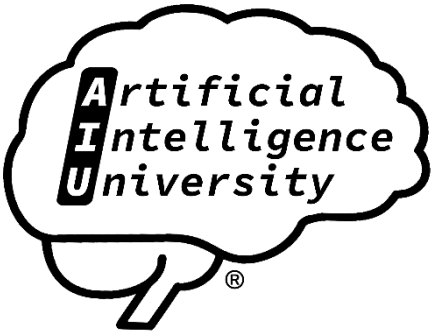


Causalontology:
THE SPEC

D. R. Dison



Causalontology:

THE SPEC

D. R. Dison

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

TABLE OF CONTENTS:

CAUSALONTOLOGY: THE STANDALONE STANDARD AND THE SHARED COMMONS..... 5

A NOTE ON THIS EDITION..... 6

CAUSALONTOLOGY'S CONNECTION TO AGI (ARTIFICIAL GENERAL INTELLIGENCE)..... 6

HOW CAUSALONTOLOGY ENABLES AGI (ARTIFICIAL GENERAL INTELLIGENCE)..... 12

BRANCH NOTE..... 19

PART 1. THE THESIS: A STANDARD AND A COMMONS, NOT A PROGRAM..... 20

THE CLAIM, FROM FIVE ANGLES: A WORLD-FIRST SYNTHESIS..... 22

THE SEVEN GOVERNING DESIGN PRINCIPLES..... 30

PART 2. THE STANDALONE REPOSITORY: SHIP THE SPECIFICATION, NOT THE CODE..... 32

PART 3. THE OBJECT MODEL: TWENTY-ONE KINDS OF THING, AND THE SCOPE OF THE STANDARD..... 35

PART 4. IDENTITY: CONTENT-ADDRESSED IDENTIFIERS, CANONICALIZATION, AND MERGE..... 49

PART 5. CROSS-LANGUAGE INTEROPERABILITY: THROUGH DATA, NOT THROUGH CODE..... 55

PART 6. THE SCHEMAS (THE MACHINE-CHECKABLE STRUCTURE, IN FULL)..... 57

PART 7. THE SEMANTICS (THE RULES BEYOND THE SCHEMAS)..... 105

PART 8. PROVENANCE, SIGNATURES, ENRICHMENT, RETRACTION, AND TRUST..... 113

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

PART 9. THE SHARED GLOBAL STORE: OPERATIONS, PROTOCOL, RESOLVE, AND TIERS..... 117

PART 10. STIGMERGY: THE STORE GUIDES THE NEXT CONTRIBUTION..... 123

PART 11. SAFETY: ABUSE RESISTANCE, CLAIMS OF CONSEQUENCE, TAKEDOWN BY TIER..... 126

PART 12. LICENSING: CODE, SPECIFICATION, AND DATA 129

PART 13. GOVERNANCE AND VERSIONING..... 130

PART 14. THE CONFORMANCE SUITE (ONE HUNDRED AND THIRTY-SEVEN VECTORS)..... 132

PART 15. QUICKSTART: ONE CLAIM, FOUR LANGUAGES, END TO END..... 160

PART 16. THE MINIMUM VIABLE PRODUCT AND THE ROADMAP..... 163

PART 17. LOCKED DECISIONS..... 176

PART 18. ONE-PAGER CHEAT SHEET..... 182

PART 21. COMMONS STORAGE AND FEDERATION DESIGN 188

GLOSSARY OF ACRONYMS AND TERMS..... 210

CONCLUSION..... 219

PART 22. THE CROSS-STRATAL SEAM, THE ORDINAL TICK, AND THE CONDUIT'S REALIZED_BY..... 220

APPENDIX: VERSION HISTORY..... 247

BACK COVER MATTER..... 255

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

CAUSALONTOLOGY: THE STANDALONE STANDARD AND THE SHARED COMMONS

A Programming-Language-Neutral Specification, a Conformance Test Suite, a Canonical Serialization, and a Content-Addressed, Provenance-First Global Data Store - with the Store's Own Partial Structures Serving as the Stigmergic Invitation to the Next Contribution

Standalone Branch, Version 27
rooted in Causalontology v9

Author: D. R. Dison, Founder of AIU (Artificial Intelligence University), Creator and Owner of PrologAI and Mentova.

Open Researcher and Contributor Identifier (ORCID): 0009-0001-9246-5758.

(https://www.linkedin.com/in/d-r-dison/)

Home: AI University (AIU) - github.com/ai-university-aiu

Note on acronyms: every acronym is spelled out in full the first time it appears, with the short form in parentheses; the short form is used thereafter.

A full glossary appears at the end.

Note on names: in prose the system is the proper noun "Causalontology"; in every machine context (the repository, package names, directory names, and Uniform Resource Identifiers) it is the lowercase identifier "causalontology".

© 2026-01-01 All Rights Reserved.

Both casings name one thing, each in its proper register.

A NOTE ON THIS EDITION

This document is prepared as part of the book named "Causalontology: AGI's Missing Link".

Because the book is written for every reader, this document explains each concept at a foundational level, as if the reader were meeting it for the first time.

Every acronym is spelled out the first time it appears, and every technical term is defined where it first occurs, with a full glossary at the end.

CAUSALONTOLOGY'S CONNECTION TO AGI (ARTIFICIAL GENERAL INTELLIGENCE)

Artificial General Intelligence (AGI) means machine intelligence with broad, human-level competence - a system that can walk into a situation it has never seen, figure out how the situation works, decide what to do, and carry what it learned into the next situation.

That is different from the narrow Artificial Intelligence (AI) we have today, which tends to be excellent at the specific things it was trained on and brittle everywhere else.

The claim of this book, stated plainly, is that the component most conspicuously missing from today's AI is an explicit, learnable, inspectable model of cause and effect, acquired by acting on

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

the world - and that Causalontology is a concrete, working proposal for exactly that component.

Here are the reasons for the connection, each one explained.

• The ladder of causation says today's AI is standing on the bottom rung.

The scientist Judea Pearl describes three levels of causal ability, often called the Ladder of Causation.

Rung one is ASSOCIATION - seeing that two things tend to occur together, the way umbrellas and wet streets go together.

Rung two is INTERVENTION - knowing what will happen if you DO something, the way flipping a switch turns on a light.

Rung three is COUNTERFACTUAL reasoning - imagining what WOULD HAVE happened if things had gone differently, the way a doctor asks whether the patient would have recovered without the medicine.

A "counterfactual" is a statement about a possibility that did not actually happen.

Systems that learn only from recorded data - including today's Large Language Models (LLMs), the text-predicting systems behind modern chat programs - are, by their mathematical construction, learners of association: rung one.

Human beings live on rungs two and three every day, and researchers across the field identify those upper rungs - intervention and

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

counterfactual thought - as prerequisites for general intelligence.

Causalontology is engineered for the upper rungs: its fundamental object records what an INTERVENTION brings about, its evidence grading ranks acting above watching, and its strength values are defined in the formal language of intervention (the do-calculus, defined later in this document).

• Research shows that talking about causes is not the same as reasoning about them.

A well-known line of research, titled "Causal Parrots", found that Large Language Models may TALK causality fluently while not actually BEING causal - they can repeat causal facts memorized from their training text, the way a parrot repeats words, without performing causal inference about anything new.

Repeating "smoking causes cancer" is rung-one recall.

Working out what a never-before-seen switch does, by pressing it, is rung-two reasoning - and that is the ability general intelligence requires.

Causalontology's entire learning story is built on the second kind: act, observe what follows, and record the relation as an inspectable object.

• The hardest current test of general intelligence points exactly here.

The Abstraction and Reasoning Corpus for Artificial General Intelligence, third generation

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

(ARC-AGI-3), is an interactive benchmark launched on March 25, 2026, in which an agent is dropped into an unfamiliar game with no instructions, no rules, and no stated goal.

To succeed, the agent must explore, discover the mechanics by acting, infer what winning means, build a model of the world, and plan efficiently.

Ordinary people solve these games essentially every time - human performance is reported at one hundred percent.

Frontier AI systems, at launch, scored below one percent.

That gap is the clearest public evidence available that something specific is missing from today's AI - and what the benchmark rewards is precisely explicit, interventional, hierarchical modeling of cause and effect, which is Causalontology's exact subject matter.

• A mathematical result says generality REQUIRES a causal model.

Recent theoretical work (titled "Robust agents learn causal world models") proves that any agent capable of adapting robustly across many environments must have learned, at least approximately, a causal model of its world - there is no route to robust generality that avoids causality.

In other words, the "G" in AGI - generality - is not merely helped by causal modeling; on this result, it cannot exist without it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Causalontology is a proposal for holding that unavoidable causal model in an explicit, shareable, inspectable form rather than an implicit and hidden one.

• Causal knowledge is what transfers, and transfer is what generality means.

Correlations - patterns of things occurring together - are fragile: they change when the situation changes.

Causal mechanisms are comparatively stable: the reason a switch lights a lamp is the same in a new house.

Research on causal representation learning argues exactly this - that representations organized around causal mechanisms are the ones that remain valid across changed conditions, which is why they transfer and generalize.

An intelligence built on transferable knowledge can face new situations with old understanding - which is the working definition of general intelligence.

• Children get to general intelligence the same way.

The developmental psychologist Jean Piaget showed that children construct their understanding of the world through interaction - touching, pushing, dropping, and watching what follows - a view called constructivism.

A child learns that a switch works by flipping it, not by staring at it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Causalontology's learning-by-intervention is the machine version of that childhood, and its refinement lineage (defined later in this document) lets rough early knowledge be enriched step by step, the way a child's understanding matures.

• Safety and trust require knowing WHY, not only WHAT.

A general intelligence will take actions that matter, so people must be able to ask why it acted and get a real answer.

A system whose knowledge is a web of inspectable cause-and-effect objects, each carrying its evidence and its author, can answer "why" all the way down - the glass-box property.

A system whose knowledge is hidden inside billions of numeric weights cannot.

Causal understanding is also what lets an agent foresee the consequences of an action before taking it, including harmful consequences - the foundation of machine prudence.

• No single mind can learn all the causes, which is why the commons matters.

Humanity's advantage was never one brilliant individual - it was accumulated, shared causal knowledge: fire, farming, medicine, engineering, each generation inheriting the causal discoveries of the last.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The shared, provenance-first Causalontology commons described in this document gives machine intelligence the same inheritance: a single, world-wide, ever-growing store of causal knowledge that every mind - human or synthetic - can draw from and add to.

General intelligence at civilization scale is a library, not a lone genius, and this document specifies that library.

HOW CAUSALONTOLOGY ENABLES AGI (ARTIFICIAL GENERAL INTELLIGENCE)

The previous section argued that explicit causal understanding is the missing link.

This section walks through the machinery, piece by piece, and shows how each piece supplies a capability that general intelligence needs.

• The Causal Relation Object turns "why" into something a machine can hold.

The fundamental unit of Causalontology is the Causal Relation Object (CRO): a cause-and-effect link made into a full object that carries what caused what, by what mechanism, with what timing, in what character, under what conditions.

Making causation a first-class object - the move called reification - is what lets a machine learn it, compare it, refine it, compose it, plan over it, and explain it.

You cannot reason carefully about something you cannot hold; the CRO is causation made holdable.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Timing as mechanism gives the power to rule things out.

Every CRO can carry a temporal window - how long after the cause the effect arrives.

That single feature enables a distinctly intelligent move: excluding a suspected cause purely on timing, the way a doctor asks "how long ago did you eat?" and clears the meal that was too recent or too old to be the culprit.

Diagnosis, troubleshooting, and scientific reasoning all depend on this move, and Causalontology builds it into the data itself.

• Modality gives nuance, and nuance gives safe action.

Each causal claim is graded as necessary, sufficient, contributory, or preventive - four different characters of influence.

An agent that knows a cause is merely contributory will not treat it as a guarantee; an agent that knows an action is preventive gains a hazard rule - a learned "do not do that".

Preventive relations are how a curious agent explores widely yet never repeats an action it has learned is harmful, which is a seed of machine safety.

• Learning by intervention delivers evidence that observation cannot.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Causalontology grades evidence, and the top grade is intervention: the source acted and watched what followed.

Acting breaks the ambiguity that haunts pure observation - when you flip the switch yourself, the light's response cannot be a coincidence of the data.

This grading, carried in every signed assertion, means the commons itself knows which of its claims rest on doing and which on watching, and consumers can weigh them accordingly.

• Hierarchy lets one mind reason at every scale.

A CRO's mechanism can be a whole sub-graph of finer CROs - the coarse claim "exercise improves health" decomposing into physiological steps, each decomposing further.

General intelligence requires exactly this ability to zoom: to plan at the level of goals, act at the level of muscles, and explain at whichever level the listener needs.

The hierarchy-consistency rule in this document keeps every level honest against the levels below it.

• Strength, confidence, and provenance make uncertainty honest.

Every assertion carries a strength (how strong the causal link is estimated to be), a confidence (how sure the source is), and a signed identity (who says so).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

An intelligence that can say "probably", "I am not sure", and "here is who told me" - and can revise when the evidence changes - is an intelligence that can operate in the real world, where certainty is rare.

The retraction record even gives every source an honest exit when it discovers its own error, and the history is never erased.

• Degenerate objects and refinement give a machine childhood.

A Causal Relation Object is valid with only its cause and effect filled in - a rough, partial piece of knowledge.

The refinement lineage then lets anyone - or any mind - enrich it later with timing, mechanism, and conditions, each enrichment linked to what it improved.

Knowledge in Causalontology therefore GROWS the way understanding grows in a person: rough first, refined with experience, never pretending to completeness it does not have.

This is constructivist learning, made into a data structure.

• The stigmergic commons turns the whole world into one growing mind.

Because partial objects, dangling references, contradictions, and undecomposed mechanisms are all first-class and queryable, the store itself continuously announces what is missing.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

"Stigmergy" is coordination through the shared environment - the way a half-built wall shows the next worker where the next brick goes.

Human contributors and autonomous minds alike can ask the commons "what is the most valuable gap near this topic?", fill it, and sign their work.

Knowledge then compounds - the mechanism by which a commons, like a civilization, becomes smarter than any of its members.

