: Inception

id: `LithuanianLanguage` | UUID: `8fa63274-70e3-5b11-9669-1b7bdb372b1a`

⊑ Logic coordinate name

The field naming a stored logic coordinate, so a surface can present saved coordinates by name.

```
⊑ Logic coordinate properties
```

Parents: ⊑ LOGIC COORDINATE PROPERTIES

Since: Inception

id: `LogicCoordinateName` | UUID: `78972f14-e0f6-5f72-bf82-59310b5f7b26`

⊑ Logic coordinate properties

Groups the logic-coordinate dimensions: which reasoner classifies, which description-logic profile it assumes, which patterns carry stated and inferred axioms, and which root it classifies beneath.

```
⊑ View coordinate model
```

Parents: ⊑ VIEW COORDINATE MODEL

Children: ⊑ LOGIC COORDINATE NAME

Since: Inception

id: `LogicCoordinateProperties` | UUID: `1fa63819-5ac1-5938-95b1-47871a5f2b17`

⊑ Logic graph for semantic

The field holding a semantic's logical expression graph: the definition a logical-expression semantic carries.

```
⊑ Semantic properties
```

Parents: ⊑ SEMANTIC PROPERTIES

Since: Inception

id: `LogicGraphForSemantic` | UUID: `fc2a0662-2396-575b-95f0-e9b38a418620`

⊑ Logical axiom

The root of the logical-axiom taxonomy: one element of a stated or inferred definition's expression graph. Mirrors the sealed interface LogicalAxiom: every axiom is an atom, a definition root, or a logical set — a closed alternative the compiler enforces in code and this taxonomy states as knowledge.

```
⊑ Logical expression model
```

Parents: ⊑ LOGICAL EXPRESSION MODEL

Children: ⊑ LOGICAL SET ⊑ ATOM ⊑ DEFINITION ROOT

Since: Inception

id: `LogicalAxiom` | UUID: `9f622a2f-ed85-57a1-a1b0-c448127fb906`

⊑∨ Logical Definition

The semantic value describing the purpose of the stated and inferred terminological axioms.

```
⊑ Tinkar Model concept ⊓ Purpose
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ PURPOSE

Children: ⊑ EL++ STATED CONCEPT DEFINITION ⊑ EL++ INFERRED CONCEPT DEFINITION

Since: Inception

id: `LogicalDefinition` | UUID: `7dccd042-b0b8-5cec-a1bc-6de676b92f4b`

⊑ Logical expression display field

The data type for fields holding a logical expression: a definition graph carried as a DiTree.

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `LogicalExpressionDisplayField` | UUID: `c16eb414-8840-54f8-9bd2-e2f1ab37e19d`

⊑ Logical expression model

The meta-schema of stated and inferred definitions: the closed taxonomy of logical-axiom kinds — mirroring, one for one, the sealed LogicalAxiom hierarchy the platform's reasoner and builders compile against — together with the operator vocabularies those axioms use. What the compiler enforces as the closed set of axiom kinds, this subtree states as the closed taxonomy of axiom-kind concepts.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑ Reflexive Feature ⊑∨ Axiom Expression ⊑ Role operator ⊑ Annotation property set ⊑ Transitive Feature ⊑ Property Sequence ⊑ Logical axiom ⊑∨ Axiomatized Component ⊑ Feature Type

Since: Inception

id: `LogicalExpressionModel` | UUID: `048b509b-7446-5adb-a81b-245386981760`

⊑ Logical expression semantic

The semantic type whose field value is a logical expression: a definition graph carried as a DiTree.

```
⊑ Semantic type
```

Parents: ⊑ Semantic type

Since: Inception

id: `LogicalExpressionSemantic` | UUID: `d19306b1-4744-5028-a715-17ca4a4d657f`

⊑ Logical set

A logical axiom that groups atoms under a definition root and fixes their force: necessary (always true of the concept), sufficient (enough to classify an individual as the concept), property, data property, interval property, and inclusion sets. Mirrors LogicalAxiom.LogicalSet in the sealed hierarchy.

```
⊑ Logical axiom
```

Parents: ⊑ Logical axiom

Children: ⊑ Necessary but not sufficient concept definition ⊑ Implication set ⊑ Sufficient set ⊑ Data property set ⊑ Inclusion set ⊑ Data Property Set Axioms ⊑ Necessary set ⊑ Property set ⊑ Interval property set ⊑ Property set Axioms

Since: Inception

id: `LogicalSet` | UUID: `70eb421f-2489-5b4f-8699-686487e4593b`

⊑ Logically equivalent to

An operator for the reasoner to determine the equivalence

```
⊑ Taxonomy operator
```

Parents: ⊑ Taxonomy operator

Since: Inception

id: `LogicallyEquivalentTo` | UUID: `3642d9a3-8e23-5289-836b-366c0b1e2900`

⊑ Long

The retired dynamic-column type for long integer values, superseded by the Long field kind.

```
⊑ Dynamic column data types
```

Parents: ⊑ Dynamic column data types

Since: Inception

id: `Long` | UUID: `dea8cdf1-de75-5991-9791-79714e4a964d`

⊑∨ Long default

The Long field's loud default: the stretched-sevens sentinel 77777777777777777 (eighteen sevens) — a repdigit signature that reads as deliberately planted and cannot plausibly occur as natural data, where Long.MIN_VALUE would read as an overflow bug (and sits one away from the platform's pre-inception time sentinel). Never Java's silent zero.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `LongDefault` | UUID: `bb92b64a-d0f4-5ead-a2ec-a3d178527e10`

⊑ Lower Bound Open

The marker that an interval's lower bound is open: the bound value itself lies outside the interval.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `LowerBoundOpen` | UUID: `a0096ba1-0718-4c03-ad8f-8143c44091e7`

⊑ Master path

The path released consumer-facing content lands on: the mainline other paths promote into.

```
⊑ Path
```

Parents: ⊑ ∨ Path

Since: Inception

id: `MasterPath` | UUID: `1f20134a-960e-11e5-8994-feff819cdc9f`

⊑ ∨ Match Rule

Why the Member match relation value is recorded: to say how a constrained field's value must match an enumerated member — which member match relation the constraint holds under.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `MatchRule` | UUID: `0e389613-6adc-5998-b3e3-7fef8a167a67`

⊑ ∨ Maximum Value Operator

The Value Operator assigned to the Maximum Value in a Range

```
⊑ Concrete value operator ⊓ Meaning
```

Parents: ⊑ ∨ Concrete value operator ⊑ Meaning

Since: Inception

id: `MaximumValueOperator` | UUID: `7b8916ab-fd50-41df-8fc2-0b2a7a78be6d`

⊑ Meaning

The interpretation or explanation field for a pattern/semantics

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Defaulted Data Types ⊑∨ Long default ⊑∨ Value-set field ⊑∨ Is-a ⊑∨ Concept versions field ⊑∨ Author ⊑∨ Concept default ⊑∨ Path field ⊑∨ Path ⊑∨ Boolean default ⊑∨ Public ID field ⊑∨ Array default ⊑∨ Description acceptability ⊑∨ Narrative Prose Element ⊑∨ Roster order ⊑∨ Concept field ⊑∨ Constraint anchor concept ⊑∨ Relationship destination ⊑∨ Semantic default ⊑∨ Status field ⊑∨ Member match relation ⊑ Presentation unit different ⊑∨ Semantic field field ⊑ Sufficient concept definition operator ⊑ EL profile set operator ⊑∨ Description case significance ⊑∨ Great Britain English dialect ⊑∨ EL++ Inferred terminological axioms ⊑∨ Pattern versions field ⊑∨ DiGraph default ⊑∨ Starter set author roster ⊑∨ Stated Definition ⊑∨ ByteArray default ⊑∨ Semantic versions set ⊑∨ Semantic pattern field ⊑∨ Language for description ⊑∨ Concept pattern for logic coordinate ⊑∨ Identifier Source ⊑∨ Inferred Definition ⊑∨ EL++ Stated terminological axioms ⊑∨ Relationship origin ⊑∨ Version Properties ⊑∨ Constrained field ⊑ Query clauses ⊑∨ Status value ⊑∨ Reference Range Minimum ⊑∨ Comment ⊑∨ ComponentIdList default ⊑∨ Constrained Pattern ⊑∨ Roster author ⊑∨ Axiom Syntax ⊑∨ String default ⊑∨ Semantic referenced component field ⊑∨ Prose Text ⊑∨ Integer default ⊑∨ Identified Component ⊑∨ ComponentIdSet default ⊑∨ Time field ⊑∨ Originated Module ⊑∨ Starter Data Authoring ⊑ Text comparison measure semantic ⊑∨ Decimal default ⊑∨ Preferred reviewer assignment ⊑∨ Float default ⊑∨ DiTree default ⊑ Field substitution ⊑∨ Description type ⊑∨ Component versions field ⊑∨ Author field ⊑∨ Component field ⊑∨ Constraint kind ⊑∨ Reference Range Maximum ⊑∨ Component default ⊑∨ Description semantic ⊑∨ Pattern meaning field ⊑∨ Instant default ⊑∨ Identifier Value ⊑∨ Time ⊑ Taxonomy operator ⊑∨ Module field ⊑∨ United States of America English dialect ⊑∨ STAMP versions field ⊑∨ Preferred reviewer ⊑∨ Axiomatized Component ⊑∨ STAMP field ⊑∨ Semantic field ⊑∨ Semantic versions field ⊑∨ Module ⊑∨ Originated Path ⊑∨ Pattern purpose field ⊑∨ Path origins ⊑∨ Subject of Commentary ⊑∨ Module origins ⊑∨ Maximum Value Operator ⊑ Semantic type

⊑∨ Value-set pattern ⊑∨ Field definition field ⊑ Literal value
⊑∨ Minimum Value Operator ⊑∨ Value Constraint Source ⊑∨ Path concept
⊑∨ Example UCUM Units ⊑∨ Value Constraint ⊑∨ Pattern field
⊑∨ Text for description

Since: Inception

id: `Meaning` | UUID: `a06158ff-e08a-5d7d-bcfa-6cbfdb138910`

⊑∨ Meaning Declaration

Why a Pattern Version Pattern's Pattern meaning field is recorded: to hold this pattern-version's own declared meaning concept.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ Chronicle and version model ⊑ Purpose

Since: Inception

id: `MeaningDeclaration` | UUID: `864fa92e-244b-5eca-a566-ae3262bac9c5`

⊑∨ Member match relation

The closed taxonomy of relations a Value-set Field Constraint Pattern semantic can require between a constrained field's value and an enumerated member. Relations are directed and named in full, and a relation is admitted only when its evaluator ships in code: relation concepts correspond one-to-one with the service-loaded evaluators — the bijection gate.

```
⊑ Constraint model ⊓ Meaning
```

Parents: ⊑ Constraint model ⊑ Meaning

Children: ⊑ Equal match relation

Since: Inception

id: `MemberMatchRelation` | UUID: `8bd2739b-7209-53b4-9c98-adbb413dd415`

⊑∨ Membership semantic

The semantic type carrying no fields: a semantic whose entire content is that its referenced component belongs to the pattern's own set. The SOLOR-era name for what this starter set's own content now calls Set membership.

```
⊑ Semantic type ⊓ Purpose
```

Parents: ⊑ Semantic type ⊑ Purpose

Since: Inception

id: `MembershipSemantic` | UUID: `4fa29287-a80e-5f83-abab-4b587973e7b7`

⊑∨ Minimum Value Operator

The Value Operator assigned to the Minimum Value in a Range

```
⊑ Concrete value operator ⊓ Meaning
```

Parents: ⊑∨ Concrete value operator ⊑ Meaning

Since: Inception

id: `MinimumValueOperator` | UUID: `ded98e42-f74a-4432-9ae7-01b94dc2fdea`

⊑ Model concept

A concept representing a model construct within Integrated Knowledge Management — the structural and data-type terminology a knowledge base uses to describe itself, as distinct from the domain content it represents.

```
⊑ Integrated Knowledge Management
```

Parents: ⊤ Integrated Knowledge Management

Children: ⊑ Constraint model ⊑ Editorial model ⊑ IkeFoundation root
⊑ Data Concept ⊑ Provenance model ⊑ Logical expression model
⊑ View coordinate model ⊑ Legacy model concepts
⊑ Chronicle and version model ⊑ STAMP dimensions ⊑ Description model
⊑ Purpose ⊑ Meaning ⊑ Tinkar Model concept
⊑ Defaults and templates model ⊑ Graph model ⊑ Identifier model

Since: Inception

id: `ModelConcept` | UUID: `7bbd4210-381c-11e7-9598-0800200c9a66`

⊑∨ Module

The concept identifying the export or authoring boundary a version belongs to — the module dimension of a STAMP, and the root of the value space a module field's concept is drawn from; distinct from Module for version, which names why the field is recorded.