• The programming-language-neutral standard makes minds interoperable.

Because Causalontology is a specification with a conformance suite - not a program - any system in any programming language can speak it, and all of them read and write the same store.

An AGI ecosystem needs exactly this: many different minds, built by different people on different technology, sharing one language of cause and effect.

• The noun backbone and the hinge ground symbols in the world.

Causalontology's continuants (the nouns), occurrents (the verbs), and realizable entities (the dispositions, functions, and roles that join them) give a mind a principled way to connect words to things, things to their powers, and powers to the processes in which they show themselves.

This addresses one of the oldest problems in AI - the symbol grounding problem, the question of how

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

a symbol inside a machine comes to be about something in the world.

• The glass box is the alignment story.

Every claim traceable to a named rule; every contribution signed by its author; every retraction on the record; every consequence-laden claim served with its evidence showing.

A general intelligence built over such a substrate can be audited, corrected, and trusted in a way that an opaque system cannot - not because anyone promises good behavior, but because the structure makes reasons inspectable.

• And there is a working existence proof.

Causalontology is not only a paper design: its reference implementation runs today in PrologAI, a glass-box cognitive architecture, driving the synthetic mind Mentova.

That substrate has recorded a perfect score - four hundred out of four hundred - on the public training set of the first Abstraction and Reasoning Corpus benchmark (ARC-AGI-1), using pure symbolic induction with no Large Language Model, and holds a perfect score on the second benchmark's public set as well.

The interactive third benchmark, ARC-AGI-3, remains what this lineage of documents has always honestly called it: a labeled, falsifiable bet - the hypothesis that a mind whose world model is an explicit, reified, interventional causal graph is the right shape for the test that today's AI fails.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

An honest closing boundary: Causalontology is offered as a NECESSARY ingredient of general intelligence, not a sufficient one.

A complete mind also needs perception, memory, attention, motivation, and the other faculties - which is exactly what the PrologAI Branch of this work builds - and enabling AGI is not the same as guaranteeing it.

The claim of this book is the missing-link claim: without explicit, learnable, shareable causal understanding, general intelligence does not arrive; with it, the road is open.

Selected sources for these two sections, in plain text:

• Judea Pearl and Dana Mackenzie, "The Book of Why: The New Science of Cause and Effect", 2018 - the Ladder of Causation.

• Matej Zecevic, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting, "Causal Parrots: Large Language Models May Talk Causality But Are Not Causal", Transactions on Machine Learning Research, 2023.

• Bernhard Schoelkopf and colleagues, "Toward Causal Representation Learning", Proceedings of the Institute of Electrical and Electronics Engineers (IEEE), 2021 - causal invariance, robustness, and transfer.

• Jonathan Richens and Tom Everitt, "Robust agents learn causal world models", 2024 - the proof that

© 2026-01-01 All Rights Reserved.

robust generality requires an approximate causal model.

• Francois Chollet, "On the Measure of Intelligence", 2019 - intelligence as skill-acquisition efficiency in the face of novelty.

• The ARC Prize Foundation, arcprize.org - the ARC-AGI-3 interactive benchmark, launched March 25, 2026; human performance one hundred percent, frontier AI below one percent at launch.

• Gary Drescher, "Made-Up Minds: A Constructivist Approach to Artificial Intelligence", 1991 - learning causal schemas by acting.

• Jean Piaget's constructivism, throughout - understanding is built by interaction.

BRANCH NOTE

Causalontology forks into two branches sharing Causalontology version 9 as their common root:

• The PrologAI Branch (sibling document Causalontology_PrologAI_v1.txt).

Causalontology realized as the PrologAI cognitive-architecture packs - the reference implementation, written in SWI-Prolog, driving the Mentova synthetic mind.

• The Standalone Branch (this document).

Causalontology delivered in its purest, programming-language-neutral form, so anyone can

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

implement and use it in any programming language and contribute to one global commons.

The reframe that separates them, and the thesis of this whole document: Causalontology's purest form is a specification, not a program.

The PrologAI Branch is one implementation of that specification.

The Standalone Branch is the specification itself.

PART 1. THE THESIS: A STANDARD AND A COMMONS, NOT A PROGRAM

Causalontology began fused to one programming language, Prolog.

To free it so that it can be used everywhere, we separate two things:

• The data structure and its rules – the Causal Relation Object (CRO), the occurrent, the continuant, the realizable entity, and the signed provenance records about them, together with the rules that say what a valid instance is.

This is programming-language-neutral.

• Any one implementation of those rules – in Prolog, in Python, in Java, in Swift.

This is language-specific.

Once separated, PrologAI becomes the reference implementation – the first and canonical one, but

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

only one - and the pure Causalontology is a written specification that any language can implement.

Two familiar pictures, held side by side:

• Causalontology as a STANDARD.

Like a specification published by the World Wide Web Consortium (W3C) or the Internet Engineering Task Force (IETF): a normative document plus a conformance test suite, with independent implementations that are all correct because they all pass the same tests.

• Causalontology as a COMMONS.

Like Wikidata (the free, collaborative knowledge base that anyone can edit, where every statement carries its sources): one massive shared data store that developers everywhere draw from and contribute to.

Causalontology is that pattern, specialized to reified causation - causation made into first-class objects.

Causalontology is unusually well-architected for both roles: the Causal Relation Object admits partial and degenerate forms, every contribution of any kind is a signed record, and the whole design is glass-box - every claim, every alias, every retraction is inspectable and traceable to who or what contributed it.

In one phrase, the paradigm has a name: Causalontology is a VERB-FIRST NOUN-HOSTING ONTOLOGY.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Verb-first: the atomic building block is the occurrent – the happening – the fundamental unit reifies the master verb, causes, and even knowing is verb-first, because knowledge enters by acting (intervention) and by active watching – in which the viewer identifies with the watched actor and internally simulates the action, as when one learns a dance from an instructional video – and not by passive watching, through which little or no knowledge enters. This dovetails with the 2.0.0 evidence_type simulation: an internally simulated intervention has a first-class evidential slot, distinct from raw observation and from physical intervention.

Noun-hosting: things (continuants) are first-class citizens, understood through what they do and what happens to them – Causalontology governs the verbs and hosts the nouns, with realizable entities as the hinge between them.

Reality is what happens, and things are its participants.

THE CLAIM, FROM FIVE ANGLES: A WORLD-FIRST SYNTHESIS

Before the engineering, the identity claim.

Causalontology is described below from five angles.

Each angle states the same claim – that Causalontology is a first-of-its-kind synthesis – and then the terms in that claim are explained for someone meeting them for the first time.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

REIFICATION / TIMING / LEARNING

"Causalontology is the world's first synthesis of reified causal
representation, temporal-mechanistic modeling, and interventional learning
into a single glass-box unified ontology."

What the words mean:

- Reified causal representation.

To "reify" something is to turn an idea into a thing you can hold, name, point at, and examine.

Most computer systems store a cause and effect as a bare arrow, "A caused B."

Causalontology instead makes each cause-and-effect link a full object in its own right - the Causal Relation Object - that carries its own details: what caused what, how strong the link is, how long the delay is, and where the knowledge came from.

Making causation a first-class object is what lets the system reason about it, improve it, and explain it.

- Temporal-mechanistic modeling.

"Temporal" means having to do with time.

"Mechanistic" means describing the actual mechanism - the step-by-step way one thing brings about another.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Causalontology treats the timing of a cause (how long before the effect, and for how long) as part of the mechanism, not as an afterthought.

This is why it can rule out a suspected cause purely on timing, the way a doctor asks "how long ago did you eat?" to decide which meal made you ill.

- Interventional learning.

To "intervene" is to act on the world - to do something - rather than only to watch.

A child learns that a switch turns on a light by flipping it, not by staring at it.

Causalontology learns its causes the same way: it acts, sees what follows, and writes down the rule.

Learning by doing gives much stronger evidence of true cause than learning by watching, because acting removes the ambiguity of mere coincidence.

- Glass-box.

A "black box" gives you an answer but hides its reasons.

A "glass box" lets you see every step.

Every conclusion Causalontology reaches can be traced back to the named rule that produced it, so you can always ask "why?" and get a readable answer.

- Ontology.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

An "ontology" is an organized inventory of what exists and how the things in it relate - a formal map of a domain's nouns and verbs.

A "unified" ontology holds all of this in one coherent framework.

NOUN / VERB GOVERNANCE

"Causalontology is the world's first synthesis of a process-first causal
calculus, a continuant noun backbone, and a realizable hinge between them -
one foundational ontology governing the verbs and hosting the nouns."

What the words mean:

- Process-first.

To be "process-first" is to treat happenings - processes and events - as the primary building blocks, rather than treating static objects as primary.

Causalontology starts from what unfolds and brings things about.

- Causal calculus.

A "calculus" here is just a precise set of rules for combining and reasoning about something - in this case, causes.

It is the machinery for chaining causes into predictions and plans.

- Continuant.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

In formal ontology a "continuant" is a thing that endures through time - a ball, a door, a person.

Ordinary language calls these nouns.

- Occurrent.

An "occurrent" is a thing that happens and unfolds through time - a push, an opening, a journey.

Ordinary language calls these verbs.

- Noun backbone.

A "backbone" is a thin, sturdy spine that other things attach to.

Causalontology keeps only a light spine of nouns and leaves the deep detail of any particular subject to specialist maps - so it "hosts" the nouns without trying to own every fact about them.

- Realizable hinge.

A "hinge" joins two parts while letting them move.

A "realizable" is something noun-shaped that is only fully itself when it happens - a disposition (fragility), a function (a key's job), or a role (being a password).

It belongs with the nouns but points at the verbs, so it is the natural joint between the two.

- Governing versus hosting.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Causalontology is authoritative about the verbs (it "governs" them - its causal objects are the last word) but modest about the nouns (it merely "hosts" them on the thin backbone and defers to specialists).

Taken together, this angle gives the paradigm its name: Causalontology is a verb-first noun-hosting ontology.

THE BRIDGE

"Causalontology is the world's first synthesis of symbolic causal reasoning,
the philosophy of process, and embodied interventional learning, unified in a
single glass-box ontology."

What the words mean:

- Symbolic causal reasoning.

"Symbolic" means working with named symbols and logical rules - like algebra or a legal argument - rather than with statistical weights.

Symbolic causal reasoning draws conclusions about causes by following inspectable rules, which is exactly what makes the reasoning explainable.

- The philosophy of process.

This is the long tradition in philosophy that says reality is fundamentally made of processes and change, not of frozen objects.

Causalontology takes that idea seriously and builds on it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

- Embodied interventional learning.

"Embodied" means having a body - or a stand-in for one, such as control of a game or a device - so that the system can actually act in a world.

Combined with intervention (acting, not just watching), this is learning that comes from a body doing things and feeling the results.

- The bridge.

The novelty is joining three worlds that are usually kept apart - rule-based reasoning, process philosophy, and learning-by-doing - into one framework you can still see inside.

SCHOLARLY FRAMING

"Causalontology is the world's first synthesis of process ontology, causal
modeling, and constructivist learning into a single foundational ontology."

What the words mean:

- Process ontology.

An ontology (organized inventory of what exists) that takes processes as primary - the formal version of "process-first" above.

- Causal modeling.

Building explicit models of what causes what, strong enough to predict the effects of actions

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

and to reason about what would have happened otherwise.

• Constructivist learning.

"Constructivism," associated with the psychologist Jean Piaget, is the view that a mind builds its understanding for itself, step by step, through interaction with the world, rather than being handed finished knowledge.

A constructivist learner constructs and revises its own models as it acts.

• Foundational ontology.

A "foundational" (or "upper-level") ontology is a general framework meant to sit underneath many specific ones and tie them together - a shared foundation other maps can build on.

SUCCINCTLY

"Causalontology is the world's first synthesis of reification, temporal causation, and interventional learning in one unified, glass-box ontology."

What the words mean:

• Reification - turning the idea of a cause into a full, examinable object.

• Temporal causation - causation that carries its own timing as part of the mechanism.

• Interventional learning - learning by acting on the world and watching what follows.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• One unified, glass-box ontology - all of it held in a single organized, see-through framework where every conclusion can be explained.

A note for the standard's readers: the claim above is the IDENTITY of Causalontology; the parts that follow are its ENGINEERING.

This Standalone Branch adds a further first of its own - to our knowledge, the first specification to deliver reified, provenance-signed causation as a programming-language-neutral standard with a shared, stigmergic commons - but that claim, like every claim in a glass-box system, is offered with its evidence: the specification, the conformance vectors, and the running implementations are the proof, and they are all inspectable.

THE SEVEN GOVERNING DESIGN PRINCIPLES

Seven principles govern every decision in this standard, and they are the tie-breakers for any ambiguity in the Parts that follow. They are numbered P1 through P7, and this is the whole set - no principle is numbered outside this range, and none is missing from within it. Each is stated here once, canonically; where a principle is also carried elsewhere as a named rule, that place is cited so the two never drift apart.

P1. Type and token are two tiers, and the arrow runs one way. Every token record MUST name the type it instantiates - there are no free-floating tokens - and no type record may reference a token, with the single exception of an assertion's evidenced_by, which is provenance, not content. This is what preserves the commons: the type tier

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

is the shared, world-wide body of causal law, while the token tier is a particular mind's history that CITES the commons rather than polluting it, so a store can publish its types while withholding its tokens.

P2. Tokens are local by default. A conformant store MUST NOT publish token-tier records (token_individual, token_occurrence, state_assertion, token_causal_claim) to a federated commons unless its operator explicitly instructs it, per record or per policy. The rationale is safety, and it is made normative and non-optional as safety rule R1 of Part 11 (spec/safety.md).

P3. Levels are scheme-relative. A stratum has meaning only within a named stratification scheme, and two strata are comparable if and only if they share a scheme; comparing ordinals across schemes is refused as a HARD failure (Semantics Rule 13, Part 7). Rival stratification schemes - a neuroscientist's layers, a physicist's energy scales, an engineer's network model - coexist, exactly as rival causal claims coexist.

P4. Content carries no belief. No content kind carries strength, confidence, probability, source, or timestamp-of-assertion; who believes a thing, and how strongly, is provenance, kept in the signed records. Absolute time on a token occurrence is not a violation, because it records when the WORLD acted, not when a source claimed.

P5. Absence must be distinguishable from assertion. Where a field's absence is ambiguous between "not yet known" and "known to be nothing", the standard provides a way to assert the latter

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

positively; this principle generates exactly one field, the Causal Relation Object's skips, and is stated as a principle rather than buried as a rationale so it is not misread as a defect.

P6. Every addition must be forced. No construct enters the standard unless a recorded, specific failure to encode a real body of causal knowledge forced it; constructs that are tempting but not forced are declined, and the declinations are recorded so no future proposer need re-litigate them. This is the required-fields minimalism of Part 3 and the forced-addition change process of GOVERNANCE.md.

P7. Whole words, never abbreviations. Every name minted by this standard is a whole English word, or an underscore-joined phrase of whole English words, in lowercase snake_case; an abbreviation, acronym, initialism, or truncation MUST NOT be used as an identifier scheme, a type value, a field name, or a label. The one exception is the proper names of external standards (ed25519, SHA-256, RFC 8785, RFC 3339, UCUM, UTC, JSON, JSON-LD, BFO, RO, PROV), which are kept verbatim because renaming them would make them wrong, not clearer. This principle is also carried as Locked Decision 30 of Part 17.

PART 2. THE STANDALONE REPOSITORY: SHIP THE SPECIFICATION, NOT THE CODE

A repository, github.com/ai-university-aiu/causalontology, whose core is normative text and schemas rather than any programming language:

```
causalontology/
  spec/
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        causalontology.md          the normative
specification (the canon)
        schema/
          causal_relation_object.schema.json   the
Causal Relation Object (Part 6.1)
          occurrent.schema.json    occurrents -
the verbs (Part 6.2)
          continuant.schema.json   continuants -
the nouns (Part 6.3)
          realizable.schema.json   realizable
entities - the hinge (Part 6.4)
          stratum.schema.json      strata - levels
of description (Part 6.10)
          bridge.schema.json       bridges -
cross-stratal identity (Part 6.11)
          cross_stratal_seam.schema.json  cross-
stratal seams - the managed jump (Part 6.19)
          port.schema.json         ports - typed
interfaces (Part 6.12)
          conduit.schema.json      conduits -
typed connections (Part 6.13)
          quality.schema.json      qualities -
property types (Part 6.14)
          individual.schema.json          a
particular thing (Part 6.15)
          token.schema.json               a
particular happening (Part 6.16)
          state.schema.json               a value
over an interval (Part 6.17)
          token_causal_claim.schema.json  a
particular causing (Part 6.18)
          assertion.schema.json    signed claim-
provenance records (Part 6.5)
          enrichment.schema.json   signed
enrichment records (Part 6.6)
          retraction.schema.json   signed
retraction records (Part 6.7)
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
          succession.schema.json   signed key-
succession records (Part 6.8)
          context.jsonld           the JavaScript
Object Notation for Linked Data (JSON-LD) context
                                   for linked-data
interoperability (Part 6.9)
          causalontology.proto     optional
Protocol Buffers (Protobuf) schema
                                   for a compact
binary encoding
        identity.md
canonicalization, hashing, and merge (Part 4)
        semantics.md               the rules
beyond the schemas (Part 7)
        provenance.md              trust,
signatures, enrichment, retraction,
                                   succession
(Part 8)
        store.md                   operations, the
Hypertext Transfer Protocol
                                   (HTTP) binding,
query, resolve, pagination
                                   (Part 9)
        safety.md                  abuse
resistance, claims of consequence, and
                                   takedown by
tier (Part 11)
      conformance/
        vectors/                   the one hundred
and thirty-seven programming-language-neutral test
cases
                                   (Part 14), one
file per vector
        README.md                  "an
implementation is conformant if and only
                                   if it passes
every vector here"
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      bindings/                       thin per-
language libraries or links to their
                                      repositories;
PrologAI is the reference
                                      binding
      store/
        server/                       a reference
store server
        stigmergy/                    gap-signal
queries and the contribution
                                      dashboard
      examples/                       the quickstart
of Part 15, executable
      GOVERNANCE.md                   how the
specification evolves (Part 13)
      LICENSE                         Apache License
2.0 for code and specification
      DATA_LICENSE                    Creative
Commons Zero (CC0) for commons data
```

The value of this repository lives in spec/ and conformance/.

If the specification and the vectors are right, any number of language implementations can be correct and interoperable.

PART 3. THE OBJECT MODEL: TWENTY-ONE KINDS OF THING, AND THE SCOPE OF THE STANDARD

Everything in the Causalontology commons is one of twenty-one kinds of object - seventeen CONTENT kinds (their identity is their content, and they carry NOTHING but that content) and four PROVENANCE kinds (signed records saying who claims, enriches, retracts, or succeeds what).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The seventeen content kinds fall into two tiers. TEN TYPE-TIER kinds describe the world in general, as laws and vocabulary: occurrent, causal_relation_object, continuant, realizable, stratum, bridge, cross_stratal_seam, port, conduit, and quality. SEVEN TOKEN-TIER kinds describe a particular history - what actually happened, on this occasion, to these very things, and what a particular mind expects and contains: token_individual, token_occurrence, state_assertion, token_causal_claim, attitude, predicted_occurrence, and prediction_error. The token tier is DELIVERED, not reserved - seven kinds strong; because a particular history can be private, token-tier objects are LOCAL BY DEFAULT (see spec/safety.md), shared only when a contributor chooses to publish them.

Causalontology 2.0.0 grew the model from the original eight kinds to seventeen by adding four capabilities the earlier canon could not express, plus the layer-skip semantics that ties them together - all catalogued at the close of this Part.

TYPE-TIER CONTENT KINDS (pure, immutable, content-addressed)

• OCCURRENT (identifier prefix occurrent:).

A process or event TYPE that unfolds through time - a verb: button-pressing, a light turning on, a fever developing.

Occurrents are the vocabulary out of which causes and effects are built.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

An occurrent is exactly a canonical label plus a REQUIRED category from a closed enumeration (Part 6.2) - nothing else.

Its multilingual aliases and its participants arrive separately, as signed enrichment records, so that every word added to the commons has an author.

• CAUSAL RELATION OBJECT (prefix causal_relation_object:).

The fundamental unit: a reified causal claim linking cause occurrents to effect occurrents, optionally carrying a mechanism (a sub-graph of finer Causal Relation Objects), a temporal window (the delay range, treated as part of the mechanism), a modality (necessary, sufficient, contributory, preventive, or enabling), enabling context conditions, and a "refines" link to a more partial version of the same claim that it enriches.

It may also carry a skips flag, which declares that the cause reaches its effect across NON-ADJACENT strata without any mechanism at the intervening levels - layer-skipping causation, described under the skip semantics at the close of this Part.

A Causal Relation Object carries NO provenance and NO strength - those belong to assertions - so the same claim from any number of sources is one object.

• CONTINUANT (prefix continuant:).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

A thing that endures through time - a noun: a button, a lamp, a person.

A continuant is exactly a canonical label plus a REQUIRED category from a closed enumeration (Part 6.3).

Its aliases, its is-a links (subsumes), and its part-of links arrive as signed enrichment records.

• REALIZABLE ENTITY (prefix realizable:).

The hinge between nouns and verbs: a disposition, function, or role that inheres in a continuant yet is realized in occurrents (fragility inheres in the glass and is realized in shattering).

Its realized_in links arrive as signed enrichment records.

As of 2.0.0 a realizable also carries an identity-bearing label - the defect repair that lets a single bearer hold many DISTINGUISHABLE dispositions, functions, or roles (a person's role as an author and their role as a reviewer are now two different realizables on the same continuant, no longer collapsed into one).

• STRATUM (prefix stratum:).

A named LEVEL of description within a stated stratification scheme - the molecular level, the cellular level, the psychological level, the economic level.

A stratum is a canonical label plus the name of the stratification scheme it belongs to, so "level

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

three of the neuroscience scheme" and "level three of the economics scheme" are different objects.

Strata are what let a single claim say WHERE, in the ladder of description, it sits.

• BRIDGE (prefix bridge:).

A cross-stratal identity map: the statement that one COARSE occurrent IS a particular SET of finer ones - the same happening, described at two levels.

"A neuron firing" at the cellular level IS "these ion channels opening and these vesicles releasing" at the molecular level: not a cause of it, but the very same event seen closer up.

A bridge is what lets Causalontology know that two claims pitched at different levels are about ONE happening, and it is the machinery behind the amended hierarchy rule (Part 7.7).

• CROSS-STRATAL SEAM (prefix cross_stratal_seam:).

A managed record of a legitimate causal jump across NON-ADJACENT strata - the honest, checkable form of what a Causal Relation Object's bare skips flag could only assert.

It names a source and a target occurrent, whose stratum ordinals are the two ends of the jump, and a mechanism_status that distinguishes "unmodeled" (an intervening mechanism EXISTS but is deliberately not drawn here) from "absent" (there is genuinely NO intervening mechanism) - the honest-ignorance distinction the bare boolean could never make.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

It may optionally DRAW a chain of intervening occurrents that traces the pathway, and a drawn chain forbids the status "absent", because a drawn mechanism is not absent.

The seam belongs to the COARSEST stratum it touches, and it subordinates the Causal Relation Object's skips, which stays valid as the coarse, mechanism-status-unspecified form of the same phenomenon (the eighteenth kind; Semantics Rule 22, Algorithm F, and the normative Part 22).

• PORT (prefix port:).

A typed INTERFACE borne by a continuant - the point at which a thing can be connected to another thing.

A power socket on a wall, a receptor on a cell surface, and the input of an amplifier are all ports: each has a type that says what may legitimately plug into it.

Ports arrive on continuants and give the commons a principled vocabulary of connection.

• CONDUIT (prefix conduit:).

A typed CONNECTION from one port to another - the thing that carries influence between two interfaces.

A conduit is a PIPE if it merely conveys what enters it and a COMPUTER if it TRANSFORMS what passes through in transit; which one it is is decided by the Perfect Wire Test - ask whether a perfect, lossless wire could stand in its place

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

with no change to the outcome; if yes, it is a pipe, and if the transformation matters, it is a computer.

Ports and conduits together are how the commons says that two things are CONNECTED, and how it tells plumbing apart from computation.

• QUALITY (prefix quality:).

A PROPERTY TYPE that a thing can bear - a mass, a temperature, a colour, a charge.

A quality is a canonical label plus a REQUIRED datatype - one of quantity, categorical, or boolean - together with an optional UCUM unit that is required only when the datatype is quantity (and absent otherwise) and an optional stratum; it carries no category field.

Qualities give continuants their measurable properties without inflating the noun backbone, and they are the natural anchor for token-tier state assertions (below).

TOKEN-TIER CONTENT KINDS (a particular history, not just laws; LOCAL BY DEFAULT for privacy, per spec/safety.md)

Where the type tier states general laws and vocabulary, the token tier records what ACTUALLY OCCURRED - a particular history, with particular things at particular times. Because a history can be personal, these objects are held locally by default and enter the shared commons only by a deliberate act of publication.

• TOKEN INDIVIDUAL (prefix token_individual:).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

A PARTICULAR thing - not the type "lamp" but THIS lamp, the one on my desk. A token individual instantiates a continuant type and stands as the subject of the particular happenings and values recorded about it.

- TOKEN OCCURRENCE (prefix token_occurrence:).

A PARTICULAR happening at absolute time - not the type "button-pressing" but THIS press, at 15:07:02 on this date. A token occurrence instantiates an occurrent type and pins it to the clock.

- STATE ASSERTION (prefix state_assertion:).

A PARTICULAR VALUE holding over a PARTICULAR INTERVAL - "this battery's charge was eighty percent from 15:00 to 15:30". A state assertion binds a quality (the property type) to a token individual (the bearer) over a stated stretch of absolute time.

- TOKEN CAUSAL CLAIM (prefix token_causal_claim:).

A PARTICULAR CAUSING - "THIS press caused THIS light to come on" - optionally placed under a covering law (a type-level Causal Relation Object) and optionally carrying a token-level counterfactual, the but-for test: WOULD this light have come on, on this occasion, had this press not happened? The token causal claim is where Judea Pearl's third rung - the counterfactual - finally has a home in the data itself.

- ATTITUDE (prefix attitude:).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

A record of what a particular mind CONTAINS - never of what is true. An attitude names a HOLDER (a MODELED agent inside the ontology - a continuant type or a particular token individual - never a cryptographic key), one of six CLOSED attitude types (believes, desires, intends, knows, expects, fears), and a CONTENT: a reference by identity to any content object, which may be FALSE and may itself be another attitude ("A believes that B believes X" - theory of mind, by composition). An attitude asserts nothing: its content is QUARANTINED from every reasoning, conflict, reachability, and classification rule (Rule 25, the doxastic rule), it carries no strength (Principle P4), and it is asserted, graded, and retracted through the ordinary provenance layer like any other content object (the nineteenth kind; the normative Part 23).

- PREDICTED OCCURRENCE (prefix predicted_occurrence:).

An EXPECTATION made first-class: an occurrent type predicted, by a stated predictor, to occur over a stated window - the sibling of the token occurrence that has not (yet) happened. A prediction is NOT an assertion that the thing happened, and it satisfies no rule that requires a token_occurrence; its interval carries exactly ONE temporal dimension - a wall-clock start or an ordinal start_tick, never both and never neither (Rule 24) - and it may carry the predicted probability as an optional strength that is PREDICTED CONTENT, part of the expectation itself, not a provenance evaluation (the twentieth kind; the normative Part 23).

- PREDICTION ERROR (prefix prediction_error:).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

How a prediction met the world: the signed, graded discrepancy - actual minus expected - between a predicted occurrence and what was, or was not, observed. When observed is present it names the token occurrence that answered the prediction, and that occurrence must instantiate the same occurrent the prediction instantiates (Rule 24); when observed is ABSENT, the prediction went unfulfilled by any recorded occurrence - the surprise of nothing arriving, itself a first-class record (the twenty-first kind; the normative Part 23).

PROVENANCE KINDS (signed, add-only records)

• ASSERTION (prefix assertion:).