```
⊑ STAMP dimensions ⊓ Meaning
```

Parents: ⊑ STAMP DIMENSIONS ⊑ MEANING

Children: ⊑ DEFAULTS AND TEMPLATES MODULE ⊑ SOLOR MODULE
⊑ DEVELOPMENT MODULE ⊑ USERS MODULE ⊑ KOMET MODULE
⊑ SOLOR OVERLAY MODULE ⊑ IKEFOUNDATION MODULE ⊑ PRIMORDIAL MODULE
⊑ SANDBOX MODULE

Since: Inception

id: `Module` | UUID: `40d1c869-b509-32f8-b735-836eac577a67`

⊑ Module exclusion set for stamp coordinate

The stamp-coordinate dimension holding modules to ignore outright: a version in an excluded module is invisible to the view regardless of any preference order.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `ModuleExclusionSetForStampCoordinate` | UUID: `3fe047f0-33b0-5254-91c2-43e65f90d30b`

⊑ ∨ Module field

A concept field whose value is the module dimension of a STAMP — the module the version belongs to.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑ ∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `ModuleField` | UUID: `e6359a86-a1df-4721-8a1a-1f1f075ec3d9`

⊑ Module for user

The per-user editing default naming which module a user's edits land in.

```
⊑ View coordinate model
```

Parents: ⊑ VIEW COORDINATE MODEL

Since: Inception

id: `ModuleForUser` | UUID: `c8fd4f1b-d842-5245-9a7d-a58dc0ac1c11`

⊑∨ Module for version

Why a version's module field is recorded: to say which export and authoring boundary the version belongs to.

```
⊑ Version Properties ⊓ Purpose
```

Parents: ⊑∨ Version Properties ⊑ Purpose

Since: Inception

id: `ModuleForVersion` | UUID: `67cd64f1-96d7-5110-b847-556c055ac063`

⊑∨ Module Lineage

Why a module's own origin record is captured: to track which modules a given module originated from.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑ Provenance model ⊑ Purpose

Since: Inception

id: `ModuleLineage` | UUID: `f8c2a07e-8969-5123-874e-aa0d0ab3f6a0`

⊑ Module options for edit coordinate

The edit-coordinate dimension listing the modules an editor may choose to commit into.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `ModuleOptionsForEditCoordinate` | UUID: `19305aff-95d9-55d9-b015-825cc68eadc7`

⊑∨ Module origins

The origin-set value a Module origins pattern semantic carries: the set of modules a module originated from — what the pattern's one field holds, distinct from Originated Module, the module whose origins are declared.

```
⊑ Provenance model ⊓ Meaning
```

Parents: ⊑ Provenance model ⊑ Meaning

Since: Inception

id: `ModuleOrigins` | UUID: `462862d4-5df9-426e-b785-a1264e24769f`

P Module origins pattern

Pattern of module origins

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Originated Module	⊑∨ Module Lineage	—
Field 1		
⊑∨ Module origins	⊑∨ Origin Module Set	⊑ Component Id set data type

Since: Inception

id: `ModuleOriginsPattern` | UUID: `536b0ec4-4974-47ae-93a6-ae6c4d169780`

⊑ Module preference list for language coordinate

The language-coordinate dimension holding modules in preference order, for choosing among descriptions that differ only by module.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `ModulePreferenceListForLanguageCoordinate` | UUID: `f36e7ca6-34a2-58b5-8b25-736457515f9c`

⊑ Module preference list for stamp coordinate

The stamp-coordinate dimension listing the modules a view accepts; together with the exclusion set it makes module scoping affirmative.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `ModulePreferenceListForStampCoordinate` | UUID: `f56ef2df-6758-5271-a587-317a4fed6c2e`

⊑ Module preference order for stamp coordinate

The stamp-coordinate dimension ranking acceptable modules: when versions tie across modules, the earlier module in this order wins.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `ModulePreferenceOrderForStampCoordinate` | UUID: `ddeda759-e89c-5186-aa40-d63070756ab4`

⊑ Modules for stamp coordinate

The stamp-coordinate dimension holding the modules a view accepts versions from.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `ModulesForStampCoordinate` | UUID: `bf69c4f1-95c9-5956-a10a-d3ba9276c019`

⊑∨ Narrative Prose Element

The referenced-component role a Prose Element Pattern semantic's target plays: the hub concept the attached curated narrative documentation is centered on.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `NarrativeProseElement` | UUID: `2c2ec9cd-78b5-5d39-8b97-8c267d2d5293`

⊑ Navigation

Groups navigation: the precomputed parent and child adjacency structures a navigator renders, and the concepts that classify them.

```
⊑ Purpose
```

Parents: ⊑ PURPOSE

Children: ⊑ Navigation concept set ⊑ Inferred navigation
⊑ ∨ Taxonomy Navigation Cache ⊑ Stated navigation

Since: Inception

id: `Navigation` | UUID: `4d9707d8-adf0-5b15-89fc-039e4ff6fec8`

⊑ Navigation concept set

The navigation-coordinate dimension holding the concept set a navigator traverses, scoping navigation to a chosen part of the taxonomy.

```
⊑ Navigation
```

Parents: ⊑ Navigation

Since: Inception

id: `NavigationConceptSet` | UUID: `fc965c5d-ad17-555e-bcb5-b78fd45c8c5f`

⊑ Navigation vertex

A vertex as it appears in navigation: one node of the precomputed navigation graph, presenting the concept it stands for.

```
⊑ Vertex
```

Parents: ⊑ Vertex

Since: Inception

id: `NavigationVertex` | UUID: `c7f01834-34ca-5f8b-8f80-193fbeb12eae`

⊑ Necessary but not sufficient concept definition

The definition status marking a concept whose stated conditions are all necessary of it but not enough to classify an instance as it: a reasoner can check membership but never infer it. The contrast to Sufficient concept definition.

```
⊑ Logical set
```

Parents: ⊑ Logical set

Since: Inception

id: `NecessaryButNotSufficientConceptDefinition` | UUID: `e1a12059-3b01-3296-9532-d10e49d0afc3`

⊑ Necessary set

A set of relationships that is always true of a concept.

```
⊑ Logical set
```

Parents: ⊑ LOGICAL SET

Since: Inception

id: `NecessarySet` | UUID: `acaa2eba-8364-5493-b24c-b3884d34bb60`

⊑ NID

The store-local integer identifier a component holds while loaded: a fast internal handle, never an exchangeable identity -- identity crossing any boundary is a public identifier.

```
⊑ Identifier model
```

Parents: ⊑ IDENTIFIER MODEL

Since: Inception

id: `NID` | UUID: `d1a17272-9785-51aa-8bde-cc556ab32ebb`

⊑ Not Applicable

The case-significance value for text where case carries no meaning at all, so the question of sensitivity does not arise.

```
⊑ Description case significance
```

Parents: ⊑∨ DESCRIPTION CASE SIGNIFICANCE

Since: Inception

id: `NotApplicable` | UUID: `d4cc29ae-c0c1-563a-985d-5165a768dd44`

⊑∨ Numeric Value Restriction

Why a Value Constraint Pattern semantic exists: to restrict which literal values are legal for a component — the numeric-domain counterpart to Field Value Restriction.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ CONSTRAINT MODEL ⊑ PURPOSE

Since: Inception

id: `NumericValueRestriction` | UUID: `eea1cc9e-1f3a-50f3-8a37-c4c3cd0bf589`

⊑ Object

An encapsulation of data together with procedures

⊑ `Legacy model concepts`

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `Object` | UUID: `72765109-6b53-3814-9b05-34ebddd16592`

⊑ Object Properties

Objects are instances of classes, the properties describe the data or attributes that an object can have

⊑ `Legacy model concepts`

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `ObjectProperties` | UUID: `3ef4311c-70c0-5149-9e06-53d745f85b15`

⊑ Or

The disjunctive connective: an expression satisfying any one of its children satisfies the whole. Mirrors LogicalAxiom.Atom.Connective.Or in the sealed hierarchy.

⊑ `Connective operator`

Parents: ⊑ CONNECTIVE OPERATOR

Since: Inception

id: `Or` | UUID: `2c940bcf-22a8-5fc9-b232-580021e758ed`

⊑ Order for axiom attachments

The per-user display preference fixing the order an axiom's attached semantics are presented in.

⊑ `View coordinate model`

Parents: ⊑ VIEW COORDINATE MODEL

Since: Inception

id: `OrderForAxiomAttachments` | UUID: `abcb0946-20e1-5483-8469-3e8fa0ce20c4`

⊑ Order for concept attachments

The per-user display preference fixing the order a concept's attached semantics are presented in.

```
⊑ View coordinate model
```

Parents: ⊑ View coordinate model

Since: Inception

id: `OrderForConceptAttachments` | UUID: `6167efcb-50e8-534d-9827-fdd60b02ae00`

⊑ Order for description attachments

The per-user display preference fixing the order a description's attached semantics, such as dialect judgments, are presented in.

```
⊑ View coordinate model
```

Parents: ⊑ View coordinate model

Since: Inception

id: `OrderForDescriptionAttachments` | UUID: `69ee3f13-e2ba-5a96-9b91-5eecfad8e587`

⊑ ∨ Origin Module Set

Why a module's origin set value is recorded: to name the modules this module originated from.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑ Provenance model ⊑ Purpose

Since: Inception

id: `OriginModuleSet` | UUID: `77723bf6-084d-519b-9275-c6e6449aef0e`

⊑ ∨ Originated Module

What a Module origins pattern semantic's referenced component is: the module whose origins are declared — the module that originated from the modules the semantic names, distinct from Module origins, which names the origin-set value itself.

```
⊑ Provenance model ⊓ Meaning
```

Parents: ⊑ PROVENANCE MODEL ⊑ MEANING

Since: Inception

id: `OriginatedModule` | UUID: `57fb5067-c863-5745-8f21-4937e68d7611`

⊑∨ Originated Path

What a Path origins pattern semantic's referenced component is: the path whose branch point is declared — the path that originated from the path the semantic names, distinct from Path origins, which names the origin-record value itself.

```
⊑ Provenance model ⊓ Meaning
```

Parents: ⊑ PROVENANCE MODEL ⊑ MEANING

Since: Inception

id: `OriginatedPath` | UUID: `8383342f-ef73-5d8c-8d61-5000795b9f9d`

P OWL Axiom Syntax Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ AXIOMATIZED COMPONENT	⊑∨ AXIOM EXPRESSION	—
Field 1		
⊑∨ AXIOM SYNTAX	⊑∨ EXPRESS AXIOM SYNTAX	⊑ STRING DATA TYPE

Since: Inception

id: `OWLAxiomSyntaxPattern` | UUID: `c0ca180b-aae2-5fa1-9ab7-4a24f2dfe16b`

⊑ Part of

The taxonomy operator asserting partitive relation: one concept names a part of what another names, as distinct from a kind of it.

```
⊑ Taxonomy operator
```

Parents: ⊑ TAXONOMY OPERATOR

Since: Inception

id: `PartOf` | UUID: `b4c3f6f9-6937-30fd-8412-d0c77f8a7f73`

⊑ Partial

The correlation grouping value for a partial match: the expressions correspond in some but not all of their parts, as opposed to an exact correspondence.

```
⊑ Grouping
```

Parents: ⊑ GROUPING

Since: Inception

id: `Partial` | UUID: `a7f9574c-8e8b-515d-9c21-9896063cc3b8`

⊑∨ Path

A set of assets under version control that can be managed distinctly from other assets. Paths “branch” from other paths when established, and can be “merged” with other paths as well.

```
⊑ STAMP dimensions ⊓ Meaning
```

Parents: ⊑ STAMP DIMENSIONS ⊑ MEANING

Children: ⊑ MASTER PATH ⊑ SANDBOX PATH ⊑ PRIMORDIAL PATH ⊑ DEVELOPMENT PATH

Since: Inception

id: `Path` | UUID: `4459d8cf-5a6f-3952-9458-6d64324b27b7`

⊑∨ Path concept

A concept that is itself a version-control path — the concept kind a path is represented by. As the Path origins pattern's first field meaning it names the field holding the origin path a path branched from.

```
⊑ Concept type ⊓ Meaning
```

Parents: ⊑ CONCEPT TYPE ⊑ MEANING

Since: Inception

id: `PathConcept` | UUID: `1b9d9f95-fc0a-55ac-b2e6-7c8b37660046`

⊑ Path coordinate name

The field naming a stored path coordinate, so a surface can present saved coordinates by name.

```
⊑ Path coordinate properties
```

Parents: ⊑ PATH COORDINATE PROPERTIES

Since: Inception

id: `PathCoordinateName` | UUID: `a293a9a0-eb1e-5418-83c7-bec376b62245`

⊑ Path coordinate properties

Character or attribute of coordinates referring to a series of connected points, that form a shape or trajectory

```
⊑ View coordinate model
```

Parents: ⊑ VIEW COORDINATE MODEL

Children: ⊑ PROMOTION PATH FOR EDIT COORDINATE ⊑ PATH COORDINATE NAME

Since: Inception

id: `PathCoordinateProperties` | UUID: `ec41e427-f009-5e45-a643-6dc658d63d83`

⊑∨ Path field

A concept field whose value is the path dimension of a STAMP — the path the version was committed on.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `PathField` | UUID: `6622a391-e2e6-45a0-97e1-c58cd0184092`

⊑ Path for path coordinate

The path-coordinate dimension naming the version-control path a position sits on; with a time, it is the coordinate's complete position.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `PathForPathCoordinate` | UUID: `748e073c-fea7-58dd-8aa3-f18fdd82ddfc`

⊑ Path for user

The per-user editing default naming which version-control path a user's edits land on.

```
⊑ View coordinate model
```

Parents: ⊑ View coordinate model

Since: Inception

id: `PathForUser` | UUID: `12131382-1535-5a77-928b-6eacad221ea2`

⊑∨ Path for version

Why a version's path field is recorded: to say which version-control path the version was committed on.

```
⊑ Version Properties ⊓ Purpose
```

Parents: ⊑∨ Version Properties ⊑ Purpose

Since: Inception

id: `PathForVersion` | UUID: `ad3dd2dd-ddb0-584c-bea4-c6d9b91d461f`

⊑∨ Path Lineage

Why a path's own origin record is captured: to track which path it branched from, and when.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑ Provenance model ⊑ Purpose

Since: Inception

id: `PathLineage` | UUID: `386b515e-dae4-5dcd-8390-cd56d7c58bcc`

⊑ Path options for edit coordinate

The edit-coordinate dimension listing the paths an editor may choose to commit on.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `PathOptionsForEditCoordinate` | UUID: `2110c10c-9174-55aa-8ffe-91650c77d0b3`

⊑∨ Path origins

A path's own origin record — the path-coordinate property saying where a path branched from. As the Path origins pattern's second field meaning it names the origin instant that record fixes: the branch point up to which the origin path's versions are visible.

```
⊑ Provenance model ⊓ Meaning
```

Parents: ⊑ Provenance model ⊑ Meaning

Since: Inception

id: `PathOrigins` | UUID: `6e6a112e-7d8c-53c7-aaf1-c46e2d69743c`

⊑ Path origins for stamp path

The record of which path a path branched from, and when: each origin is a position up to which the origin path's versions are visible from the branching path.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `PathOriginsForStampPath` | UUID: `f33e1668-34dd-53dd-8728-31b29934b482`

P Path origins pattern

Pattern of path origins

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Originated Path	⊑∨ Path Lineage	—
e.g. ⊑ Master path		
Field 1		

Meaning	Purpose	Data type
⊑∨ Path concept	⊑∨ Branch Source	⊑ Component data type
e.g. ⊑ *Development path*		
Field 2		
⊑∨ Path origins	⊑∨ Branch Point	⊑ Instant literal
e.g. Latest		

Since: Inception

id: `PathOriginsPattern` | UUID: `70f89dd5-2cdb-59bb-bbaa-98527513547c`

P Pattern Chronology Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Pattern field	⊑∨ Chronicle Identity and History	—
Field 1		
⊑∨ Public ID field	⊑∨ Uniquely identify knowledge graph components	⊑ Component D
Field 2		
⊑∨ Pattern versions field	⊑∨ Pattern versions set	⊑ Component I

Since: Inception

id: `PatternChronologyPattern` | UUID: `5bc93adb-9d39-43fe-a7a4-1492245b7efb`

⊑∨ Pattern field

A component field narrowed to patterns: its value references a pattern. The Pattern Chronology Pattern's referenced-component meaning, and the parent of Semantic pattern field.

```
⊑ Component field ⊓ Meaning
```

Parents: ⊑∨ Component field ⊑ Meaning

Children: ⊑∨ Semantic pattern field

Since: Inception

id: `PatternField` | UUID: `751790c7-e1e4-42bc-b531-54c54bd6eebd`

⊑∨ Pattern meaning field

A concept field whose value is a pattern version's declared meaning concept — naming what a semantic of the pattern is about.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ Concept field ⊑ Meaning

Since: Inception

id: `PatternMeaningField` | UUID: `996d0023-a355-422f-a84d-16dda6ece1b0`

⊑∨ Pattern Membership

Why a Semantic Chronology Pattern's Semantic pattern field is recorded: to identify which pattern this semantic is an instance of.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ Chronicle and version model ⊑ Purpose

Since: Inception

id: `PatternMembership` | UUID: `3eeb5f15-b87e-5201-a214-0f1d8bbbf00e`

⊑∨ Pattern purpose field

A concept field whose value is a pattern version's declared purpose concept — naming why semantics of the pattern are captured.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ Concept field ⊑ Meaning

Since: Inception

id: `PatternPurposeField` | UUID: `352c821b-7a11-454c-a127-48ad3206573d`

P Pattern Version Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Pattern versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type
Field 2		
⊑∨ Pattern meaning field	⊑∨ Meaning Declaration	⊑ Component data type
Field 3		
⊑∨ Pattern purpose field	⊑∨ Purpose Declaration	⊑ Component data type
Field 4		
⊑∨ Field definition field	⊑∨ Field definitions set	⊑ Component Id set data type

Since: Inception

id: `PatternVersionPattern` | UUID: `a90f8a4d-ae13-476b-98b8-814914f9704e`

⊑∨ Pattern versions field

The component versions field narrowed to pattern chronicles: its value collects a pattern's own versions. The Pattern Version Pattern's referenced-component meaning.

```
⊑ Component versions field ⊓ Meaning
```

Parents: ⊑∨ Component versions field ⊑ Meaning

Since: Inception

id: `PatternVersionsField` | UUID: `7b8ecbbf-55b4-41bc-acbf-51824e74446a`

⊑∨ Pattern versions set

Why a pattern chronicle's versions field is recorded: to hold every version the pattern has ever had — the pattern-chronicle specialization of Component versions set.

```
⊑ Component versions set ⊓ Purpose
```

Parents: ⊑∨  Component versions set ⊑  Purpose

Since: Inception

id: `PatternVersionsSet` | UUID: `a254ccee-ef02-4504-9645-0a2ed7af955d`

⊑  Phenomenon

A unique thought, fact, or circumstance

```
⊑ Integrated Knowledge Management
```

Parents: ⊤  Integrated Knowledge Management

Children: ⊑  Uncategorized phenomenon ⊑  Health concept
⊑∨  Example UCUM Units

Since: Inception

id: `Phenomenon` | UUID: `c2e8bc47-3353-5e02-b0d1-2a5916efed4d`

⊑  Polish dialect

The Polish dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in Polish usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑  Dialect

Since: Inception

id: `PolishDialect` | UUID: `315cd879-1557-5a30-b325-a5d3df9e1c2b`

⊑  Polish Language

The Polish language, as a value for a description's language field: a language coordinate that names it selects Polish-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑∨  Language

Since: Inception

id: `PolishLanguage` | UUID: `c924b887-da88-3a72-b8ea-fa86990467c9`

⊑ Position on path

A position: a path plus a time on it, the point a stamp coordinate resolves latest against.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `PositionOnPath` | UUID: `31173582-a49d-51c6-813f-f42d0976aaea`

⊑ Preferred

The acceptability value marking a description as the one to show first in a dialect: at most one preferred description per description type per dialect.

```
⊑ Description acceptability
```

Parents: ⊑ ∨ DESCRIPTION ACCEPTABILITY

Since: Inception

id: `Preferred` | UUID: `266f1bc3-3361-39f3-bffe-69db9daea56e`

⊑ ∨ Preferred reviewer

Illustrative constrained field: the reviewer a Preferred Reviewer Pattern semantic names.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑ ∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `PreferredReviewer` | UUID: `bd91d84a-f50c-5d1c-b7be-af948dfa5d80`

⊑ ∨ Preferred reviewer assignment

Illustrative pattern whose single field is constrained to membership in the starter set author roster, demonstrating a Value-set Field Constraint Pattern end to end.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `PreferredReviewerAssignment` | UUID: `e9557b20-943e-505a-9a8e-dc6b666cdedf`

P Preferred Reviewer Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Preferred reviewer assignment	⊑∨ Review Routing	—
Field 1		
⊑∨ Preferred reviewer	⊑∨ Assigned Reviewer	⊑ Component data type
default ⊑ *Gretel*		

Since: Inception

id: `PreferredReviewerPattern` | UUID: `88b444be-682f-5981-8cb8-bd6f1cb4c53a`

⊑ Presentation unit different

Marks that a value is presented in a unit other than the one it was recorded in, so a reader knows a conversion stands between record and display.

```
⊑ Meaning
```

Parents: ⊑ Meaning

Since: Inception

id: `PresentationUnitDifferent` | UUID: `e86d3887-717b-545f-b6b5-611210913b23`

⊑ Primordial module

The module for content present from a store's beginning: the baseline's bootstrap module dimension.

```
⊑ Module
```

Parents: ⊑∨ Module

Since: Inception

id: `PrimordialModule` | UUID: `c2012321-3903-532e-8a5f-b13e4ca46e86`

⊑ Primordial path

The root of the path graph: the path from which every other path ultimately originates, and the one no path precedes. Released foundational knowledge lands here, beginning with a knowledge set's inception release, stamped at the Inception instant. Time on this path before inception is pre-inception (historically, premundane).

```
⊑ Path
```

Parents: ⊑∨ PATH

Since: Inception

id: `PrimordialPath` | UUID: `e95b6718-f824-5540-817b-8e79544eb97a`

⊑ Primordial state

Concept used to represent a status for components that have not yet been released and exist in their most basic form.

```
⊑ Status value
```

Parents: ⊑∨ STATUS VALUE

Since: Inception

id: `PrimordialState` | UUID: `b17bde5d-98ed-5416-97cf-2d837d75159d`

⊑ Primordial UUID for chronicle

The first universally unique identifier in a chronicle's public identity: the stable primary key among its identifiers.

```
⊑ Chronicle properties
```

Parents: ⊑ CHRONICLE PROPERTIES

Since: Inception

id: `PrimordialUUIDForChronicle` | UUID: `e0fcafbc-7191-5cdc-b14a-19d4d97f71bd`

⊑ Promotion Path for Edit Coordinate

The edit-coordinate dimension naming the path content is promoted to when moving between paths, such as sandbox work promoted to a mainline path.

⊑ Path coordinate properties

Parents: ⊑ PATH COORDINATE PROPERTIES

Since: Inception

id: `PromotionPathForEditCoordinate` | UUID: `db124d3b-c1bb-530e-8fd4-577f570355be`

⊑ Property pattern implication

A baseline duplicate of Property sequence implication, retired to Legacy: the platform's axiom meaning is Property sequence implication; this identity is retained for compatibility.

⊑ Legacy model concepts

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `PropertyPatternImplication` | UUID: `e0de0d09-6e27-5738-bc8f-0fc94bb115fc`

⊑ Property Sequence

An ordered sequence of properties, as used by a property-sequence implication: the chain whose composition is asserted to imply another property.

⊑ Logical expression model

Parents: ⊑ LOGICAL EXPRESSION MODEL

Since: Inception

id: `PropertySequence` | UUID: `d0d759fd-510f-475a-900e-b1439b4536e1`

⊑ Property sequence implication

The atom asserting a property-chain implication: the properties in the sequence, composed in order, imply the property this axiom names. Mirrors LogicalAxiom.Atom.PropertySequenceImplication.

⊑ Atom

Parents: ⊑ ATOM

Since: Inception

id: `PropertySequenceImplication` | UUID: `9a47a5db-42a6-49ee-9083-54bc305a9456`

⊑ Property set

The logical set grouping property axioms, such as transitivity and reflexivity declarations, under a definition root. Mirrors LogicalAxiom.LogicalSet.PropertySet.