A signed statement by a source about a content object: "I, source S, assert claim C with strength estimate s, on evidence of type t, with confidence c, at time T."

Many assertions may reference one claim; they never overwrite one another.

An assertion about a non-causal content object means the source vouches for that vocabulary item's well-formedness and aptness.

As of 2.0.0 an assertion may also cite TOKEN evidence through an evidenced_by link - pointing at the particular occurrences that support it - and its evidence ladder gains the SIMULATION grade for an internally simulated (imagined) intervention, ranked below observation and above the weaker grades: the full ladder, strongest to

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

weakest, reads intervention > observation > simulation > derivation > human_hint > imported.

• ENRICHMENT (prefix enrichment:).

A signed statement adding one entry to one enrichment field of one content object: "I, source S, add entry E to field F of object X, at time T."

The valid fields per kind are fixed (Part 7.12): aliases and participants for occurrents; aliases, subsumes, and part_of for continuants; realized_in for realizables; and occurrent_subsumes and occurrent_part_of for occurrents, so that occurrents too can be CLASSIFIED (one happening-type is a kind of another) and DECOMPOSED (one happening-type is part of another) - the same is-a and part-of discipline the nouns already enjoyed, now extended to the verbs.

The store MATERIALIZES an object's enrichment sets as views over these records, each entry carrying its contributor(s).

The same entry contributed by two sources is two records and one displayed entry with two contributors - corroboration, not duplication.

• RETRACTION (prefix retraction:).

A signed statement by a source withdrawing one of its OWN earlier records - an assertion OR an enrichment: "I, source S, retract record X at time T."

Retraction is add-only like everything else: the retracted record remains in history, but default views exclude it (Part 8.6).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

This is also how an author cleanly removes an alias it now regrets.

• SUCCESSION (prefix succession:).

A signed statement rotating a source's cryptographic identity: "key K2 succeeds key K1", signed by K1.

Records under any key in the succession chain attribute to one source lineage (Part 8.7).

THE SCOPE OF THE STANDARD, DECLARED

This specification models BOTH LEVELS of causation. Its TYPE TIER models TYPE-LEVEL (generic) causation: claims that relate KINDS of occurrents - "button-pressing causes light-on", "eating spoiled shellfish causes gastroenteritis within one to six hours". Its TOKEN TIER models TOKEN-LEVEL (actual, particular-occasion) causation - "THIS press at 15:07 caused THIS light to come on".

The token-level kinds - token_individual, token_occurrence, state_assertion, and token_causal_claim - each carry their own particular participants and absolute times, and each may cite the type-level claims of the commons as its covering laws. In 4.0.0 three further token-tier kinds join them - attitude, predicted_occurrence, and prediction_error - so a particular history can also record what a mind CONTAINS, what it EXPECTED, and how the expectation FARED (Part 23). What earlier editions reserved as a FUTURE object kind is now a shipped part of the standard.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Because a particular history can concern identifiable people, the token tier is LOCAL BY DEFAULT (spec/safety.md): consumers who need token-level reasoning (episodic memory, legal causation, incident analysis) hold their tokens privately and publish only what they choose. The only item still reserved for a future major release is the optional bounty layer (Part 10).

The load-bearing design decision of this standard, now applied uniformly: the separation of CONTENT from PROVENANCE.

What is claimed or named is an immutable, content-addressed object; everything anyone DOES about it - asserting it, enriching it, retracting one's word, rotating one's key - is a signed, add-only record.

It is what lets independent contributors of the same relation produce one object; what lets contradictory confidences coexist; what makes every word in the commons attributable; what gives an honest exit (retraction) and a durable identity (succession); and what makes the whole store merge like a Conflict-free Replicated Data Type (CRDT - a data structure that merges concurrent edits deterministically with no central coordinator).

WHAT 2.0.0 ADDED, IN PLAIN WORDS

The jump from eight kinds to seventeen answered four questions the earlier canon could pose but not record, and added the layer-skip semantics that binds them:

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

(1) WHAT LEVEL a claim sits at - answered by the STRATUM. Every claim can now say where, in a named ladder of description, it belongs.

(2) That two claims at different levels are the SAME HAPPENING - answered by the BRIDGE. A coarse occurrent can be declared to BE a set of finer ones: not a cause of them, but the identical event seen closer up.

(3) That two things are CONNECTED - answered by PORTS and CONDUITS. A thing bears typed interfaces (ports), and a typed connection (a conduit) runs from one port to another, sorted by the Perfect Wire Test into a pipe that conveys and a computer that transforms.

(4) That something ACTUALLY OCCURRED - answered by the TOKEN TIER. The commons now holds a history, not only laws: particular things, particular happenings at absolute time, particular values over intervals, and particular causings with their but-for counterfactuals.

THE SKIP SEMANTICS.

Alongside the four additions comes a fifth idea that binds the levels together: the skips flag on a Causal Relation Object. Normally a cause reaches its effect through a mechanism at the next level down, and the hierarchy rule checks that the finer steps chain up to the coarse claim. But some causes genuinely LEAP across many levels in a single physical step. A stress hormone released into the blood crosses roughly ten levels of description - from the endocrine system down to the molecule - and binds DNA directly, with no intervening cellular relay to be found. The skips

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

flag lets a claim ASSERT exactly this: that the cause crosses non-adjacent strata WITHOUT a mechanism at the intervening levels. The absence of a mechanism then becomes a positive FINDING - a claim that layer-skipping is real here - rather than a gap to be filled forever by a mechanism that does not exist.

This is why 2.0.0 also amends Rule 7 (hierarchy consistency) to reason over BRIDGED reachability (Part 7.7): with bridges declaring same-happening identities across levels and skips declaring honest leaps, the consistency check can finally be run across MULTIPLE SCALES at once, which the type-only canon could not do. Two further refinements complete the set: the ENABLING modality, a fifth value of a Causal Relation Object's modality, for a cause that makes an effect POSSIBLE without bringing it about; and Principle P7 - whole words, never abbreviations - which governs every identifier scheme and type value in the standard (occurrent, causal_relation_object, stratum, bridge, port, conduit, quality, token_individual, token_occurrence, state_assertion, token_causal_claim, assertion, enrichment, retraction, succession), so that no reader ever meets a three-letter code where a readable word belongs.

PART 4. IDENTITY: CONTENT-ADDRESSED IDENTIFIERS, CANONICALIZATION, AND MERGE

4.1 The rule.

Every object's identifier is derived deterministically from its own canonical content

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

by a cryptographic hash – the Secure Hash Algorithm 256-bit (SHA-256).

An identifier is written scheme:localpart, where the scheme is a whole-word kind name – occurrent, causal_relation_object, continuant, realizable, stratum, bridge, port, conduit, quality, token_individual, token_occurrence, state_assertion, token_causal_claim, assertion, enrichment, retraction, or succession (Principle P7: whole words, never abbreviations) – and the localpart is the lowercase hexadecimal encoding of the SHA-256 digest (sixty-four hexadecimal characters).

Optionally a human-readable Uniform Resource Identifier (URI) or Persistent Uniform Resource Locator (PURL) may be registered as an alias for an identifier.

4.2 Why content-addressing is load-bearing.

• Coordination-free merge: two contributors anywhere who independently express the same claim compute the same identifier, so their contributions are the same object.

De-duplication needs no central authority.

• Tamper-evidence: change the content and the identifier changes; history becomes verifiable.

• A migration path: content-addressing is exactly what a decentralized store needs (Part 9), so adopting it on day one keeps that option open without redesign.

An honest note on vocabulary.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Content-addressing is ideal for STRUCTURED claims (a Causal Relation Object's content is rich and distinctive) and inherently awkward for bare VOCABULARY items, whose content is little more than a label - which is why Wikidata chose opaque identifiers for its items.

Causalontology keeps content-addressing even for vocabulary, and buys convergence with three disciplines: a canonical-label rule, a CLOSED category enumeration (so the identity inputs are deterministic), and a REQUIRED resolve-before-mint workflow (check the store for an existing occurrent or continuant before creating one).

Residual near-synonyms that slip through are handled the same way Wikidata handles duplicates: socially, by a merge notice - here, an assertion marking one item as the alias of another.

4.3 Canonicalization, precisely.

Two implementations must hash identically, so the byte form that is hashed is fixed by adopting an existing standard:

• Request for Comments (RFC) 8785, the JSON Canonicalization Scheme (JCS), published by the Internet Engineering Task Force.

JCS fixes key ordering, string escaping, and - the notoriously tricky part - number serialization (the number written 1.0 and the number written 1 canonicalize to the same bytes), so identity cannot depend on formatting accidents.

• Procedure to compute an object's identifier:

© 2026-01-01 All Rights Reserved.

1. Take the object as a JavaScript Object Notation (JSON) document.
2. Remove the fields that are NOT identity-bearing for its kind (4.4).
3. Apply RFC 8785 canonicalization to the remainder.
4. Hash the resulting bytes with SHA-256.
5. The identifier is scheme + ":" + lowercase hexadecimal digest.

4.4 Identity-bearing fields, by kind.

Kind	Identity-bearing fields
occurrent	type, label, category, stratum
causal_relation_object	type, causes, effects, mechanism, temporal, modality, context, refines, skips
continuant	type, label, category
realizable	type, kind, bearer, label
stratum	type, label, scheme, ordinal, unit, governs
bridge	type, coarse, fine, relation
cross_stratal_seam	type, source, target, mechanism_status, chain
port	type, bearer, label, direction, accepts, realizable
conduit	type, label, from, to, carries, transform, realized_by
quality	type, label, datatype, unit, stratum
token_individual	type, instantiates, designator, part_of
token_occurrence	type, instantiates, interval, participants, locus, observer

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
   state_assertion     type, subject, quality,
value, interval
   token_causal_claim  type, causes, effects,
covering_law,
                       actual_delay, counterfactual
   assertion    type, about, source,
evidence_type, evidence, strength,
                       confidence, timestamp,
evidenced_by
   enrichment   type, about, field, entry, source,
timestamp
   retraction   type, retracts, source, timestamp
   succession   type, predecessor, successor,
timestamp
```

Exclusions, now maximally simple:

- "id" is always excluded (it IS the hash).
- "signature" is excluded from the four provenance kinds; the signature is computed over those same canonical bytes (Part 8.3).
- NOTHING ELSE IS EXCLUDED. As of Version 4, content objects consist of exactly their identity-bearing fields - there are no mutable tails, so there is nothing on any object that identity does not cover.

Note that an enrichment's source and timestamp ARE identity-bearing: the same entry contributed by two sources (or twice by one source at different times) is deliberately two distinct records - that is what corroboration is.

4.5 Merge semantics, now trivial.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The store's replication story collapses to two rules:

• Content objects are IMMUTABLE.

Writing one whose identity already exists is a no-operation (idempotent).

There is nothing on it to merge.

• Provenance records are ADD-ONLY.

A record is either present or not; writing it twice is idempotent (Ed25519 signatures are deterministic per RFC 8032, so even the signature bytes agree).

A replica set therefore merges by SET UNION of objects and records, in any order, with no coordinator - the store is a Conflict-free Replicated Data Type by construction, with no special cases.

The familiar grow-only enrichment sets still exist, but as DERIVED, MATERIALIZED VIEWS over enrichment records (Part 8.5), never as state inside an object.

Nothing is ever removed by merge; removal from view is done by retraction (the author) or suppression (policy), and the underlying history is never erased.

4.6 Enrichment of claims: the refines link.

Because identifiers are content-derived, enriching a partial Causal Relation Object necessarily mints

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

a NEW object (its identity-bearing content changed).

The "refines" field carries the lineage: the enriched object names the partial object it supersedes.

Rules: the refines graph must be acyclic; a refinement must preserve every field value its parent specified and may only ADD fields the parent left unspecified (changing a parent's specified value is not a refinement - it is a rival claim); and gap queries exclude any object that is the target of a valid refinement (Part 10), which is how a gap visibly closes.

PART 5. CROSS-LANGUAGE INTEROPERABILITY: THROUGH DATA, NOT THROUGH CODE

If Causalontology is implemented separately in Python, Java, and Swift, how do those codebases interface?

They do not call one another.

They all speak the same canonical serialization and all talk to the same store protocol.

Interoperability is achieved through shared DATA and a shared PROTOCOL, never through a Foreign Function Interface (FFI) or any language bridge - the same lesson as many programs sharing one database or reading one file format.

5.1 The canonical serialization.

• The canonical wire format is JSON, with the formal JSON Schemas of Part 6.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• The recommended profile is JSON for Linked Data (JSON-LD), which layers a small context file onto plain JSON so that Causalontology data is also valid Resource Description Framework (RDF) data - the standard model for linked data on the web.

This single choice buys interoperability with the entire semantic-web and Wikidata ecosystem.

• For high scale, Protocol Buffers (Protobuf) or the Concise Binary Object Representation (CBOR) may be used as equivalent binary encodings; the JSON form remains normative and the binary forms must round-trip to it without loss.

5.2 Per-language Software Development Kits (SDKs).

• causalontology-py on the Python Package Index; causalontology-java on Maven Central; causalontology-swift on the Swift Package Manager; further languages as contributed.

Package names are lowercase, matching the identifier register of the name.

• Data types can be generated from the schemas; only the semantics (Part 7), identity and merge (Part 4), and signing (Part 8) are hand-written, and they are small.

• SDKs SHOULD offer convenience sugar such as mint_with_aliases(label, category, aliases), which emits one pure content object plus one self-attributed, signed enrichment record per alias - so ergonomics never tempt an implementation to put mutable state back inside objects.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Every SDK must expose the abstract operations of Part 9.1 and must ship the conformance vectors as its own test suite.

5.3 The conformance rule.

An implementation is Causalontology-conformant if and only if it passes every vector in conformance/vectors/ for the specification version it declares.

That single rule is what guarantees that Prolog, Python, Java, and Swift agree without sharing a line of code - the mechanism behind the W3C test suites, the JSON Schema test suite, and the WebAssembly specification tests.

Continuous Integration (CI) in each binding runs the vectors on every change and refuses to publish a release that fails.

PART 6. THE SCHEMAS (THE MACHINE-CHECKABLE STRUCTURE, IN FULL)

All schemas use the 2020-12 draft of JSON Schema.

Structural validity is defined by these schemas; semantic validity adds the rules of Part 7.

Two deliberate, locked decisions run through them: in the Causal Relation Object, only id, causes, and effects are required, so partial and degenerate objects are first-class (the engine of subsumption and stigmergy); and content objects carry ONLY identity-bearing fields, so every mutable datum in the commons is a signed record with an author.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

6.1 spec/schema/causal_relation_object.schema.json - the Causal Relation Object

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/causal_relation
_object.schema.json",
    "title": "Causal Relation Object",
    "type": "object",
    "required": [
      "id",
      "causes",
      "effects"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "$ref": "#/$defs/croId"
      },
      "type": {
        "const": "causal_relation_object"
      },
      "causes": {
        "type": "array",
        "minItems": 1,
        "items": {
          "$ref": "#/$defs/occId"
        }
      },
      "effects": {
        "type": "array",
        "minItems": 1,
        "items": {
          "$ref": "#/$defs/occId"
        }
      },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "mechanism": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/croId"
        },
        "description": "identifiers of finer
Causal Relation Objects whose composition realizes
this one (hierarchical decomposition; must be
acyclic; consistency per semantics.md rule 7,
amended to BRIDGED reachability in 2.0.0)"
      },
      "temporal": {
        "$ref": "#/$defs/temporalWindow"
      },
      "modality": {
        "enum": [
          "necessary",
          "sufficient",
          "contributory",
          "preventive",
          "enabling"
        ]
      },
      "context": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/occId"
        },
        "description": "enabling conditions under
which the relation holds"
      },
      "refines": {
        "$ref": "#/$defs/croId",
        "description": "the more partial Causal
Relation Object this one enriches (lineage;
acyclic; must preserve the parent's specified
fields)"
      },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "skips": {
        "type": "boolean",
        "description": "TRUE asserts that this
relation crosses non-adjacent strata WITHOUT being
re-encoded at the intervening strata: the absence
of a mechanism is a POSITIVE FINDING, not a gap
(2.0.0 4.4)"
      }
    },
    "$defs": {
      "croId": {
        "type": "string",
        "pattern": "^causal_relation_object:[0-9a-
f]{64}$"
      },
      "occId": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$"
      },
      "temporalWindow": {
        "type": "object",
        "required": [
          "minimum_delay",
          "maximum_delay",
          "unit"
        ],
        "additionalProperties": false,
        "properties": {
          "minimum_delay": {
            "type": "number",
            "minimum": 0
          },
          "maximum_delay": {
            "type": "number",
            "minimum": 0
          },
          "unit": {
            "enum": [
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
              "instant",
              "seconds",
              "minutes",
              "hours",
              "days",
              "weeks",
              "months",
              "years",
              "ticks"
            ],
            "description": "3.0.0: the wall-clock
units carry a fixed seconds mapping; 'ticks' is a
discrete, dimensionless ORDINAL unit (a step
counter) with NO wall-clock mapping - a tick
window is ordered by integer comparison, and a
tick window and a wall-clock window are different
dimensions that do not compare."
          }
        }
      }
    }
  }
```