```
⊑ Logical set
```

Parents: ⊑ LOGICAL SET

Since: Inception

id: `PropertySet` | UUID: `e273b5c0-c012-5e53-990c-aec5c2cb33a7`

⊑ Property set Axioms

A baseline grouping for property-set axioms, retained for identity compatibility: the platform's set meaning is Property set.

```
⊑ Logical set
```

Parents: ⊑ LOGICAL SET

Since: Inception

id: `PropertySetAxioms` | UUID: `ca2fdefd-0481-41cb-8074-41a78f94034d`

P Prose Element Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑ ∨ NARRATIVE PROSE ELEMENT	⊑ ∨ KNOWLEDGE-BASE DOCUMENTATION	—
e.g. ⊑ AUTHOR FOR EDIT COORDINATE		
Field 1		
⊑ ∨ PROSE TEXT	⊑ ∨ KNOWLEDGE-BASE DOCUMENTATION	⊑ STRING DATA TYPE

Meaning	Purpose	Data type
*e.g. Where a `StampCoordinate` only ever reads which version is current, an `EditCoordinate` names the STAMP an author's *next* edit will be stamped with — a distinct concern from any of the four dimensions above. k:AuthorForEditCoordinate[] names the author a new edit is attributed to (mentioned already in the Author subsection above); k:DefaultModuleForEditCoordinate[] names which module a new edit lands in by default, with k:ModuleOptionsForEditCoordinate[] naming the full set an author may choose from instead, and k:PathOptionsForEditCoordinate[] playing the analogous role for path. k:PromotionPathForEditCoordinate[] names a distinct idea again: not where a new edit is committed, but which path a component is *promoted* to once its authoring is complete — the sandbox-to-mainline movement k:SandboxPath[]/k:SandboxComponent[] content eventually undergoes. k:DestinationModuleForEditCoordinate[] rounds this out for the analogous author-to-export-module handoff.*		

Since: Inception

id: `ProseElementPattern` | UUID: `b150c682-25cc-50c8-bad5-c1c8b1ff4519`

⊑∨ Prose Text

The single String field of a Prose Element Pattern semantic: an embedded AsciiDoc prose block carrying id-bearing k: tokens, re-parsed as real AsciiDoc when spliced into the guide.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `ProseText` | UUID: `6e8c6be4-0a84-5952-9f91-61c4d90a21d5`

⊑ Provenance model

The meta-schema of where content came from: the role concepts of commit provenance, module lineage (which modules a module originated from), and path lineage (which path a path branched from, and when). The STAMP dimensions themselves are operational value spaces at the taxonomy root; this subsystem holds the concepts that describe provenance relationships among them.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑∨ Branch Source ⊑∨ Module Lineage ⊑ Creative Commons BY license ⊑∨ Commit Provenance ⊑∨ Originated Module ⊑∨ Version Provenance ⊑∨ Starter Data Authoring ⊑∨ Origin Module Set ⊑∨ Path Lineage ⊑∨ Originated Path ⊑∨ Path origins ⊑∨ Module origins ⊑∨ Branch Point

Since: Inception

id: `ProvenanceModel` | UUID: `a93e37fd-7333-5577-8cba-8cd4a0823612`

⊑∨ Public ID field

The field kind holding a chronicle's public identity — the universally unique identifiers that name a component across every system it is exchanged with. The first field of every chronicle-shape pattern, recorded to uniquely identify knowledge graph components.

```
⊑ String field ⊓ Meaning
```

Parents: ⊑ String field ⊑ Meaning

Since: Inception

id: `PublicIDField` | UUID: `196838c5-55f4-4e40-8618-b9ce60685c2f`

⊑ Purpose

The reason for which a Tinkar value in a pattern was created or for which it exist.

```
⊑ Model concept
```

Parents: ⊑ Model concept

Children: ⊑∨ Branch Source ⊑∨ Description Attachment ⊑∨ Is-a ⊑∨ Knowledge-Base Documentation ⊑∨ Purpose Declaration ⊑∨ Identifier Authority ⊑∨ Module Lineage ⊑∨ Axiom Expression ⊑∨ Identifier Text ⊑∨ Description acceptability ⊑∨ Status for version ⊑∨ Concrete value operator ⊑∨ Language ⊑∨ External Identity Mapping ⊑∨ Numeric Value Restriction ⊑∨ Description Role ⊑∨ STAMP versions set ⊑∨ Case Sensitivity Rule ⊑∨ Express axiom syntax ⊑∨ Set membership ⊑∨ Semantic field fields set ⊑∨ Version Snapshot ⊑∨ Constraint Scope ⊑∨ Meaning Declaration ⊑∨ Commit Provenance ⊑∨ Pattern Membership ⊑∨ Version Provenance ⊑∨ Taxonomy Reference Point ⊑∨ Match Rule ⊑∨ Origin Module Set ⊑∨ Taxonomy Navigation Cache ⊑∨ Roster Membership ⊑∨ Value Disambiguation ⊑∨ Reference Range ⊑∨ Assigned Reviewer ⊑∨ Description semantic ⊑∨ Attachment Target ⊑∨ Author for version ⊑∨ Unit Example ⊑∨ Uniquely identify knowledge graph components

⊑∨  Membership semantic ⊑∨  Pattern versions set ⊑∨  Path Lineage
⊑∨  Time for version ⊑∨  Chronicle Identity and History ⊑∨  Constraint Rule
⊑∨  Review Routing ⊑∨  Description ⊑∨  Display Sequence ⊑∨  Path for version
⊑∨  Concept versions set ⊑∨  Legal Value Source ⊑∨  Data Type Default Provision
⊑∨  Module for version ⊑∨  Branch Point ⊑∨  Field definitions set
⊑∨  Logical Definition ⊑∨  Version History ⊑∨  Component versions set
⊑∨  Reference Range Authority ⊑∨  Roster Entry ⊑∨  Editorial Clarification
⊑∨  Field Value Restriction ⊑∨  Data type default exemplar ⊑  Navigation

Since: Inception

id: `Purpose` | UUID: `c3dffc48-6493-54df-a2f0-14be8ba03091`

⊑∨  Purpose Declaration

Why a Pattern Version Pattern's Pattern purpose field is recorded: to hold this pattern-version's own declared purpose concept.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑  Chronicle and version model ⊑  Purpose

Since: Inception

id: `PurposeDeclaration` | UUID: `2d841ab9-c435-5ce6-8e4b-8308daf22bab`

⊑  Query clauses

A distinct component/query that serves a specific purpose

```
⊑ Meaning
```

Parents: ⊑  Meaning

Children: ⊑  View coordinate key ⊑  Boolean reference

Since: Inception

id: `QueryClauses` | UUID: `4905348c-ba1d-58ae-821f-52877d9acee3`

⊑∨ Reference Range

The range of values specific to a component

```
⊑ Tinkar Model concept ⊓ Purpose
```

Parents: ⊑  Tinkar Model concept ⊑  Purpose

Children: ⊑∨ Reference Range Minimum ⊑∨ Reference Range Maximum

Since: Inception

id: `ReferenceRange` | UUID: `87ce975b-309b-47f4-a6c6-4ae6df6649a1`

⊑∨ Reference Range Authority

Why a value constraint's source value is recorded: to name the organization that specifies the constraint.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `ReferenceRangeAuthority` | UUID: `4e756e4e-be8d-583c-bbb4-97b43438edfe`

⊑∨ Reference Range Maximum

The upper bound of a reference range: the greatest value the range admits, paired with an operator saying whether the bound itself is included.

```
⊑ Reference Range ⊓ Meaning
```

Parents: ⊑∨ Reference Range ⊑ Meaning

Since: Inception

id: `ReferenceRangeMaximum` | UUID: `72d58983-b1e1-4ca9-833f-0e40c1defd39`

⊑∨ Reference Range Minimum

The lower bound of a reference range: the least value the range admits, paired with an operator saying whether the bound itself is included.

```
⊑ Reference Range ⊓ Meaning
```

Parents: ⊑∨ Reference Range ⊑ Meaning

Since: Inception

id: `ReferenceRangeMinimum` | UUID: `37c35a88-9e27-42ca-b626-186773c4b612`

⊑ Referenced component for semantic

The field naming the component a semantic is attached to, in the semantic-properties vocabulary. Duplicates Component for semantic and the Semantic referenced component field kind; retained for identity compatibility.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `ReferencedComponentForSemantic` | UUID: `a9ba4749-c11f-5f35-a991-21796fb89ddc`

⊑ Referenced component subtype restriction

Restricts a pattern's referenced component to a subtree beneath a named concept, narrowing the kind-level Referenced component type restriction to particular content.

```
⊑ Role operator
```

Parents: ⊑ ROLE OPERATOR

Since: Inception

id: `ReferencedComponentSubtypeRestriction` | UUID: `8af1045e-1122-5072-9f29-ce7da9337915`

⊑ Referenced component type restriction

Restricts which component KIND a pattern's referenced component may be -- concept, semantic, pattern, or STAMP -- checked when an editor supplies one. The coarse restriction; its subtype counterpart narrows further, to a particular subtree.

```
⊑ Role operator
```

Parents: ⊑ ROLE OPERATOR

Since: Inception

id: `ReferencedComponentTypeRestriction` | UUID: `902f97b6-2ef4-59d7-b6f9-01278a00061c`

⊑ Reflexive Feature

Marks a property as reflexive: everything stands in that relation to itself.

```
⊑ Logical expression model
```

Parents: ⊑ LOGICAL EXPRESSION MODEL

Since: Inception

id: `ReflexiveFeature` | UUID: `7e779e4a-61ed-5c4a-aacc-03cf524b7c73`

⊑ Regular name description type

There may be descriptions/synonyms marked as “regular.”

```
⊑ Description type
```

Parents: ⊑∨ DESCRIPTION TYPE

Since: Inception

id: `RegularNameDescriptionType` | UUID: `8bfba944-3965-3946-9bcb-1e80a5da63a2`

⊑∨ Relationship destination

Signifies path to child concepts which are more specific than the Tinkar term

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `RelationshipDestination` | UUID: `a3dd69af-355c-54ce-ba13-2902a7ae9551`

⊑∨ Relationship origin

Signifies path to parent concepts which are more general than the Tinkar term

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `RelationshipOrigin` | UUID: `ad22d43b-3ee7-550b-9660-a6e68af347c2`

⊑∨ Review Routing

Why a Preferred Reviewer Pattern semantic exists: to direct a component's future edits to a specific reviewer.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑ EDITORIAL MODEL ⊑ PURPOSE

Since: Inception

id: `ReviewRouting` | UUID: `e1cc2998-97f6-5b8c-847f-587af5c47df8`

⊑ Role

Is an abstract representation of a high-level role for a therapeutic medicinal product; the concepts are not intended to describe a detailed indication for therapeutic use nor imply that therapeutic use is appropriate in all clinical situations.

```
⊑ Typed atom
```

Parents: ⊑ TYPED ATOM

Children: ⊑ ROLE TYPE ⊑ ROLE RESTRICTION

Since: Inception

id: `Role` | UUID: `46ae9325-dd24-5008-8fda-80cf1f0977c7`

⊑ Role group

An association between a set of attribute or axiom value pairs that causes them to be considered together within a concept definition or post coordinated expression.

```
⊑ Role type
```

Parents: ⊑ ROLE TYPE

Since: Inception

id: `RoleGroup` | UUID: `a63f4bf2-a040-11e5-8994-feff819cdc9f`

⊑ Role operator

Concept that is used to describe universal vs existential restrictions.

```
⊑ Logical expression model
```

Parents: ⊑ LOGICAL EXPRESSION MODEL

Children: ⊑ EXISTENTIAL RESTRICTION ⊑ REFERENCED COMPONENT TYPE RESTRICTION ⊑ UNIVERSAL RESTRICTION ⊑ REFERENCED COMPONENT SUBTYPE RESTRICTION

Since: Inception

id: `RoleOperator` | UUID: `f9860cb8-a7c7-5743-9d7c-ffc6e8a24a0f`

⊑ Role restriction

A restriction on which concepts may fill a role: the legacy carrier of role-value constraints.

```
⊑ Role
```

Parents: ⊑ ROLE

Since: Inception

id: `RoleRestriction` | UUID: `988bb02a-9b4a-4ef9-937e-fa8a6afc6c42`

⊑ Role type

Refers to a concept that represents a particular kind of relationship that can exist between two entities. It defines the specific function or responsibility that one entity plays in relation to another.

```
⊑ Role
```

Parents: ⊑ ROLE

Children: ⊑ INTERVAL ROLE TYPE ⊑ LATERALITY ⊑ ROLE GROUP ⊑ HAS ACTIVE INGREDIENT ⊑ FEATURE ROLE TYPE ⊑ HAS DOSE FORM

Since: Inception

id: `RoleType` | UUID: `76320274-be2a-5ba0-b3e8-e6d2e383ee6a`

⊑ Role type to add

The retired action parameter naming which role type a rule-driven action would add to a concept's definition.

```
⊑ Action properties
```

Parents: ⊑ ACTION PROPERTIES

Since: Inception

id: `RoleTypeToAdd` | UUID: `0c3ca846-0374-5a5c-8da4-67e0e2e28868`

⊑ Root for logic coordinate

The logic-coordinate dimension naming the taxonomy root a classification is computed beneath.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑  IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `RootForLogicCoordinate` | UUID: `862cc189-bbcb-51a0-89a4-16e1854be247`

⊑∨  Roster author

The author concept named by one roster entry.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨  CONCEPT FIELD ⊑  MEANING

Since: Inception

id: `RosterAuthor` | UUID: `c6075b23-9497-5059-bec3-41cd43168d59`

⊑∨  Roster Entry

Why a roster author value is recorded: to name one member of the roster.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑  EDITORIAL MODEL ⊑  PURPOSE

Since: Inception

id: `RosterEntry` | UUID: `caf2fcea-264f-5501-8aaf-c4de674b4098`

⊑∨  Roster Membership

Why a Starter Set Author Roster Pattern semantic exists: to enumerate this starter set's own author roster as a value-set source.

```
⊑ Editorial model ⊓ Purpose
```

Parents: ⊑  EDITORIAL MODEL ⊑  PURPOSE

Since: Inception

id: `RosterMembership` | UUID: `8ac7bd2d-3db5-5644-8575-a1591bc3a1c0`

⊑∨  Roster order

The roster entry's display order — the "additional characteristic" field a value-set constraint's Value-set field disambiguates against.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `RosterOrder` | UUID: `dea12527-af2c-5c05-9f6e-ccab55036d19`

⊑ Russian dialect

The Russian dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in Russian usage; a language coordinate's dialect order consults it when choosing what to show.

```
⊑ Dialect
```

Parents: ⊑ DIALECT

Since: Inception

id: `RussianDialect` | UUID: `300126d1-2604-579f-a59b-e3c1179a173a`

⊑ Russian language

The Russian language, as a value for a description's language field: a language coordinate that names it selects Russian-language descriptions for rendering.

```
⊑ Language
```

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `RussianLanguage` | UUID: `0818dbb7-3fe1-59d7-99c2-c8dc42ff2cce`

⊑ Sandbox component

An ISAAC-era marker for components in sandbox experimentation, retired to Legacy: sandbox scoping is carried by module and path dimensions.

```
⊑ Legacy model concepts
```

Parents: ⊑ LEGACY MODEL CONCEPTS

Since: Inception

id: `SandboxComponent` | UUID: `c93829b2-aa78-5a84-ac9a-c34307844166`

⊑ Sandbox module

The module for sandbox experimentation content.

```
⊑ Module
```

Parents: ⊑∨ MODULE

Children: ⊑ SANDBOX PATH MODULE

Since: Inception

id: `SandboxModule` | UUID: `c5daf0e9-30dc-5b3e-a521-d6e6e72c8a95`

⊑ Sandbox path

A path for components under testing.

```
⊑ Path
```

Parents: ⊑∨ PATH

Since: Inception

id: `SandboxPath` | UUID: `80710ea6-983c-5fa0-8908-e479f1f03ea9`

⊑ Sandbox path module

The module for content specific to a sandbox path.

```
⊑ Sandbox module
```

Parents: ⊑ SANDBOX MODULE

Since: Inception

id: `SandboxPathModule` | UUID: `715bd36d-6090-5b37-8ae7-88c9e532010e`

P Semantic Chronology Pattern

Pattern shape

Meaning	Purpose	Dat
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ SEMANTIC FIELD	⊑∨ CHRONICLE IDENTITY AND HISTORY	—

Meaning	Purpose	Dat
Field 1		
⊑∨ Public ID field	⊑∨ Uniquely identify knowledge graph components	⊑
Field 2		
⊑∨ Semantic pattern field	⊑∨ Pattern Membership	⊑
Field 3		
⊑∨ Semantic referenced component field	⊑∨ Attachment Target	⊑
Field 4		
⊑∨ Semantic versions set	⊑∨ Version History	⊑

Since: Inception

id: `SemanticChronologyPattern` | UUID: `5f0ad6ca-638e-4052-82b0-3f564ac99b3f`

⊑ Semantic data type

A field that holds a reference to a semantic.

```
⊑ Display Fields
```

Parents: ⊑ Display Fields

Since: Inception

id: `SemanticDataType` | UUID: `9c3dfc88-51e4-5e51-a59a-88dd580162b7`

⊑∨ Semantic default

The Semantic field's loud default: the fully qualified name description semantic of Uninitialized Component — a real, resolvable semantic whose subject is itself the loud placeholder, so the reference is valid yet plainly unrevised.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ Defaults and templates model ⊑ Meaning

Since: Inception

id: `SemanticDefault` | UUID: `5f7efa76-4723-57aa-b099-20769f7a6a1f`

⊑∨ Semantic field

A component field narrowed to semantics: its value references a semantic. The Semantic Chronology Pattern's referenced-component meaning — the chronicle kind that must also record which pattern its semantic is an instance of and what it is attached to.

```
⊑ Component field ⊓ Meaning
```

Parents: ⊑∨ Component field ⊑ Meaning

Since: Inception

id: `SemanticField` | UUID: `8b6c69d7-a5aa-4db2-bcea-8c7b2817b02f`

⊑ Semantic field concepts

The concept type for concepts that serve as semantic field values in the base model.

```
⊑ Concept type
```

Parents: ⊑ Concept type

Since: Inception

id: `SemanticFieldConcepts` | UUID: `b4316cb8-14fe-5b32-b03b-f5f966c87819`

⊑∨ Semantic field field

The field kind naming a semantic version's field-value slot: each value is one field the version actually carries, as distinct from the field definition its pattern declares.

```
⊑ Field categories ⊓ Meaning
```

Parents: ⊑ Field categories ⊑ Meaning

Since: Inception

id: `SemanticFieldField` | UUID: `f6572c76-b5c0-41da-99c0-4344694e7e3c`

⊑∨ Semantic field fields set

Why a semantic version's field-value field is recorded: to hold, together, every field value that version carries — the value-side counterpart to a pattern's Field definitions set.

```
⊑ Field categories ⊓ Purpose
```

Parents: ⊑ Field categories ⊑ Purpose

Since: Inception

id: `SemanticFieldFieldsSet` | UUID: `8dcfc1a1-31f2-46f7-8247-0a17a6d7c6c0`

⊑ Semantic field name

The field holding a display name for one of a semantic's fields, for surfaces that label fields individually.

```
⊑ Semantic properties
```

Parents: ⊑ SEMANTIC PROPERTIES

Since: Inception

id: `SemanticFieldName` | UUID: `15489c68-673d-503e-bff7-e9d59e5dc15c`

⊑ Semantic list for chronicle

The field holding every semantic attached to a chronicle's component -- its descriptions, identifiers, axioms, and memberships.

```
⊑ Chronicle properties
```

Parents: ⊑ CHRONICLE PROPERTIES

Since: Inception

id: `SemanticListForChronicle` | UUID: `c809b2c0-9235-5f64-bbda-34210d91bdf8`

⊑ ∨ Semantic pattern field

A pattern field whose value identifies which pattern a semantic is an instance of — the field a semantic chronicle records for pattern membership.

```
⊑ Pattern field ⊓ Meaning
```

Parents: ⊑ ∨ PATTERN FIELD ⊑ MEANING

Since: Inception

id: `SemanticPatternField` | UUID: `19dd5dd3-1075-4113-a437-5f1f7c2d55bc`

⊑ Semantic properties

The attributes or characteristics of a concept, term, or element that convey meaning or semantics in a given context

```
⊑ Chronicle and version model
```

Parents: ⊑ Chronicle and version model

Children: ⊑ Semantic field name ⊑ Logic graph for semantic ⊑∨ Uniquely identify knowledge graph components

Since: Inception

id: `SemanticProperties` | UUID: `b717ae48-5488-5dda-a980-97855001cc99`

⊑∨ Semantic referenced component field

A component field whose value identifies the component a semantic is attached to — the field a semantic chronicle records for its attachment target.

```
⊑ Component field ⊓ Meaning
```

Parents: ⊑∨ Component field ⊑ Meaning

Since: Inception

id: `SemanticReferencedComponentField` | UUID: `4111ba1e-c818-4c5d-9fed-34d07298d009`

⊑ Semantic type

Groups semantics by the kind of value they carry: concept, component, logical expression, or bare membership.

```
⊑ Meaning
```

Parents: ⊑ Meaning

Children: ⊑ Logical expression semantic ⊑ Component semantic ⊑∨ Membership semantic ⊑ Concept semantic

Since: Inception

id: `SemanticType` | UUID: `3daac6c4-78c5-5271-9c63-6e28f80e0c52`

P Semantic version field pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Semantic versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type
Field 2		
⊑∨ Semantic field field	⊑∨ Semantic field fields set	⊑ Component Id set data type

Since: Inception

id: `SemanticVersionFieldPattern` | UUID: `82f93e84-cee1-44bc-bb6d-4cc2a722048b`

⊑∨ Semantic versions field

The component versions field narrowed to semantic chronicles: its value collects a semantic's own versions. The Semantic version field pattern's referenced-component meaning.

```
⊑ Component versions field ⊓ Meaning
```

Parents: ⊑∨ Component versions field ⊑ Meaning

Since: Inception

id: `SemanticVersionsField` | UUID: `aeb73410-a679-4ea8-93fe-7c4785599778`

⊑∨ Semantic versions set

Why a semantic chronicle's versions field is recorded: to hold every version the semantic has ever had — the semantic-chronicle specialization of Component versions set.

```
⊑ Component versions set ⊓ Meaning
```

Parents: ⊑∨ Component versions set ⊑ Meaning

Since: Inception

id: `SemanticVersionsSet` | UUID: `4fd69aed-556f-4938-94cc-ea7ea707ccef`

⊑∨ Set membership

A component's role as an element of a pattern's own set of semantics — the modern replacement for the deprecated SOLOR term Membership semantic.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ CHRONICLE AND VERSION MODEL ⊑ PURPOSE

Since: Inception

id: `SetMembership` | UUID: `a4716ecb-2838-504c-9501-e69948bd4e62`

⊑ Signed integer

The retired dynamic-column type for signed whole numbers, superseded by Integer data type.

```
⊑ Dynamic column data types
```

Parents: ⊑ DYNAMIC COLUMN DATA TYPES

Since: Inception

id: `SignedInteger` | UUID: `1d1c2073-d98b-3dd3-8aad-a19c65aa5a0c`

⊑ SnoRocket classifier

The SnoRocket reasoner's identity: inferred content it writes is stamped with this concept as author, the classifier-as-author provenance model.

```
⊑ Author
```

Parents: ⊑∨ AUTHOR

Since: Inception

id: `SnoRocketClassifier` | UUID: `1f201fac-960e-11e5-8994-feff819cdc9f`

P SOLOR concept assemblage

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONCEPT PATTERN FOR LOGIC COORDINATE	⊑∨ MEMBERSHIP SEMANTIC	—

Since: Inception

id: `SOLORConceptAssemblage` | UUID: `d39b3ecd-9a80-5009-a8ac-0b947f95ca7c`

P Solor Concepts Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONCEPT PATTERN FOR LOGIC COORDINATE	⊑∨ SET MEMBERSHIP	—

Since: Inception

id: `SolorConceptsPattern` | UUID: `bf3a840d-fe61-5a1f-87ae-b023768fb476`

⊑ SOLOR module

The module for SOLOR core content: the inherited upstream content boundary.

```
⊑ Module
```

Parents: ⊑∨ MODULE

Since: Inception

id: `SOLORModule` | UUID: `f680c868-f7e5-5d0e-91f2-615eca8f8fd2`

⊑ SOLOR overlay module

The module for SOLOR overlay content: upstream additions layered over the core module.