Notes. No provenance and no strength: both live in assertions (6.5). Cause, effect, and context references MUST be occurrent identifiers - free-text labels are not permitted, because they would break content-addressed de-duplication; the words live on the occurrent's enrichment records, with authors. Every field is identity-bearing: the object is fully immutable.

6.2 spec/schema/occurrent.schema.json - occurrents, the verbs (pure)

```
  {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/occurrent.schem
a.json",
    "title": "Occurrent",
    "type": "object",
    "required": [
      "id",
      "label",
      "category"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$"
      },
      "type": {
        "const": "occurrent"
      },
      "label": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "canonical label: English,
lowercase snake_case, whole words (press_button,
light_on); homonyms disambiguated by qualification
(charge_battery, charge_attack)"
      },
      "category": {
        "enum": [
          "action",
          "event",
          "process",
          "state",
          "state_change"
        ],
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "description": "REQUIRED and CLOSED so
identity inputs are deterministic; extending this
enumeration is a MAJOR version change
(GOVERNANCE.md)"
      },
      "stratum": {
        "type": "string",
        "pattern": "^stratum:[0-9a-f]{64}$",
        "description": "OPTIONAL, IDENTITY-
BEARING: the level of description at which this
occurrent is pitched (2.0.0 N3.3)"
      }
    }
  }
```

Notes. Pure as of Version 4: no aliases field, no participants field - those arrive as signed enrichment records (6.6) and are served as materialized views (Part 8.5). Identity = hash(type, label, category), all three required and the category closed - two contributors defining the same concept converge on the same identifier deterministically. The REQUIRED workflow before minting is resolve-before-mint (Part 9.4).

6.3 spec/schema/continuant.schema.json - continuants, the nouns (pure)

```
{
  "$schema": "https://json-
schema.org/draft/2020-12/schema",
  "$id":
"https://causalontology.org/schema/continuant.sche
ma.json",
  "title": "Continuant",
  "type": "object",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    "required": [
      "id",
      "label",
      "category"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^continuant:[0-9a-f]{64}$"
      },
      "type": {
        "const": "continuant"
      },
      "label": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "canonical label, same
discipline as occurrents; homonyms by
qualification (bat_animal, bat_club)"
      },
      "category": {
        "enum": [
          "object",
          "agent",
          "place",
          "substance",
          "collection",
          "information"
        ],
        "description": "REQUIRED and CLOSED;
extending this enumeration is a MAJOR version
change"
      }
    }
  }
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Notes. Pure: aliases, subsumes, and part_of arrive as enrichment records.

6.4 spec/schema/realizable.schema.json - realizable entities, the hinge (pure)

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://causalontology.org/schema/realizable.schema.json",
  "title": "Realizable Entity",
  "type": "object",
  "required": [
    "id",
    "kind",
    "bearer"
  ],
  "additionalProperties": false,
  "properties": {
    "id": {
      "type": "string",
      "pattern": "^realizable:[0-9a-f]{64}$"
    },
    "type": {
      "const": "realizable"
    },
    "kind": {
      "enum": [
        "disposition",
        "function",
        "role"
      ]
    },
    "bearer": {
      "type": "string",
      "pattern": "^continuant:[0-9a-f]{64}$"
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      },
      "label": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "OPTIONAL, IDENTITY-
BEARING: the name of this disposition, function,
or role. Repairs the 1.0.0 identity collision that
limited a bearer to one realizable per kind (2.0.0
4.3)"
      }
    }
  }
```

Notes. The label is identity-bearing as of 2.0.0: two roles on the same bearer with different labels are two distinct realizables. Pure: realized_in links (the seam from the nouns into the verbs) arrive as enrichment records.

6.5 spec/schema/assertion.schema.json - signed claim-provenance records

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/assertion.schem
a.json",
    "title": "Assertion",
    "type": "object",
    "required": [
      "id",
      "about",
      "source",
      "evidence_type",
      "confidence",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "timestamp"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^assertion:[0-9a-f]{64}$"
        },
        "type": {
          "const": "assertion"
        },
        "about": {
          "type": "string",
          "pattern":
"^(causal_relation_object|occurrent|continuant|rea
lizable|stratum|bridge|cross_stratal_seam|port|con
duit|quality|token_individual|token_occurrence|sta
te_assertion|token_causal_claim|attitude|predicted
_occurrence|prediction_error):[0-9a-f]{64}$",
          "description": "the content object this
assertion is about (widened in 2.0.0 to the nine
new content kinds; widened in 4.0.0 to attitude,
predicted_occurrence, and prediction_error, and to
the cross_stratal_seam, whose absence here was a
3.0.0 drift repaired in this major version); about
a non-causal object it means the source vouches
for the vocabulary item"
        },
        "source": {
          "type": "string",
          "pattern": "^ed25519:[0-9a-f]{64}$"
        },
        "evidence_type": {
          "enum": [
            "intervention",
            "observation",
            "simulation",
            "derivation",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
            "human_hint",
            "imported"
          ],
          "description": "how the source came to the
claim; ordering from strongest to weakest is
intervention, observation, simulation, derivation,
human_hint, imported (2.0.0 6.2)"
        },
        "evidence": {
          "type": "string"
        },
        "strength": {
          "type": "number",
          "minimum": 0,
          "maximum": 1,
          "description": "the source's estimate of
causal strength (semantics.md rule 5); meaningful
only when about references a Causal Relation
Object"
        },
        "confidence": {
          "type": "number",
          "minimum": 0,
          "maximum": 1
        },
        "timestamp": {
          "type": "string",
          "format": "date-time"
        },
        "evidenced_by": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern":
"^(token_occurrence|token_causal_claim|state_asser
tion):[0-9a-f]{64}$"
          },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "description": "the PARTICULAR token
records that are the evidence for this claim
(2.0.0 6.2). IDENTITY-BEARING."
      },
      "signature": {
        "type": "string",
        "pattern": "^[0-9a-f]{128}$",
        "description": "Ed25519 signature over the
record's canonical identity-bearing bytes;
excluded from the identity hash"
      }
    }
  }
```

Notes. The about pattern reaches every content kind - including attitude, predicted_occurrence, prediction_error, and cross_stratal_seam. It is a CLOSED pattern; extending it is a MAJOR change by Locked Decision 24 that alters no existing record's bytes.

6.6 spec/schema/enrichment.schema.json - signed enrichment records

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/enrichment.sche
ma.json",
    "title": "Enrichment",
    "type": "object",
    "required": [
      "id",
      "about",
      "field",
      "entry",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "source",
        "timestamp"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^enrichment:[0-9a-f]{64}$"
        },
        "type": {
          "const": "enrichment"
        },
        "about": {
          "type": "string",
          "pattern":
"^(occurrent|continuant|realizable):[0-9a-
f]{64}$",
          "description": "the content object being
enriched (Causal Relation Objects are never
enriched in place - they are refined)"
        },
        "field": {
          "enum": [
            "aliases",
            "participants",
            "subsumes",
            "part_of",
            "realized_in",
            "occurrent_subsumes",
            "occurrent_part_of"
          ],
          "description": "field-to-kind validity is
fixed by semantics.md rule 12 (amended in 2.0.0:
two occurrent forms added)"
        },
        "entry": {
          "oneOf": [
            {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
            "type": "object",
            "required": [
              "lang",
              "text"
            ],
            "additionalProperties": false,
            "properties": {
              "lang": {
                "type": "string"
              },
              "text": {
                "type": "string",
                "minLength": 1
              }
            },
            "description": "the entry shape for
the aliases field (lang is a BCP 47 language tag)"
          },
          {
            "type": "string",
            "pattern":
"^(occurrent|continuant):[0-9a-f]{64}$",
            "description": "the entry shape for
the reference fields, checked per field by
semantics.md rule 12"
          }
        ]
      },
      "source": {
        "type": "string",
        "pattern": "^ed25519:[0-9a-f]{64}$"
      },
      "timestamp": {
        "type": "string",
        "format": "date-time"
      },
      "signature": {
        "type": "string",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "pattern": "^[0-9a-f]{128}$"
      }
    }
  }
```

Notes. This record is Version 4's repair of the attribution gap: every alias, every taxonomy link, every participant in the commons now has a signed author. The design rejected the tempting shortcut of an unsigned "contributed_by" tag, which would have enabled a FRAMING ATTACK (adding a poisoned entry attributed to someone else). Attribution without a signature is only a claim of attribution.

6.7 spec/schema/retraction.schema.json - signed retraction records

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/retraction.sche
ma.json",
    "title": "Retraction",
    "type": "object",
    "required": [
      "id",
      "retracts",
      "source",
      "timestamp"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "pattern": "^retraction:[0-9a-f]{64}$"
      },
      "type": {
        "const": "retraction"
      },
      "retracts": {
        "type": "string",
        "pattern": "^(assertion|enrichment):[0-9a-
f]{64}$",
        "description": "the assertion or
enrichment being withdrawn"
      },
      "source": {
        "type": "string",
        "pattern": "^ed25519:[0-9a-f]{64}$",
        "description": "must be the retracted
record's source, or a key in its succession
lineage"
      },
      "reason": {
        "type": "string"
      },
      "timestamp": {
        "type": "string",
        "format": "date-time"
      },
      "signature": {
        "type": "string",
        "pattern": "^[0-9a-f]{128}$"
      }
    }
  }
```

6.8 spec/schema/succession.schema.json - signed key-succession records

```
  {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/succession.sche
ma.json",
    "title": "Succession",
    "type": "object",
    "required": [
      "id",
      "predecessor",
      "successor",
      "timestamp"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^succession:[0-9a-f]{64}$"
      },
      "type": {
        "const": "succession"
      },
      "predecessor": {
        "type": "string",
        "pattern": "^ed25519:[0-9a-f]{64}$"
      },
      "successor": {
        "type": "string",
        "pattern": "^ed25519:[0-9a-f]{64}$"
      },
      "timestamp": {
        "type": "string",
        "format": "date-time"
      },
      "signature": {
        "type": "string",
        "pattern": "^[0-9a-f]{128}$",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "description": "signed by the PREDECESSOR
key"
      }
    }
  }
```

6.9 spec/schema/context.jsonld - the JSON-LD context (draft) Maps Causalontology's fields onto Resource Description Framework properties so a Causalontology document is also linked data, aligned where possible with Basic Formal Ontology (BFO - the upper ontology from which Causalontology takes the continuant/occurrent division), the Relation Ontology (RO - the Open Biological and Biomedical Ontologies' standard relation vocabulary), and the W3C Provenance Ontology (PROV - the web standard for provenance).