```
⊑ Module
```

Parents: ⊑∨ MODULE

Since: Inception

id: `SOLOROverlayModule` | UUID: `9ecc154c-e490-5cf8-805d-d2865d62aef3`

⊑ Spanish language

The Spanish language, as a value for a description's language field: a language coordinate that names it selects Spanish-language descriptions for rendering.

⊑ Language

Parents: ⊑∨ LANGUAGE

Since: Inception

id: SpanishLanguage | UUID: 0fcf44fb-d0a7-3a67-bc9f-eb3065ed3c8e

P STAMP Chronology Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STAMP FIELD	⊑∨ CHRONICLE IDENTITY AND HISTORY	—
Field 1		
⊑∨ PUBLIC ID FIELD	⊑∨ UNIQUELY IDENTIFY KNOWLEDGE GRAPH COMPONENTS	⊑ COMPONENT DA
Field 2		
⊑∨ STAMP VERSIONS FIELD	⊑∨ STAMP VERSIONS SET	⊑ COMPONENT ID

Since: Inception

id: STAMPChronologyPattern | UUID: e16abc7a-2a7b-42af-b168-d77aec8116ea

⊑ STAMP dimensions

The five value spaces the versioning machinery selects a version's STAMP constituents from: Status value, Time, Author, Module, and Path. Machinery, so it files under Model concept — yet each dimension's subtree is operational content, not meta-schema: the statuses versions actually carry, the authors who actually commit, the modules and paths content actually lives on. A dimension root's double duty as a STAMP field meaning does not make its subtree a description of anything.

⊑ Model concept

Parents: ⊑ MODEL CONCEPT

Children: ⊑∨ AUTHOR ⊑∨ PATH ⊑∨ STATUS VALUE ⊑∨ TIME ⊑∨ MODULE

Since: Inception

id: `STAMPDimensions` | UUID: `aea9ca27-3c50-522f-acda-ae881f954603`

⊑∨ STAMP field

A component field whose value references a STAMP — the five-dimension provenance tuple a version was committed with. The STAMP Chronology Pattern's referenced-component meaning, and the first field of every version-shape pattern, recorded for version provenance.

```
⊑ Component field ⊓ Meaning
```

Parents: ⊑∨ Component field ⊑ Meaning

Since: Inception

id: `STAMPField` | UUID: `3d821e64-a2ee-4414-8949-1bc92ef5d5b6`

P STAMP pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Version Properties	⊑∨ Commit Provenance	—
Field 1		
⊑∨ Status value	⊑∨ Status for version	⊑ Component data type
Field 2		
⊑∨ Time	⊑∨ Time for version	⊑ Long
Field 3		
⊑∨ Author	⊑∨ Author for version	⊑ Component data type
Field 4		
⊑∨ Module	⊑∨ Module for version	⊑ Component data type
Field 5		
⊑∨ Path	⊑∨ Path for version	⊑ Component data type

Since: Inception

id: `STAMPPattern` | UUID: `9fd67fee-abf9-551d-9d0e-76a4b1e8b4ee`

P STAMP version field pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STAMP versions field	⊑∨ Version Snapshot	—
Field 1		
⊑∨ STAMP field	⊑∨ Version Provenance	⊑ Component data type
Field 2		
⊑∨ Status field	⊑∨ Status for version	⊑ Component data type
Field 3		
⊑∨ Time field	⊑∨ Time for version	⊑ String data type
Field 4		
⊑∨ Author field	⊑∨ Author for version	⊑ Component data type
Field 5		
⊑∨ Module field	⊑∨ Module for version	⊑ Component data type
Field 6		
⊑∨ Path field	⊑∨ Path for version	⊑ Component data type

Since: Inception

id: `STAMPVersionFieldPattern` | UUID: `73c798cf-bc77-49a2-84f7-4c0f4bc4c012`

⊑∨ STAMP versions field

The component versions field narrowed to STAMP chronicles: its value collects a STAMP's own versions. The STAMP version field pattern's referenced-component meaning.

```
⊑ Component versions field ⊓ Meaning
```

Parents: ⊑∨  Component versions field ⊑  Meaning

Since: Inception

id: `STAMPVersionsField` | UUID: `b8251bea-4248-4a46-8b4a-349500693a9f`

⊑∨  STAMP versions set

Why a STAMP chronicle's versions field is recorded: to hold every version the STAMP has ever had — the STAMP-chronicle specialization of Component versions set.

```
⊑ Component versions set ⊓ Purpose
```

Parents: ⊑∨  Component versions set ⊑  Purpose

Since: Inception

id: `STAMPVersionsSet` | UUID: `edb90567-7822-4129-a406-b359b825f922`

⊑  Standard Korean dialect

The standard Korean dialect: the general-usage dialect of Korean, as opposed to any specialized register.

```
⊑ Korean dialect
```

Parents: ⊑  Korean dialect

Since: Inception

id: `StandardKoreanDialect` | UUID: `f90722cc-5e40-5b9b-a2a6-f4dfa312a6a9`

⊑∨  Starter Data Authoring

Define necessary minimum viable concepts to use Tinkar Data

```
⊑ Provenance model ⊓ Meaning
```

Parents: ⊑  Provenance model ⊑  Meaning

Since: Inception

id: `StarterDataAuthoring` | UUID: `070deb74-acc5-46bf-b9c6-eaee1b58ef52`

⊑∨  Starter set author roster

Illustrative value-set source: an ordered roster of this starter set's own authors, demonstrating a Value-set Field Constraint Pattern's value-set pattern.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `StarterSetAuthorRoster` | UUID: `73077978-3e44-5ab3-a056-2dc88151a33e`

P Starter Set Author Roster Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STARTER SET AUTHOR ROSTER	⊑∨ ROSTER MEMBERSHIP	—
e.g. P STARTER SET AUTHOR ROSTER PATTERN		
Field 1		
⊑∨ ROSTER AUTHOR	⊑∨ ROSTER ENTRY	⊑ COMPONENT DATA TYPE
e.g. ⊑ GRETEL		
Field 2		
⊑∨ ROSTER ORDER	⊑∨ DISPLAY SEQUENCE	⊑ LONG
e.g. 1		

Since: Inception

id: `StarterSetAuthorRosterPattern` | UUID: `47072664-f25d-5c58-9850-4124240c4e97`

⊑∨ Stated Definition

Relationships/Axioms of a concept that have been explicitly stated and defined

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `StatedDefinition` | UUID: `28608bd3-ac73-4fe8-a5f0-1efe0d6650a8`

⊑ Stated navigation

Navigation computed from stated definitions: parent and child adjacency as authored, before classification.

```
⊑ Navigation
```

Parents: ⊑ NAVIGATION

Since: Inception

id: `StatedNavigation` | UUID: `614017af-9903-53d9-aab4-15fd02193dce`

P Stated Navigation Pattern

A pattern specifying the origins and destinations for concepts based on the stated terminological axioms.

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ IS-A	⊑∨ TAXONOMY NAVIGATION CACHE	—
Field 1		
⊑∨ RELATIONSHIP DESTINATION	⊑∨ IS-A	⊑ COMPONENT ID SET DATA TYPE
Field 2		
⊑∨ RELATIONSHIP ORIGIN	⊑∨ IS-A	⊑ COMPONENT ID SET DATA TYPE

Since: Inception

id: `StatedNavigationPattern` | UUID: `d02957d6-132d-5b3c-adba-505f5778d998`

⊑ Stated pattern for logic coordinate

The pattern an author's own stated axioms are recorded under, for a Logic Coordinate.

```
⊑ ImmutableCoordinate Properties
```

Parents: ⊑ IMMUTABLECOORDINATE PROPERTIES

Since: Inception

id: `StatedPatternForLogicCoordinate` | UUID: `cfd2a47e-8169-5e71-9122-d5b73efd990a`

⊑ Stated premise type

The axiom origin marking a definition as authored: the expression an editor asserted, before any classification.

```
⊑ Axiom origin
```

Parents: ⊑ AXIOM ORIGIN

Since: Inception

id: `StatedPremiseType` | UUID: `3b0dbd3b-2e53-3a30-8576-6c7fa7773060`

⊑∨ Status field

A concept field whose value is the status dimension of a STAMP — the version's state (active, inactive, or otherwise) at the time it was committed.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `StatusField` | UUID: `f2c79ebb-3095-44ea-831f-992aed48801f`

⊑∨ Status for version

Why a version's status field is recorded: to say whether the version is active, inactive, or withdrawn, so a view can accept or ignore it.

```
⊑ Version Properties ⊓ Purpose
```

Parents: ⊑∨ VERSION PROPERTIES ⊑ PURPOSE

Since: Inception

id: `StatusForVersion` | UUID: `0608e233-d79d-5076-985b-9b1ea4e14b4c`

⊑∨ Status value

The status of the STAMP Coordinate(Active, Cancelled, Inactive, Primordial)

```
⊑ STAMP dimensions ⊓ Meaning
```

Parents: ⊑ STAMP DIMENSIONS ⊑ MEANING

Children: ⊑ PRIMORDIAL STATE ⊑ INACTIVE STATE ⊑ ACTIVE STATE ⊑ CANCELED STATE ⊑ WITHDRAWN STATE

Since: Inception

id: `StatusValue` | UUID: `10b873e2-8247-5ab5-9dec-4edef37fc219`

⊑ String data type

A sequence of characters, either as a literal constant or as a variable. Strings could be used to represent terms from code systems or URLs, textual definitions, etc.

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `StringDataType` | UUID: `a46aaf11-b37a-32d6-abdc-707f084ec8f5`

⊑∨ String default

The String field's loud default: the literal "UNINITIALIZED" — a value that works as an initial value but reads as obviously needing revision, never Java's silent empty string.

```
⊑ Defaults and templates model ⊓ Meaning
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL ⊑ MEANING

Since: Inception

id: `StringDefault` | UUID: `60fb29ec-e2ad-55a2-bf03-9d1eb35786ba`

⊑ String field

The field kind for string-valued fields: a pattern field holding text.

```
⊑ Field definition data type field
```

Parents: ⊑ FIELD DEFINITION DATA TYPE FIELD

Children: ⊑∨ PUBLIC ID FIELD

Since: Inception

id: `StringField` | UUID: `27d3905b-b19a-41ff-bed1-fc55f49f8ce4`

⊑∨ Subject of Commentary

The referenced-component role a Comment Pattern semantic's attachment target plays: the subject free-text commentary is about — distinct from Comment, which names the commentary's own content type, not the component being commented on.

```
⊑ Editorial model ⊓ Meaning
```

Parents: ⊑ EDITORIAL MODEL ⊑ MEANING

Since: Inception

id: `SubjectOfCommentary` | UUID: `53e639c4-723c-5bef-a86e-edeaecaa1cf2`

⊑ Sufficient concept definition

The definition status marking a concept as sufficiently defined: its stated conditions are enough to classify an instance as the concept, so a reasoner can place it by definition alone.

```
⊑ Sufficient concept definition operator
```

Parents: ⊑ SUFFICIENT CONCEPT DEFINITION OPERATOR

Since: Inception

id: `SufficientConceptDefinition` | UUID: `6d9cd46e-8a8f-310a-a298-3e55dcf7a986`

⊑ Sufficient concept definition operator

Groups the operators marking a definition as sufficient: conditions enough to classify anything meeting them as the concept, not merely necessary of it.

```
⊑ Meaning
```

Parents: ⊑ MEANING

Children: ⊑ SUFFICIENT CONCEPT DEFINITION

Since: Inception

id: `SufficientConceptDefinitionOperator` | UUID: `dfa80f36-dbe6-5006-8509-c497a26ceab5`

⊑ Sufficient set

A set of relationships that differentiate a concept and its subtypes from all other concepts. A concept that contains at least one set of necessary and sufficient conditions is considered defined.

⊑ Logical set

Parents: ⊑ LOGICAL SET

Since: Inception

id: `SufficientSet` | UUID: `8aa48cfd-485b-5140-beb9-0d122f7812d9`

⊑ Swedish language

The Swedish language, as a value for a description's language field: a language coordinate that names it selects Swedish-language descriptions for rendering.

⊑ Language

Parents: ⊑ ∨ LANGUAGE

Since: Inception

id: `SwedishLanguage` | UUID: `9784a791-8fdb-32f7-88da-74ab135fe4e3`

⊑ Taxonomy field constraint kind

The kind of taxonomy-relative rule a Taxonomy Field Constraint Pattern semantic expresses: which concepts are legal values for the constrained pattern field, derived from the taxonomy under the checking view.