```
  {
    "@context": {
      "co":   "https://causalontology.org/ns#",
      "bfo":
"http://purl.obolibrary.org/obo/BFO_",
      "ro":
"http://purl.obolibrary.org/obo/RO_",
      "prov": "http://www.w3.org/ns/prov#",
      "causal_relation_object":
"co:CausalRelationObject",
      "occurrent":  { "@id": "bfo:0000003" },
      "continuant": { "@id": "bfo:0000002" },
      "realizable": { "@id": "bfo:0000017" },
      "causes":     { "@id": "co:hasCause",
"@type": "@id",
                     "@comment": "each cause is
ro:0002411 (causally upstream of) the effects" },
      "effects":    { "@id": "co:hasEffect",
"@type": "@id" },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "mechanism":  { "@id": "co:hasMechanism",
"@type": "@id" },
      "temporal":   { "@id": "co:temporalWindow"
},
      "modality":   { "@id": "co:modality" },
      "context":    { "@id": "co:enablingContext",
"@type": "@id" },
      "refines":    { "@id": "co:refines",
"@type": "@id" },
      "label":      "co:canonicalLabel",
      "category":   "co:category",
      "about":      { "@id": "co:about", "@type":
"@id" },
      "field":      "co:enrichmentField",
      "entry":      "co:enrichmentEntry",
      "realized_in":{ "@id": "bfo:0000054",
"@type": "@id" },
      "subsumes":   { "@id": "co:isA", "@type":
"@id" },
      "part_of":    { "@id": "bfo:0000050",
"@type": "@id" },
      "source":     { "@id":
"prov:wasAttributedTo", "@type": "@id" },
      "evidence_type": "co:evidenceType",
      "evidence":   "co:evidence",
      "strength":   "co:strengthEstimate",
      "confidence": "co:confidence",
      "timestamp":  { "@id":
"prov:generatedAtTime" },
      "retracts":   { "@id":
"prov:wasInvalidatedBy", "@reverse": true },
      "predecessor":{ "@id": "co:predecessorKey",
"@type": "@id" },
      "successor":  { "@id": "co:successorKey",
"@type": "@id" }
    }
  }
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Because a Causal Relation Object is a statement ABOUT a causal statement, the
natural Resource Description Framework home for it is RDF-star (the extension
of RDF for making statements about statements); the context above suffices
for plain-RDF publication, and a fuller Web Ontology Language (OWL) mapping
is a governance-track deliverable.

6.10 spec/schema/stratum.schema.json - strata, levels of description (pure)

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://causalontology.org/schema/stratum.schema.json",
  "title": "Stratum",
  "type": "object",
  "required": [
    "id",
    "type",
    "label",
    "scheme",
    "ordinal"
  ],
  "additionalProperties": false,
  "properties": {
    "id": {
      "type": "string",
      "pattern": "^stratum:[0-9a-f]{64}$"
    },
    "type": {
      "const": "stratum"
    },
    "label": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "canonical label,
snake_case, English (for example: cellular,
macromolecular)"
      },
      "scheme": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "the stratification scheme
this stratum belongs to; strata are comparable
only within a scheme"
      },
      "ordinal": {
        "type": "integer",
        "description": "position within the
scheme; STRICTLY LOWER IS FINER-GRAINED; need not
be contiguous"
      },
      "unit": {
        "type": "string",
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "the principal unit of
analysis at this stratum (for example: cell,
synapse, region)"
      },
      "governs": {
        "type": "array",
        "items": {
          "type": "string",
          "pattern": "^[a-z][a-z0-9_]*$"
        },
        "description": "the governing frameworks
or bodies of law at this stratum (for example:
quantum_mechanics, cell_biology)"
      }
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    }
  }
```

6.11 spec/schema/bridge.schema.json - bridges, cross-stratal identity (pure)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/bridge.schema.j
son",
    "title": "Bridge",
    "type": "object",
    "required": [
      "id",
      "type",
      "coarse",
      "fine",
      "relation"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^bridge:[0-9a-f]{64}$"
      },
      "type": {
        "const": "bridge"
      },
      "coarse": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$",
        "description": "the occurrent at the
coarser stratum"
      },
      "fine": {
        "type": "array",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "minItems": 1,
        "items": {
          "type": "string",
          "pattern": "^occurrent:[0-9a-f]{64}$"
        },
        "description": "the occurrents at the
finer stratum that jointly constitute, aggregate
to, realize, or subvene the coarse one"
      },
      "relation": {
        "enum": [
          "constitutes",
          "aggregates",
          "realizes",
          "supervenes_on"
        ],
        "description": "the cross-stratal relation
asserted; CLOSED enumeration"
      }
    }
  }
```

6.12 spec/schema/port.schema.json - ports, typed interfaces (pure)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/port.schema.jso
n",
    "title": "Port",
    "type": "object",
    "required": [
      "id",
      "type",
      "bearer",
      "label",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "direction",
      "accepts"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^port:[0-9a-f]{64}$"
      },
      "type": {
        "const": "port"
      },
      "bearer": {
        "type": "string",
        "pattern": "^continuant:[0-9a-f]{64}$",
        "description": "the continuant TYPE that
has this interface"
      },
      "label": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "the name of the interface
(for example: perforant_path, fornix)"
      },
      "direction": {
        "enum": [
          "in",
          "out",
          "bidirectional"
        ]
      },
      "accepts": {
        "type": "array",
        "minItems": 1,
        "items": {
          "type": "string",
          "pattern": "^occurrent:[0-9a-f]{64}$"
```

```
        },
        "description": "the kinds of occurrent
this port accepts (direction in) or emits
(direction out) or both (bidirectional)"
      },
      "realizable": {
        "type": "string",
        "pattern": "^realizable:[0-9a-f]{64}$",
        "description": "optional: the disposition,
function, or role this port serves"
      }
    }
  }
```

6.13 spec/schema/conduit.schema.json - conduits, typed connections (pure)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/conduit.schema.
json",
    "title": "Conduit",
    "type": "object",
    "required": [
      "id",
      "type",
      "from",
      "to",
      "carries"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^conduit:[0-9a-f]{64}$"
      },
```

```
      "type": {
        "const": "conduit"
      },
      "label": {
        "type": "string",
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "the name of the connection
(for example: corticospinal_tract)"
      },
      "from": {
        "type": "string",
        "pattern": "^port:[0-9a-f]{64}$"
      },
      "to": {
        "type": "string",
        "pattern": "^port:[0-9a-f]{64}$"
      },
      "carries": {
        "type": "array",
        "minItems": 1,
        "items": {
          "type": "string",
          "pattern": "^occurrent:[0-9a-f]{64}$"
        },
        "description": "the kinds of occurrent
this conduit conducts"
      },
      "transform": {
        "type": "string",
        "pattern": "^causal_relation_object:[0-9a-
f]{64}$",
        "description": "OPTIONAL. ABSENT asserts
the conduit is TRANSMISSIVE. PRESENT names the
Causal Relation Object the conduit performs upon
what it carries, asserting the conduit is
COMPUTATIONAL."
      },
      "realized_by": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": "string",
        "pattern": "^[a-z][a-z0-9_]*:.+$",
        "minLength": 3,
        "description": "3.0.0 OPTIONAL. A scheme-
qualified REFERENCE, by identity, to the native
law or signal that REALIZES this conduit's
transform (for example a native rate law a
computational conduit stands for, or a runtime
signal a transmissive conduit stands for). It is a
reference, not an embedded law - the native
dynamics stay native; the standard only records
the binding. ABSENT (unbound) is legal and
identity-preserving; a conduit that carried no
realized_by in 2.0.0 keeps its identifier under
3.0.0."
      }
    }
  }
```

6.14 spec/schema/quality.schema.json - qualities, property types (pure)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/quality.schema.
json",
    "title": "Quality",
    "type": "object",
    "required": [
      "id",
      "type",
      "label",
      "datatype"
    ],
    "additionalProperties": false,
    "properties": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "id": {
        "type": "string",
        "pattern": "^quality:[0-9a-f]{64}$"
      },
      "type": {
        "const": "quality"
      },
      "label": {
        "type": "string",
        "minLength": 1,
        "pattern": "^[a-z][a-z0-9_]*$",
        "description": "for example:
cortisol_concentration, membrane_potential, mass"
      },
      "datatype": {
        "enum": [
          "quantity",
          "categorical",
          "boolean"
        ]
      },
      "unit": {
        "type": "string",
        "description": "the canonical UCUM unit
code; REQUIRED when datatype is quantity, and MUST
be absent otherwise"
      },
      "stratum": {
        "type": "string",
        "pattern": "^stratum:[0-9a-f]{64}$",
        "description": "optional: the level of
description at which this quality is defined"
      }
    }
  }
```

The next four schemas are the TOKEN TIER - a particular history, LOCAL BY

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

DEFAULT for privacy (Part 11.4, spec/safety.md). Their absolute times use RFC
3339 date-and-time strings.

6.15 spec/schema/individual.schema.json - token individuals, a particular thing

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://causalontology.org/schema/individual.schema.json",
  "title": "Token Individual",
  "type": "object",
  "required": [
    "id",
    "type",
    "instantiates"
  ],
  "additionalProperties": false,
  "properties": {
    "id": {
      "type": "string",
      "pattern": "^token_individual:[0-9a-f]{64}$"
    },
    "type": {
      "const": "token_individual"
    },
    "instantiates": {
      "type": "string",
      "pattern": "^continuant:[0-9a-f]{64}$",
      "description": "the continuant TYPE of which this is an instance; MANDATORY (P1)"
    },
    "designator": {
      "type": "string",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "minLength": 1,
        "description": "an OPAQUE local designator
distinguishing this individual from others of its
type. SEE THE PRIVACY REQUIREMENT, N5.1.3 /
spec/safety.md."
      },
      "part_of": {
        "type": "string",
        "pattern": "^token_individual:[0-9a-
f]{64}$",
        "description": "optional: the particular
individual of which this is a part (TOKEN
mereology)"
      }
    }
  }
```

6.16 spec/schema/token.schema.json - token occurrences, a particular happening at absolute time

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/token.schema.js
on",
    "title": "Token Occurrence",
    "type": "object",
    "required": [
      "id",
      "type",
      "instantiates",
      "interval"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": "string",
        "pattern": "^token_occurrence:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "token_occurrence"
      },
      "instantiates": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$",
        "description": "the occurrent TYPE of
which this is an instance; MANDATORY (P1)"
      },
      "interval": {
        "$ref": "#/$defs/interval"
      },
      "participants": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/rolefill"
        }
      },
      "locus": {
        "type": "string",
        "pattern": "^token_individual:[0-9a-
f]{64}$",
        "description": "optional: the particular
place or bearer at which it occurred"
      },
      "observer": {
        "type": "string",
        "pattern": "^ed25519:[0-9a-f]{64}$",
        "description": "optional: the source whose
experience this token belongs to"
      }
    },
    "$defs": {
      "interval": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": "object",
        "required": [
          "start"
        ],
        "additionalProperties": false,
        "properties": {
          "start": {
            "type": "string",
            "format": "date-time",
            "description": "RFC 3339, UTC,
mandatory Z suffix"
          },
          "end": {
            "type": "string",
            "format": "date-time"
          },
          "open": {
            "type": "boolean",
            "description": "true asserts the
occurrence had NOT ENDED as of this record"
          }
        }
      },
      "rolefill": {
        "type": "object",
        "required": [
          "role",
          "filler"
        ],
        "additionalProperties": false,
        "properties": {
          "role": {
            "enum": [
              "agent",
              "patient",
              "instrument",
              "recipient",
              "source",
```

```
            "destination",
            "medium"
          ],
          "description": "CLOSED enumeration;
extending it is a MAJOR change"
        },
        "filler": {
          "type": "string",
          "pattern": "^token_individual:[0-9a-
f]{64}$"
        }
      }
    }
  }
}
```

6.17 spec/schema/state.schema.json - state assertions, a particular value over a particular interval

```
{
  "$schema": "https://json-
schema.org/draft/2020-12/schema",
  "$id":
"https://causalontology.org/schema/state.schema.js
on",
  "title": "State Assertion",
  "type": "object",
  "required": [
    "id",
    "type",
    "subject",
    "quality",
    "value",
    "interval"
  ],
  "additionalProperties": false,
  "properties": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      "id": {
        "type": "string",
        "pattern": "^state_assertion:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "state_assertion"
      },
      "subject": {
        "type": "string",
        "pattern": "^token_individual:[0-9a-
f]{64}$",
        "description": "the PARTICULAR individual
in this state"
      },
      "quality": {
        "type": "string",
        "pattern": "^quality:[0-9a-f]{64}$"
      },
      "value": {
        "$ref": "#/$defs/value"
      },
      "interval": {
        "$ref":
"https://causalontology.org/schema/token.schema.js
on#/$defs/interval",
        "description": "over what span of absolute
time the subject held this value"
      }
    },
    "$defs": {
      "value": {
        "oneOf": [
          {
            "type": "object",
            "required": [
              "quantity",
              "unit"
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
            ],
            "additionalProperties": false,
            "properties": {
              "quantity": {
                "type": "number"
              },
              "unit": {
                "type": "string",
                "minLength": 1,
                "description": "UCUM unit code;
MUST match the quality's unit"
              }
            }
          },
          {
            "type": "object",
            "required": [
              "categorical"
            ],
            "additionalProperties": false,
            "properties": {
              "categorical": {
                "type": "string",
                "minLength": 1
              }
            }
          },
          {
            "type": "object",
            "required": [
              "boolean"
            ],
            "additionalProperties": false,
            "properties": {
              "boolean": {
                "type": "boolean"
              }
            }
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
          }
        ]
      }
    }
  }
```

6.18 spec/schema/token_causal_claim.schema.json - token causal claims, a particular causing

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/token_causal_cl
aim.schema.json",
    "title": "Token Causal Claim",
    "type": "object",
    "required": [
      "id",
      "type",
      "causes",
      "effects"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^token_causal_claim:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "token_causal_claim"
      },
      "causes": {
        "type": "array",
        "minItems": 1,
        "items": {
          "type": "string",
```

```
          "pattern": "^token_occurrence:[0-9a-
f]{64}$"
        }
      },
      "effects": {
        "type": "array",
        "minItems": 1,
        "items": {
          "type": "string",
          "pattern": "^token_occurrence:[0-9a-
f]{64}$"
        }
      },
      "covering_law": {
        "type": "string",
        "pattern": "^causal_relation_object:[0-9a-
f]{64}$",
        "description": "OPTIONAL. The type-level
Causal Relation Object this token claim
instantiates."
      },
      "actual_delay": {
        "type": "object",
        "required": [
          "duration",
          "unit"
        ],
        "additionalProperties": false,
        "properties": {
          "duration": {
            "type": "number",
            "minimum": 0
          },
          "unit": {
            "enum": [
              "instant",
              "seconds",
              "minutes",
```

```
              "hours",
              "days",
              "weeks",
              "months",
              "years",
              "ticks"
            ],
            "description": "3.0.0: 'ticks' is a
discrete, dimensionless ORDINAL unit; a tick-unit
observed delay compares to a tick-unit covering-
law window by integer, and never to a wall-clock
window."
          }
        },
        "description": "the delay ACTUALLY
OBSERVED on this occasion"
      },
      "counterfactual": {
        "type": "boolean",
        "description": "true asserts the BUT-FOR
test for THIS OCCASION ONLY"
      }
    }
  }
```

6.19 spec/schema/cross_stratal_seam.schema.json - cross-stratal seams, the managed jump across non-adjacent strata (a type-tier content kind; defined in full in the normative Part 22)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/cross_stratal_s
eam.schema.json",
    "title": "Cross Stratal Seam",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    "description": "3.0.0. A first-class MANAGED
record of a legitimate jump across NON-ADJACENT
strata. Where the Causal Relation Object's boolean
'skips' flag can only assert THAT a relation
crosses non-adjacent strata, a Cross Stratal Seam
records HOW HONEST that jump is: its
mechanism_status distinguishes 'a mechanism exists
but is deliberately unmodeled here' from 'no
mechanism exists', and an optional chain draws the
intervening adjacent-stratum pathway. Its HOME is
the coarsest stratum it touches (the endpoint of
greater ordinal), which a layer-to-stratum binding
can check. The 'skips' boolean is SUBORDINATED to
this construct: a bare skipping Causal Relation
Object remains valid and is the coarse, mechanism-
status-unspecified form.",
    "type": "object",
    "required": [
      "id",
      "type",
      "source",
      "target",
      "mechanism_status"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^cross_stratal_seam:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "cross_stratal_seam"
      },
      "source": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$",
```


© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "description": "one endpoint occurrent of
the seam; its stratum ordinal is one end of the
jump"
      },
      "target": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$",
        "description": "the other endpoint
occurrent of the seam; its stratum ordinal is the
other end of the jump"
      },
      "mechanism_status": {
        "enum": [
          "unmodeled",
          "absent"
        ],
        "description": "the HONEST-IGNORANCE
distinction. 'unmodeled': an intervening mechanism
EXISTS across the skipped strata but is
deliberately not drawn here. 'absent': there is
genuinely NO intervening mechanism - the jump is
direct. This is exactly what the boolean skips
flag could not say."
      },
      "chain": {
        "type": "array",
        "minItems": 1,
        "items": {
          "type": "string",
          "pattern": "^occurrent:[0-9a-f]{64}$"
        },
        "description": "OPTIONAL. The intervening
pathway DRAWN as an ordered list of occurrents at
INTERVENING strata (each strictly between the two
endpoints' stratum ordinals, same scheme, strictly
monotone from one endpoint toward the other). When
present the seam is DRAWN, so its mechanism_status
```


© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
must be 'unmodeled', never 'absent' (a drawn
mechanism is not absent)."
      }
    }
  }
```

6.20 spec/schema/attitude.schema.json - attitudes, what a mind contains (a token-tier content kind; defined in full in the normative Part 23)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/attitude.schema
.json",
    "title": "Attitude",
    "description": "4.0.0. A first-class record of
a PROPOSITIONAL ATTITUDE: what a holder's mind
CONTAINS, never what is true. The holder is a
MODELED agent inside the ontology (a continuant
type or a particular token individual), never a
cryptographic key; the content is a reference by
identity to ANY content object - which may be
FALSE, and may itself be another attitude
(nesting). An attitude is NOT an assertion and
carries no strength (Principle P4): it is
asserted, graded, and retracted through the
ordinary provenance layer, and the SOURCE that
signs an assertion ABOUT an attitude is a
different thing from the HOLDER of the attitude
(Rule 25).",
    "type": "object",
    "required": [
      "id",
      "type",
      "holder",
      "attitude_type",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "content"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^attitude:[0-9a-f]{64}$"
        },
        "type": {
          "const": "attitude"
        },
        "holder": {
          "type": "string",
          "pattern":
"^(continuant|token_individual):[0-9a-f]{64}$",
          "description": "the agent whose mind
contains the content: a MODELED agent inside the
ontology (a continuant type or a particular token
individual), never a cryptographic key - the
signer of an assertion about this attitude is a
SOURCE, and the two are different things (Rule
25)"
        },
        "attitude_type": {
          "enum": [
            "believes",
            "desires",
            "intends",
            "knows",
            "expects",
            "fears"
          ],
          "description": "CLOSED enumeration;
extending it is a MAJOR change (Locked Decision
24)"
        },
        "content": {
          "type": "string",
```

```
        "pattern":
"^(occurrent|causal_relation_object|continuant|rea
lizable|stratum|bridge|cross_stratal_seam|port|con
duit|quality|token_individual|token_occurrence|sta
te_assertion|token_causal_claim|attitude|predicted
_occurrence|prediction_error):[0-9a-f]{64}$",
        "description": "a reference by identity to
ANY content object, INCLUDING another attitude
(nesting: A believes that B believes X). The
content records what the holder's mind contains,
not what is true - it may be FALSE, and a content
that contradicts the actual record raises NO
conflict (Rule 25: that mismatch IS a false
belief, first-class and shareable)"
      }
    }
  }
```

6.21 spec/schema/predicted_occurrence.schema.json - predicted occurrences, an expectation made first-class (a token-tier content kind; defined in full in the normative Part 23)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/predicted_occur
rence.schema.json",
    "title": "Predicted Occurrence",
    "description": "4.0.0. A first-class record of
an EXPECTATION: an occurrent type predicted, by a
stated predictor, to occur over a stated window. A
prediction is NOT an assertion that the thing
happened - it is the sibling of the
token_occurrence that has not (yet) happened, and
it satisfies no rule that requires a
token_occurrence (Rule 24). Its interval MUST
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
carry exactly ONE temporal dimension: a wall-clock
start (with optional end) or an ordinal start_tick
(with optional end_tick), never both and never
neither. The optional strength is the predicted
probability - part of the PREDICTED CONTENT
itself, identity-bearing when present, not a
provenance evaluation (the narrow, deliberate
4.0.0 extension of Principle P4).",
    "type": "object",
    "required": [
      "id",
      "type",
      "instantiates",
      "interval",
      "predictor"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^predicted_occurrence:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "predicted_occurrence"
      },
      "instantiates": {
        "type": "string",
        "pattern": "^occurrent:[0-9a-f]{64}$",
        "description": "the occurrent TYPE
predicted to occur; MANDATORY (P1)"
      },
      "interval": {
        "$ref": "#/$defs/predicted_interval"
      },
      "predictor": {
        "type": "string",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "pattern":
"^(continuant|token_individual):[0-9a-f]{64}$",
        "description": "the predicting source - a
MODELED agent or construct inside the ontology (a
continuant type or a particular token individual),
never a cryptographic key"
      },
      "strength": {
        "type": "number",
        "description": "OPTIONAL, identity-bearing
when present: the predicted probability, part of
the predicted content itself, NOT a provenance
evaluation (4.0.0 Locked Decision extension of
Principle P4)"
      }
    },
    "$defs": {
      "predicted_interval": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
          "start": {
            "type": "string",
            "format": "date-time",
            "description": "RFC 3339, UTC,
mandatory Z suffix - the wall-clock dimension"
          },
          "end": {
            "type": "string",
            "format": "date-time"
          },
          "start_tick": {
            "type": "integer",
            "minimum": 0,
            "description": "the ordinal dimension:
a discrete tick with NO wall-clock mapping (Rule
23)"
          },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
            "end_tick": {
              "type": "integer",
              "minimum": 0
            }
          },
          "description": "the predicted window. Rule
24 enforces exactly ONE temporal dimension: start
(wall-clock, optional end) or start_tick (ordinal,
optional end_tick), never both
(dimension_conflict, HARD) and never neither
(missing_dimension, HARD)"
        }
      }
    }
```

6.22 spec/schema/prediction_error.schema.json - prediction errors, how a prediction met the world (a token-tier content kind; defined in full in the normative Part 23)

```
  {
    "$schema": "https://json-
schema.org/draft/2020-12/schema",
    "$id":
"https://causalontology.org/schema/prediction_erro
r.schema.json",
    "title": "Prediction Error",
    "description": "4.0.0. The first-class record
of how a prediction met the world: the graded
discrepancy between a predicted_occurrence and
what was - or was not - observed. When observed is
present it names the token_occurrence that
answered the prediction and MUST instantiate the
same occurrent the prediction instantiates (Rule
24, pairing_mismatch, HARD); when observed is
ABSENT the predicted occurrence was not fulfilled
by any recorded occurrence. discrepancy reads
actual minus expected: positive = exceeded
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
expectation, negative = fell short, zero =
fulfilled.",
    "type": "object",
    "required": [
      "id",
      "type",
      "predicted",
      "discrepancy"
    ],
    "additionalProperties": false,
    "properties": {
      "id": {
        "type": "string",
        "pattern": "^prediction_error:[0-9a-
f]{64}$"
      },
      "type": {
        "const": "prediction_error"
      },
      "predicted": {
        "type": "string",
        "pattern": "^predicted_occurrence:[0-9a-
f]{64}$",
        "description": "the prediction this error
grades; MANDATORY - an error without a prediction
is not an error"
      },
      "observed": {
        "type": "string",
        "pattern": "^token_occurrence:[0-9a-
f]{64}$",
        "description": "OPTIONAL: the token
occurrence that answered the prediction. ABSENT
means the predicted occurrence was not fulfilled
by any recorded occurrence - the surprise of
nothing arriving"
      },
      "discrepancy": {
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": "number",
        "description": "signed, graded; actual
minus expected - positive = exceeded expectation,
negative = fell short, zero = fulfilled (the
comparator sign convention)"
      }
    }
  }
```

PART 7. THE SEMANTICS (THE RULES BEYOND THE SCHEMAS)

A document is SCHEMA-VALID if it matches its JSON Schema, and SEMANTICALLY VALID if it also satisfies these rules.

(JSON Schema cannot compare two fields or walk a graph; these rules can.)

7.1 Temporal windows. minimum_delay must be less than or equal to maximum_delay.

7.2 Acyclicity.

The mechanism graph is acyclic (no Causal Relation Object may, through any chain of mechanisms, contain itself).

The refines graph is acyclic.

The MATERIALIZED subsumes and part_of graphs (Part 8.5) are acyclic; see 7.13 for how cycle attempts are handled.

7.3 Refinement.

A Causal Relation Object R that refines a parent P is a valid refinement if and only if: R's causes

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

and effects equal P's; for every field P specifies (mechanism, temporal, modality, context), R specifies the same value; and R specifies at least one field P leaves unspecified.

Anything else that names P in refines is semantically invalid as a refinement.

(A claim that DISAGREES with P is simply a separate claim - and possibly a conflict, 7.6.)

7.4 Temporal admissibility (timing as mechanism), with fixed constants.

Given a Causal Relation Object with a temporal window and an elapsed time between candidate cause and observed effect, the cause is ADMISSIBLE if and only if the elapsed time lies within [minimum_delay, maximum_delay] after both are converted to seconds using EXACTLY these constants (average Gregorian calendar values, so that every implementation computes identically):