⊑ Constraint model

Parents: ⊑ CONSTRAINT MODEL

Children: ⊑ KIND-OF FIELD CONSTRAINT ⊑ DESCENDANT FIELD CONSTRAINT ⊑ IMMEDIATE CHILD FIELD CONSTRAINT ⊑ LEAF DESCENDANT FIELD CONSTRAINT

Since: Inception

id: `TaxonomyFieldConstraintKind` | UUID: `3bb160c7-cf0f-5937-824a-88fb819bd3f3`

P Taxonomy Field Constraint Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		

Meaning	Purpose	Data type
⊑∨ Constrained Pattern	⊑∨ Field Value Restriction	—
e.g. ***P*** *Description Pattern*		
Field 1		
⊑∨ Constrained field	⊑∨ Constraint Scope	⊑ Component data type
e.g. ⊑∨ *Language for description*		
Field 2		
⊑∨ Constraint kind	⊑∨ Constraint Rule	⊑ Component data type
e.g. ⊑ *Kind-of field constraint*		
Field 3		
⊑∨ Constraint anchor concept	⊑∨ Taxonomy Reference Point	⊑ Component data type
e.g. ⊑∨ *Language*		

Since: Inception

id: `TaxonomyFieldConstraintPattern` | UUID: `3a6f5ed0-730a-57a9-8575-8d0efb816149`

⊑∨ Taxonomy Navigation Cache

Why a navigation pattern's referenced component carries an IsA semantic: a pre-computed parent/child structure so traversal doesn't require re-deriving from axioms every time.

```
⊑ Navigation ⊓ Purpose
```

Parents: ⊑ Navigation ⊑ Purpose

Since: Inception

id: `TaxonomyNavigationCache` | UUID: `6d84de25-0aee-52bc-83bb-9e2fa647b442`

⊑ Taxonomy operator

An operator or set of operations applied within the context of a taxonomy

```
⊑ Meaning
```

Parents: ⊑ MEANING

Children: ⊑ LOGICALLY EQUIVALENT TO ⊑ PART OF

Since: Inception

id: `TaxonomyOperator` | UUID: `e9252365-7a43-57ea-bf94-3f23bab4ef99`

⊑∨ Taxonomy Reference Point

Why the Constraint anchor concept value is recorded: to give a kind-of, descendant, leaf-descendant, or immediate-child rule something concrete to measure against.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ CONSTRAINT MODEL ⊑ PURPOSE

Since: Inception

id: `TaxonomyReferencePoint` | UUID: `80a0d6b3-1a18-5afe-9155-75084beb06e7`

⊑ Template concept

Parent of the per-purpose template attachment concepts: each child is minted for one template purpose, and a template semantic of a pattern for that purpose is a complete semantic of that pattern whose referenced component is that child — starting content offered when authoring new components. As with a default value semantic, attachment there is a support declaration, never a domain assertion, and no child joins a domain member set.

```
⊑ Defaults and templates model
```

Parents: ⊑ DEFAULTS AND TEMPLATES MODEL

Since: Inception

id: `TemplateConcept` | UUID: `663fb8eb-8035-5e22-b423-9fca85a3f11c`

⊑ Temporal Axiom

Groups axioms about time: the temporal counterpart of the interval-axiom vocabulary.

```
⊑ Axioms
```

Parents: ⊑ AXIOMS

Since: Inception

id: `TemporalAxiom` | UUID: `5144d836-18d8-4881-a377-2d4640b710a9`

⊑ Text comparison measure semantic

Text comparison with a focus on semantic meaning involves evaluating the similarity or relatedness between pieces of text based on their underlying meaning rather than just their surface structure.

```
⊑ Meaning
```

Parents: ⊑ MEANING

Children: ⊑ CASE INSENSITIVE EVALUATION ⊑ CASE SENSITIVE EVALUATION

Since: Inception

id: `TextComparisonMeasureSemantic` | UUID: `b1531e68-4e7a-5194-b1f9-9aaace269372`

⊑∨ Text for description

Captures the human readable text for a description in Komet

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `TextForDescription` | UUID: `8bdcbe5d-e92e-5c10-845e-b585e6061672`

⊑∨ Time

A point in time, as a general concept — the meaning counterpart to Status value/Author/Module/Path: what kind of value a STAMP's time field holds, distinct from TimeForVersion, which names why it is recorded.

```
⊑ STAMP dimensions ⊓ Meaning
```

Parents: ⊑ STAMP DIMENSIONS ⊑ MEANING

Since: Inception

id: `Time` | UUID: `13557c25-add3-50fe-aeb1-edb52befcd40`

⊑∨ Time field

An instant field whose value is the time dimension of a STAMP — when the version was committed.

```
⊑ Instant field ⊓ Meaning
```

Parents: ⊑ INSTANT FIELD ⊑ MEANING

Since: Inception

id: `TimeField` | UUID: `15293325-c16b-4f2e-8109-5b22b3355bcd`

⊑∨ Time for version

Why a version's time field is recorded: to say when the version was committed, so a view positioned at a moment can resolve what was current then.

```
⊑ Version Properties ⊓ Purpose
```

Parents: ⊑∨ VERSION PROPERTIES ⊑ PURPOSE

Since: Inception

id: `TimeForVersion` | UUID: `a9b0dfb2-f463-5dae-8ba8-7f2e8385571b`

P Tinkar base model component pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ STARTER DATA AUTHORING	⊑∨ SET MEMBERSHIP	—
e.g. ⊑ UNINITIALIZED COMPONENT		

Since: Inception

id: `TinkarBaseModelComponentPattern` | UUID: `6070f6f5-893d-5144-adce-7d305c391cf9`

⊑ Tinkar Model concept

Root of Tinkar's own meta-schema terminology: the concepts Tinkar uses to describe its data model itself — components, descriptions, dialects, fields, and axioms — rather than any modeled domain.

```
⊑ Model concept
```

Parents: ⊑ MODEL CONCEPT

Children: ⊑ Concept type ⊑ Description list for concept ⊑∨ Is-a
⊑∨ Description acceptability ⊑∨ Relationship destination
⊑∨ Concrete value operator ⊑∨ Language ⊑ Concept details tree table
⊑∨ Description case significance ⊑ Display Fields ⊑∨ Stated Definition
⊑∨ Identifier Source ⊑ Description-logic profile ⊑ Component type focus
⊑∨ Inferred Definition ⊑∨ Relationship origin ⊑ Blank Concept
⊑∨ Description type ⊑∨ Reference Range ⊑ Field categories ⊑ Concept version
⊑∨ Description semantic ⊑∨ Identifier Value ⊑ Grouping ⊑∨ Description
⊑ Axioms ⊑∨ Logical Definition ⊑∨ Value Constraint Source ⊑ Dialect
⊑∨ Value Constraint ⊑∨ Text for description

Since: Inception

id: `TinkarModelConcept` | UUID: `bc59d656-83d3-47d8-9507-0e656ea95463`

⊑ Tinkar Starter Data Author

The author identity the upstream starter data was committed under, retained for the historic versions consumers already hold; this set's own content is authored by IKE Community.

```
⊑ Author
```

Parents: ⊑∨ Author

Since: Inception

id: `TinkarStarterDataAuthor` | UUID: `dd96b2ea-6d7b-3791-ad74-bbdc67c493c1`

⊑ Transitive Feature

Marks a property as transitive: if it relates the first to the second and the second to the third, it relates the first to the third.

```
⊑ Logical expression model
```

Parents: ⊑ Logical expression model

Since: Inception

id: `TransitiveFeature` | UUID: `53f866d0-fd61-5c85-a16c-150bd619a0ac`

⊑ Tree

A graph in which every vertex except the root has exactly one predecessor, so every vertex is reached by exactly one path from the root. The platform's DiTree carrier holds definition expressions this way; the general DiGraph carrier relaxes the single-predecessor restriction.

⊑ Graph

Parents: ⊑ GRAPH

Children: ⊑ EL++ DITREE

Since: Inception

id: Tree | UUID: 5a233bd9-2650-5217-bc3e-5d01a5ad50ce

⊑ Tree amalgam properties

Data structure that consists of nodes connected by edges (a mixture or blend of different elements)

⊑ Legacy model concepts

Parents: ⊑ LEGACY MODEL CONCEPTS

Children: ⊑ INVERSE TREE LIST ⊑ TREE LIST

Since: Inception

id: TreeAmalgamProperties | UUID: d6151a47-4610-5a5c-abd0-42c82be9b633

⊑ Tree list

The legacy tree-amalgam field holding a navigation tree's parent-to-child adjacency, paired with Inverse tree list.

⊑ Tree amalgam properties

Parents: ⊑ TREE AMALGAM PROPERTIES

Since: Inception

id: TreeList | UUID: c11dd7a1-0ba1-5378-81d6-3efdba1e074b

⊑ Typed atom

An atom that carries a type concept naming which relation or feature it asserts: a role (existential restriction over a role type), an interval role (a role bounded to an interval), or a feature (a concrete-domain comparison against a feature type). Mirrors LogicalAxiom.Atom.TypedAtom in the sealed hierarchy.

⊑ Atom

Parents: ⊑ Atom

Children: ⊑ Role ⊑ Feature ⊑ Interval role

Since: Inception

id: `TypedAtom` | UUID: `62538a4c-cc26-514a-a3ef-a59c66c2bc42`

⊑ Uncategorized phenomenon

The domain placeholder for phenomena not yet classified into the model: content admitted before its category is settled.

```
⊑ Phenomenon
```

Parents: ⊑ Phenomenon

Since: Inception

id: `UncategorizedPhenomenon` | UUID: `722f5ac8-1f5c-5d8f-96bb-370d79596f66`

⊑ Uninitialized Component

The platform's null-object component: the value a component reference carries when it has never been set. The machinery produces it -- the nonexistent-stamp sentinel names it as both module and path, and path resolution guards on it -- and rendering surfaces show nothing for it, treating it like an unknown identifier. Distinct from Blank Concept, which an editor writes into a field a person deliberately cleared.

```
⊑ Chronicle and version model
```

Parents: ⊑ Chronicle and version model

Since: Inception

id: `UninitializedComponent` | UUID: `55f74246-0a25-57ac-9473-a788d08fb656`

⊑∨ Uniquely identify knowledge graph components

The purpose of this externally valid identifier—based on a list of UUIDs—is to provide a globally unique reference to each entity that is reliable across different systems and databases. This ensures consistent identification, serialization, and lookup of entities regardless of their storage location or implementation details.

```
⊑ Semantic properties ⊓ Purpose
```

Parents: ⊑ Semantic properties ⊑ Purpose

Since: Inception

id: `UniquelyIdentifyKnowledgeGraphComponents` | UUID: `dde9a93d-250c-449b-bea0-ba1133d1387b`

⊑∨ Unit Example

Why example UCUM units are recorded on a value constraint: to show a representative unit of measure for the constrained value, for a reader's reference.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ Constraint model ⊑ Purpose

Since: Inception

id: `UnitExample` | UUID: `bfc8c53c-4eee-58d4-a73f-f7d56bd6e04c`

⊑ Unit of Measure

The field naming the unit an interval's bounds are expressed in.

```
⊑ Interval Set Axioms
```

Parents: ⊑ Interval Set Axioms

Since: Inception

id: `UnitOfMeasure` | UUID: `40afdda5-89d6-4b80-8181-1ddd6eb92dc8`

⊑∨ United States of America English dialect

The United States English dialect dimension: the anchor for dialect semantics recording whether a description is preferred or acceptable in US usage.

```
⊑ English Dialect ⊓ Meaning
```

Parents: ⊑ English Dialect ⊑ Meaning

Children: ⊑ US Nursing dialect

Since: Inception

id: `UnitedStatesOfAmericaEnglishDialect` | UUID: `bca0a686-3516-3daf-8fcf-fe396d13cfad`

⊑ Universal Restriction

Universal restrictions constrain the relationships along a given property to concepts that are members of a specific class.

⊑ Role operator

Parents: ⊑  Role operator

Since: Inception

id: `UniversalRestriction` | UUID: `fc18c082-c6ad-52d2-b568-cc9568ace6c9`

⊑  Universally unique identifier

A universally unique identifier that uniquely represents a concept in Tinkar

⊑ Identifier Source

Parents: ⊑ ∨  Identifier Source

Since: Inception

id: `UniversallyUniqueIdentifier` | UUID: `845274b5-9644-3799-94c6-e0ea37e7d1a4`

⊑  Unmodeled role concept

An ISAAC-era marker for role concepts awaiting modeling, retired to Legacy.

⊑ Legacy model concepts

Parents: ⊑  Legacy model concepts

Since: Inception

id: `UnmodeledRoleConcept` | UUID: `4be7118f-e6ab-5dc7-bcba-b2cc8b028492`

⊑  Upper Bound Open

The marker that an interval's upper bound is open: the bound value itself lies outside the interval.

⊑ Interval Set Axioms

Parents: ⊑  Interval Set Axioms

Since: Inception

id: `UpperBoundOpen` | UUID: `c20b3b1e-112f-4cb2-b901-4046db844629`

P US Dialect Pattern

Records whether a description is preferred or acceptable in the United States English dialect — one field, that description's acceptability for this dialect.

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Description acceptability	⊑∨ Description semantic	—
e.g. Uninitialized Component (SOLOR)		
Field 1		
⊑∨ United States of America English dialect	⊑∨ Description acceptability	⊑ Component data
e.g. ⊑ *Preferred*		

Since: Inception

id: `USDialectPattern` | UUID: `08f9112c-c041-56d3-b89b-63258f070074`

⊑ US Nursing dialect

The US nursing dialect: a specialization of United States English for nursing practice, where preferred terms differ from general US usage.

```
⊑ United States of America English dialect
```

Parents: ⊑∨ United States of America English dialect

Since: Inception

id: `USNursingDialect` | UUID: `6e447636-1085-32ff-bc36-6748a45255de`

⊑ Users module

The module user-authored content lands in, keeping personal editing separate from released content.

```
⊑ Module
```

Parents: ⊑∨ Module

Since: Inception

id: `UsersModule` | UUID: `349161ba-9a6a-5c9c-a78f-281f19cfc057`

⊑ UUID data type

Distinction of data type of UUID

```
⊑ Dynamic column data types
```

Parents: ⊑ DYNAMIC COLUMN DATA TYPES

Since: Inception

id: `UUIDDataType` | UUID: `2faa9262-8fb2-11db-b606-0800200c9a66`

⊑ UUID display field

The data type for fields holding a universally unique identifier value.

```
⊑ Display Fields
```

Parents: ⊑ DISPLAY FIELDS

Since: Inception

id: `UUIDDisplayField` | UUID: `dea8cb0f-9bb5-56bb-af27-a14943cb24ba`

⊑ UUID list for component

The complete list of universally unique identifiers in a component's public identity; merging two components unions these lists.

```
⊑ Chronicle properties
```

Parents: ⊑ CHRONICLE PROPERTIES

Since: Inception

id: `UUIDListForComponent` | UUID: `f8e3238e-7424-5a40-8649-a8d164382fec`

⊑∨ Value Constraint

A component has specific value requirements that needs to be met

```
⊑ Tinkar Model concept ⊓ Meaning
```

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `ValueConstraint` | UUID: `8c55fb86-92d8-42a9-ad70-1e23abbf7eec`

P Value Constraint Pattern

A pattern specifying value constraint pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ Value Constraint	⊑∨ Numeric Value Restriction	—
Field 1		
⊑∨ Value Constraint Source	⊑∨ Reference Range Authority	⊑ Concept data type
Field 2		
⊑∨ Minimum Value Operator	⊑∨ Concrete value operator	⊑ Concept data type
Field 3		
⊑∨ Reference Range Minimum	⊑∨ Reference Range	⊑ Float data type
Field 4		
⊑∨ Maximum Value Operator	⊑∨ Concrete value operator	⊑ Component data type
Field 5		
⊑∨ Reference Range Maximum	⊑∨ Reference Range	⊑ Float data type
Field 6		
⊑∨ Example UCUM Units	⊑∨ Unit Example	⊑ String data type

Since: Inception

id: `ValueConstraintPattern` | UUID: `922697f7-36ba-4afc-9dd5-f29d54b0fdec`

⊑∨ Value Constraint Source

The source organization of that specifies the constraint

⊑ `Tinkar Model concept` ⊓ `Meaning`

Parents: ⊑ TINKAR MODEL CONCEPT ⊑ MEANING

Since: Inception

id: `ValueConstraintSource` | UUID: `09aa031a-6290-4ec9-a44c-23928a767da3`

⊑∨ Value Disambiguation

Why the Value-set field value is recorded: to say which Model Feature of the value-set pattern actually holds the member value, when that pattern carries other fields too, such as a sort order.

```
⊑ Constraint model ⊓ Purpose
```

Parents: ⊑ CONSTRAINT MODEL ⊑ PURPOSE

Since: Inception

id: `ValueDisambiguation` | UUID: `c2788745-e921-5499-a133-8709624e6c6b`

⊑∨ Value-set field

For a value-set constraint, the Model Feature of the value-set pattern that holds the member value, named by its meaning concept: a declared field's meaning, or the value-set pattern's referenced-component meaning when the members are the referenced components themselves (a membership pattern) — disambiguating when that pattern carries other fields, such as a sort order.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `ValueSetField` | UUID: `313f88e6-fe6f-5086-9a0f-9822846143b6`

P Value-set Field Constraint Pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ CONSTRAINED PATTERN	⊑∨ FIELD VALUE RESTRICTION	—
e.g. ***P*** *PREFERRED REVIEWER PATTERN*		

Meaning	Purpose	Data type
Field 1		
⊑∨ CONSTRAINED FIELD	⊑∨ CONSTRAINT SCOPE	⊑ COMPONENT DATA TYPE
e.g. ⊑∨ *PREFERRED REVIEWER*		
Field 2		
⊑∨ VALUE-SET PATTERN	⊑∨ LEGAL VALUE SOURCE	⊑ COMPONENT DATA TYPE
e.g. **P** *STARTER SET AUTHOR ROSTER PATTERN*		
Field 3		
⊑∨ VALUE-SET FIELD	⊑∨ VALUE DISAMBIGUATION	⊑ COMPONENT DATA TYPE
e.g. ⊑∨ *ROSTER AUTHOR*		
Field 4		
⊑∨ MEMBER MATCH RELATION	⊑∨ MATCH RULE	⊑ COMPONENT DATA TYPE
e.g. ⊑ *EQUAL MATCH RELATION*		

Since: Inception

id: `ValueSetFieldConstraintPattern` | UUID: `86ca7486-15ec-52d0-9823-298a5a29314a`

⊑∨ Value-set pattern

For a value-set constraint, the pattern whose active semantics enumerate the legal values.

```
⊑ Concept field ⊓ Meaning
```

Parents: ⊑∨ CONCEPT FIELD ⊑ MEANING

Since: Inception

id: `ValueSetPattern` | UUID: `af79a5f0-6505-5f84-a4ec-1ffd119b401f`

P Version control path pattern

Pattern shape

Meaning	Purpose	Data type
Referenced component — the component (concept, semantic, pattern, or STAMP) this pattern's semantics attach to		
⊑∨ PATH	⊑∨ SET MEMBERSHIP	—
e.g. ⊑ *MASTER PATH*		

Since: Inception

id: `VersionControlPathPattern` | UUID: `add1db57-72fe-53c8-a528-1614bda20ec6`

⊑∨ Version History

Why a Semantic Chronology Pattern's Semantic versions set is recorded: to hold every version of this semantic, in order — the semantic-level counterpart to the versions-field/set pair every other chronicle-shape pattern already names distinctly.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ CHRONICLE AND VERSION MODEL ⊑ PURPOSE

Since: Inception

id: `VersionHistory` | UUID: `658df6b1-3647-5569-a252-12e5e3d9ab09`

⊑ Version list for chronicle

The field holding a chronicle's complete version list, in commit order.

```
⊑ Chronicle properties
```

Parents: ⊑ CHRONICLE PROPERTIES

Since: Inception

id: `VersionListForChronicle` | UUID: `d6f27f80-8e20-58fe-8d69-66ad4644f969`

⊑∨ Version Properties

What a STAMP pattern semantic means: the five-dimension provenance tuple of a version — status, time, author, module, and path together, as one value; the properties every version carries, rather than any single dimension alone.

```
⊑ Chronicle and version model ⊓ Meaning
```

Parents: ⊑ Chronicle and version model ⊑ Meaning

Children: ⊑∨ Status for version ⊑ Description version properties
⊑∨ Author for version ⊑∨ Time for version ⊑∨ Path for version
⊑∨ Module for version

Since: Inception

id: `VersionProperties` | UUID: `93f844df-38e5-5167-ba94-2c948b8bd07c`

⊑∨ Version Provenance

Why a version-shape pattern's own STAMP field is recorded: to say who committed this version, in what state, module, path, and when.

```
⊑ Provenance model ⊓ Purpose
```

Parents: ⊑ Provenance model ⊑ Purpose

Since: Inception

id: `VersionProvenance` | UUID: `2c40b894-236c-589c-9528-3cebd6eca49c`

⊑∨ Version Snapshot

Why a version-shape pattern's referenced component carries this semantic: to record one point-in-time state of a chronicle — the STAMP it was committed with, plus whatever field values were true then.

```
⊑ Chronicle and version model ⊓ Purpose
```

Parents: ⊑ Chronicle and version model ⊑ Purpose

Since: Inception

id: `VersionSnapshot` | UUID: `53741119-2eb8-5c07-86a8-927efa563f5b`

⊑ Vertex

A node of a graph: it carries a meaning — the concept naming what this vertex asserts or denotes — and zero or more properties, each a concept-keyed value. Edges are recorded as vertex adjacency, so the vertex is the unit of both content and connection.

```
⊑ Graph model
```

Parents: ⊑ Graph model

Children: ⊑ Navigation vertex

Since: Inception

id: `Vertex` | UUID: `2185c628-0fb7-588a-8c39-76be9da2fb95`

⊑ Vertex display field

The data type for fields holding a single graph vertex.

⊑ Display Fields

Parents: ⊑ Display Fields

Since: Inception

id: `VertexDisplayField` | UUID: `3e56c6b6-5371-11eb-ae93-0242ac130002`

⊑ Vertex meaning

The concept a vertex denotes: what this node of the graph asserts, names, or stands for. Every vertex carries exactly one meaning; processing dispatches on it — a definition root, an and-connective, a role — the way code dispatches on a sealed type.

⊑ Graph model

Parents: ⊑ Graph model

Since: Inception

id: `VertexMeaning` | UUID: `b1b1afe0-7789-54da-87d1-d029ef997fd4`

⊑ Vertex property

A concept-keyed value carried on a vertex: the key is a concept naming what the value is (a role type, an operator, a literal), and the value is any field data type. Properties are how a vertex carries the particulars its meaning calls for.

⊑ Graph model

Parents: ⊑ Graph model

Since: Inception

id: `VertexProperty` | UUID: `8252c165-c7ed-50f4-9799-3340e7194a3a`

⊑ Vertex sort

The navigation-coordinate dimension naming the sort applied to a vertex's children when a navigator renders them.

⊑ ImmutableCoordinate Properties

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `VertexSort` | UUID: `e973f077-a99d-59e6-b7bd-804e87e0e639`

⊑ Vertex state set

The navigation-coordinate dimension holding the states a component must be in to appear in navigation; active-only in the default view.

⊑ ImmutableCoordinate Properties

Parents: ⊑ ImmutableCoordinate Properties

Since: Inception

id: `VertexStateSet` | UUID: `347cd3f2-8130-5032-8960-091e194e9afe`

⊑ View coordinate key

A query-clause key identifying which view coordinate a stored query executes under, so queries reference a view without embedding one.

⊑ Query clauses

Parents: ⊑ Query clauses

Since: Inception

id: `ViewCoordinateKey` | UUID: `4211cf36-bd75-586a-805c-51f059e2eaaa`

⊑ View coordinate model

The meta-schema of views: the coordinate property families a rendering or editing surface assembles into a reproducible way of looking — stamp, language, logic, and path coordinate properties, and the immutable-coordinate property vocabulary. A coordinate's constituents are dimensions; these concepts name them.

⊑ Model concept

Parents: ⊑ Model concept

Children: ⊑ ImmutableCoordinate Properties ⊑ Order for concept attachments ⊑ Path for user ⊑ Language coordinate properties ⊑ Module for user ⊑ Logic coordinate properties ⊑ Path coordinate properties ⊑ Order for axiom attachments ⊑ Order for description attachments

Since: Inception

id: `ViewCoordinateModel` | UUID: `f83c63a3-6d1d-55bf-a0ab-759079023592`

⊑ Withdrawn state

Concept used to represent a status for components that are withdrawn.

```
⊑ Status value
```

Parents: ⊑ ∨ Status value

Since: Inception

id: `WithdrawnState` | UUID: `35fd4750-6e43-5fa3-ba7f-f2ad376052bc`

Last updated 2026-07-24 15:19:24 -0700

Colophon

This document was produced using the IKE Documentation Pipeline.

Rendering pipeline	AsciiDoc → HTML5 → PDF (Prince XML)
AsciiDoc backend	html5
Document date	2026-07-24
Build timestamp	2026-07-25T11:05:54.470812-07:00