```
    instant = 0 seconds          (a window in unit
"instant" means minimum_delay = maximum_delay = 0)
    seconds = 1
    minutes = 60
    hours   = 3,600
    days    = 86,400
    weeks   = 604,800
    months  = 2,629,746          (30.436875 days,
the average Gregorian month)
    years   = 31,556,952         (365.2425 days, the
average Gregorian year)
```

A Causal Relation Object with no temporal window imposes no timing constraint.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

This single rule is the clinician's "how long ago did you eat?" - a candidate cause excluded purely on timing.

7.5 The formal meaning of strength.

An assertion's strength about a Causal Relation Object is grounded in the do-calculus (Judea Pearl's formalization of intervention, where do(X) means X is brought about by action rather than merely observed):

• modality sufficient or contributory: strength is the source's estimate of the probability that the effects occur within the temporal window, given an intervention that brings about the causes under the stated context - P(effects within window | do(causes), context).

• modality preventive: strength is the source's estimate of the probability that the effects are PREVENTED (do not occur within the window), given the intervention, under the context.

• modality necessary: strength is the source's estimate of the probability that the effects do NOT occur when the causes are absent, under the context.

• modality enabling: strength is the source's estimate of the probability that the effects become POSSIBLE (their probability rises above zero) given the intervention under the context - an enabling cause opens a door without pushing anyone through it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Strength (a property of the world, estimated) and confidence (the source's sureness of its own assertion) are deliberately distinct fields.

7.6 Conflict, formally.

Two Causal Relation Objects A and B CONFLICT if and only if all of the following hold: • same causes: A.causes and B.causes are equal as sets of identifiers, and same effects likewise; • compatible contexts: A.context and B.context are equal, or one is a subset of the other, or either is absent; • overlapping windows: A.temporal and B.temporal overlap (after conversion by the constants of 7.4), or either is absent; • incompatible modalities: one of them is preventive and the other is necessary, sufficient, or contributory.

The modalities necessary, sufficient, contributory, and enabling are mutually COMPATIBLE (they grade the same direction of influence); preventive opposes them.

Conflicts are surfaced (Part 10), never auto-resolved: both claims stand, each with its assertions, and consumers or new evidence adjudicate.

7.7 Hierarchy consistency, pinned.

Where a Causal Relation Object P carries a mechanism, consistency is decided by reachability over the MECHANISM GRAPH, defined exactly: • Nodes: occurrent identifiers.

• Edges: for each member m of P.mechanism (each a Causal Relation Object), add a directed edge from

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

every element of m.causes to every element of m.effects.

• P is HIERARCHY-CONSISTENT if and only if, for every pair (c, e) with c in P.causes and e in P.effects, there is a directed path from c to e in that graph.

If any member of P.mechanism is not present in the store, the check is INDETERMINATE and surfaces as a dangling_reference gap rather than a failure.

A determinate failing check does not invalidate P; it raises an inconsistent_hierarchy repair signal (Part 10).

AMENDED RULE 7 (BRIDGED reachability, 2.0.0). When P and its mechanism members are pitched at DIFFERENT strata, plain reachability is too strict, because a coarse occurrent and its finer realization are the SAME happening, not two nodes on a path. The amended rule therefore folds bridges into the graph: before checking reachability, every bridge is used to MERGE its coarse occurrent with the set of finer occurrents it is identical to, so that a path may pass through a same-happening identity as freely as through a causal edge. This is what lets mechanism-consistency be checked across MULTIPLE SCALES at once - the multi-scale reasoning the type-only canon could not perform.

SKIPS AND THE CHECK. A Causal Relation Object whose skips flag is true (Part 7.14) asserts a leap across non-adjacent strata with no intervening mechanism; the bridged-reachability check treats a declared skip as a legitimate edge and does NOT raise an empty_mechanism or

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

inconsistent_hierarchy gap against it. The honest leap is a finding, not a fault.

7.8 Open-world reading.

Absence is not denial.

A claim not present in the store is unknown, not false; a field not present on a partial object is unspecified, not empty.

Negative causal knowledge is expressed positively, with the preventive modality, never by absence.

7.9 Merge.

The rules of Part 4.5 are normative: content objects are immutable and idempotent; provenance records are add-only and idempotent; replicas merge by set union.

7.10 Retraction.

A retraction is valid only if its source equals the retracted record's source, or is a key in that source's succession lineage (7.11).

A valid retraction causes DEFAULT views to exclude the retracted record - an assertion drops out of trust aggregation; an enrichment's entry drops out of the materialized view (unless another, unretracted record contributes the same entry).

The retracted record remains in history (glass-box: the record of having-contributed is never erased).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Retraction is not undone; a source that changes its mind again issues a NEW record with a later timestamp.

7.11 Succession.

Succession records form chains (K1 to K2, K2 to K3).

A key's LINEAGE is the chain closure.

Records signed by any key in a lineage attribute to the one source.

A key may name only one successor; a second succession from the same predecessor is surfaced as a conflict for trust policies to adjudicate.

Honest limit: succession protects PLANNED rotation.

If a private key is stolen, the thief holds equal cryptographic standing to name a successor; recovery from compromise is therefore a GOVERNANCE action (registry re-binding at Tier A, Part 11), not a cryptographic one.

This is a known limit of all key-based identity, stated rather than hidden.

7.12 Enrichment validity.

An enrichment record is semantically valid if and only if its field is legal for the kind of the object it enriches and its entry has the matching shape:

Field	Legal on	Entry shape

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
   aliases       occurrent,        { lang, text }
(a language-tagged
                 continuant        synonym for the
canonical label)
   participants  occurrent         a continuant
identifier (continuant:...)
   subsumes      continuant        a continuant
identifier (the is-a parent)
   part_of       continuant        a continuant
identifier (the whole)
   realized_in   realizable        an occurrent
identifier (occurrent:...)
   occurrent_subsumes  occurrent    an occurrent
identifier (the is-a parent happening-type)
   occurrent_part_of   occurrent    an occurrent
identifier (the whole happening-type)
```

The two occurrent_* fields give occurrents the same is-a and part-of discipline the continuants already had. Anything else - participants on a continuant, realized_in on an occurrent, occurrent_subsumes on a continuant, an alias entry that is a bare identifier - is semantically invalid.

7.13 Materialized acyclicity.

Stores that enforce writes (Tiers A and B) MUST reject an enrichment whose acceptance would create a cycle in the materialized subsumes or part_of graph.

Under decentralized merge (Tier C) a cycle can nevertheless arise from concurrent writes at different replicas; a conformant VIEW then breaks the cycle deterministically - exclude the cycle-completing record with the LATEST timestamp (ties

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

broken by lexicographic record identifier) - and surfaces the excluded record as a repair gap.

Determinism of the tie-break is what keeps all replicas' views identical.

7.14 Skip semantics (layer-skipping causation).

A Causal Relation Object may set skips true to assert that its cause reaches its effect across NON-ADJACENT strata WITHOUT a mechanism at the intervening levels. This is a positive claim about the world - that the causation genuinely leaps levels in one physical step (a stress hormone crossing roughly ten levels of description to bind DNA directly) - and not a confession of ignorance.

Consequences: a skipping Causal Relation Object is NOT expected to carry a mechanism, and the absence of one raises no empty_mechanism gap; the bridged-reachability check (7.7) treats the declared skip as a legitimate edge; and a rival claim that a mechanism DOES exist at the intervening levels is a normal conflict (7.6), to be settled by evidence. Negative mechanistic knowledge, like all negative causal knowledge (7.8), is thus expressed positively - here by the skips flag - never by mere absence.

PART 8. PROVENANCE, SIGNATURES, ENRICHMENT, RETRACTION, AND TRUST

8.1 Content and records.

A content object (Part 3) says WHAT is claimed or named; an assertion says WHO claims it, HOW they came to it, and HOW STRONGLY; an enrichment says WHO added WHICH word or link; a retraction says a

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

source has withdrawn its own earlier record; a succession says a source has rotated its key.

One object, any number of records; no record overwrites another.

This is Wikidata's model of multiple values with references and ranks, made explicit, cryptographic, and universal - in Version 4 there is NO datum in the commons without an author.

8.2 Source identity.

A source is a cryptographic identity: an Ed25519 public key (the Edwards-curve Digital Signature Algorithm over Curve25519, RFC 8032 - a fast, deterministic signature scheme available in every mainstream language), written ed25519:<lowercase hexadecimal>.

A human name, an institution, or a synthetic mind such as Mentova can bind a display name to its key in a registry, but the key is the identity.

Nobody can forge a record in a source's name without its private key - which is precisely why attribution lives in signed records and never in unsigned tags.

8.3 Signing.

The signature is computed over the record's identity-bearing canonical bytes (the RFC 8785 form of the record with id and signature removed - exactly the bytes that are hashed for its identifier).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Verification therefore needs nothing but the record itself and the public key inside it.

A record whose signature does not verify is rejected or quarantined (Part 11).

Because Ed25519 is deterministic, the same record signed by the same key always yields the same signature – so re-submission is exactly idempotent.

8.4 Evidence types, graded.

The evidence_type field records how the source came to the claim, in Causalontology's canonical grading: • intervention – the source acted and observed the result (do-calculus evidence; the strongest grade – acting beats watching); • observation – the source watched without intervening (correlational); • simulation – the source internally simulated the intervention, a model-based or counterfactual rollout (2.0.0; real evidence, ranked below observation and above the weaker testimony grades, and never silently counted as observation); • derivation – inferred from other claims in the store (the evidence field should cite them); • human_hint – a person supplied it as guidance (high-confidence but defeasible); • imported – carried in from an external knowledge base (typically arriving as a degenerate Causal Relation Object – the subsumption principle).

8.5 Materialized views.

A store serves each content object together with its enrichment sets, DERIVED from the enrichment records: • For each legal field of the object's

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

kind, the view is the set of entries from unretracted enrichment records about the object with that field, subject to the acyclicity rule of 7.13.

• Each entry carries its CONTRIBUTORS: the list of (source, timestamp) pairs of the records that contributed it.

Two sources contributing the same entry is one displayed entry with two contributors - CORROBORATION.

• Default views exclude retracted records (7.10) and quarantined material (Part 11); a consumer may opt into raw history knowingly.

8.6 Retraction in trust policies.

The DEFAULT policy of every conformant consumer and store view excludes records that have a valid retraction.

A consumer MAY opt into seeing retracted history (it remains in the store, glass-box), but must opt in knowingly.

Aggregations recompute without the retracted assertion; materialized views recompute without the retracted enrichment.

8.7 Succession in trust policies.

A consumer resolving a source resolves its whole lineage (7.11): trust granted to a source follows its successor keys; retractions by a successor bind the predecessor's records.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Conflicting successions (two successors named by one predecessor) freeze automatic following - the lineage is flagged and the consumer's policy or the registry decides.

8.8 Trust without forced consensus.

The store never decides who is right.

It records claims with signed assertions and lets every consumer choose a trust policy: filter by source, by evidence type, by confidence, by recency; aggregate strength estimates as it sees fit.

A RECOMMENDED default aggregation (not mandated): the confidence-weighted mean of the strength estimates from sources the consumer trusts, with intervention-grade evidence weighted above observation.

Contradictory claims coexist by design; the conflict machinery (7.6, Part 10) makes disagreement visible instead of hiding it.

PART 9. THE SHARED GLOBAL STORE: OPERATIONS, PROTOCOL, RESOLVE, AND TIERS

9.1 The abstract operation set.

Every conformant implementation (a Software Development Kit with or without a remote store) must expose these operations; the conformance vectors test them by name:

canonicalize(object) -> bytes the RFC 8785 identity-bearing bytes

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
   identify(object) -> identifier       SHA-256
over canonicalize (Part 4)
   validate_schema(object) -> report    schema
validity, with reasons
   validate_semantics(object) -> report semantic
validity (Part 7), with
                                        reasons
   admissible(cro, elapsed_seconds) -> boolean
temporal admissibility, using
                                        the
constants of 7.4
   conflicts(croA, croB) -> boolean     the
conflict test (7.6)
   hierarchy_consistent(cro, members) ->
consistent | inconsistent |

indeterminate     the 7.7 reachability
                                        check
   sign(record, private_key) -> record  attach an
Ed25519 signature
   verify(record) -> boolean            check the
signature (8.3)

   And, against a store:
   put(object) -> identifier            write a
content object; idempotent
   put_record(record) -> identifier     write an
assertion, enrichment,

retraction, or succession; signature
                                        verified;
idempotent
   get(identifier) -> object with materialized
enrichments (8.5)
   assertions_about(identifier) -> page of
assertions
   enrichments_about(identifier) -> page of
enrichment records
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
   retractions_of(identifier) -> page of
retractions
   lineage(source_key) -> the succession chain
closure
   resolve(text, lang) -> page of candidates (9.4)
   query(pattern) -> page of objects    (9.3)
   gaps(near, kinds) -> page of gaps    the
stigmergy read (Part 10)
```

9.2 The Hypertext Transfer Protocol binding (Tier A).

The reference store speaks plain HTTP with JSON bodies:

POST /objects	write a content object; the server recomputes the identifier from the body and rejects a mismatch; idempotent
GET /objects/{id}	fetch an object WITH its materialized enrichment view (entries and contributors, 8.5); add ?view=raw for the pure object only, or ?view=history to include retracted records, marked
POST /records	write an assertion, enrichment, retraction, or succession; the server verifies the Ed25519 signature; unsigned or unverifiable records are quarantined per Part 11; enforcing tiers also

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
                                    apply 7.12 and 7.13
at this point
   GET  /records/{id}
   GET  /assertions?about={id}
   GET  /enrichments?about={id}
   GET  /retractions?about={record_id}
   GET  /successions?key={ed25519_key}
   GET  /resolve?text=...&lang=..     alias
resolution (9.4)
   POST /query                      the query-by-example
document of 9.3; stores
                                    SHOULD also expose a
SPARQL endpoint at /sparql
                                    (SPARQL is the
SPARQL Protocol and RDF Query
                                    Language, the
standard query language for
                                    Resource Description
Framework data)
   GET  /gaps?near={id}&kind={kind}   the
stigmergy read
   GET  /conflicts?near={id}          surfaced
contradictions

   PAGINATION, on every endpoint that returns a
list: the request takes limit
   (default 100, maximum 1,000) and cursor (an
opaque continuation token from
   the previous page); the response body carries
items and next_cursor (null
   on the last page). Cursors are valid for at
least one hour.

   AUTHENTICATION: an account bearer token for
write access at Tier A, IN
   ADDITION to the per-record signatures - the
token controls resource use;
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

the signature carries the trust. Rate limits per account and per source key.

9.3 The query language (normative minimum).

A query is a JSON document - query-by-example over one kind, with equality and containment filters:

```
{
  "kind":  "causal_relation_object",
one of the twenty-one kinds
  "where": {
    "causes_contains":  "occurrent:...",
set-containment filters
    "effects_contains": "occurrent:...",
    "modality":         "sufficient",
equality filters
    "refines":
"causal_relation_object:...",
    "about":            "occurrent:...",
for provenance kinds
    "source":           "ed25519:...",
    "is_partial":       true,
missing >= 1 optional field
    "missing":          "temporal"         a
named field is unspecified
  },
  "limit": 100, "cursor": null
}
```

Every conformant store MUST implement this query form; SPARQL, full-text search, and richer graph patterns are SHOULD-level extensions. This minimum is deliberately small: it is exactly enough to power resolution, the gap

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

queries, provenance lookup, and claim lookup - the operations conformance
depends on.

9.4 Resolve, with a conformance minimum.
resolve(text, lang) maps words to vocabulary.

The NORMATIVE MINIMUM, tested by vector: • Canonical-label match: an occurrent or continuant whose label equals the query text after canonical normalization (lowercase; runs of whitespace replaced by single underscores) MUST be returned.

• Alias match: an item having an unretracted alias enrichment whose text equals the query text case-insensitively (and whose lang matches the query language, when one is given) MUST be returned.

• Ranking: canonical-label matches first, then alias matches, then anything else.

Fuzzier matching (stemming, edit distance, embeddings) is PERMITTED and implementation-defined, but must rank after the exact matches above.

The REQUIRED resolve-before-mint workflow (Part 4.2) depends only on the minimum, so it behaves identically on every conformant store.

9.5 The store is a knowledge graph.

Reified causal objects map naturally onto the Resource Description Framework with RDF-star (statements about statements - exactly what a Causal Relation Object is), queryable with SPARQL.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The Wikidata precedent proves the pattern at the scale of over one hundred million items edited by the whole world, every statement carrying references.

9.6 Deployment tiers - adopt the first, keep the options: • Tier A, central hosted service: one graph store behind the HTTP binding.

Simplest; single source of truth.

START HERE.

• Tier B, federated: many store nodes synchronizing by push and pull, in the manner of the Git version-control system; contributors can run local stores.

• Tier C, decentralized: content-addressed objects in a Merkle Directed Acyclic Graph (a hash-linked, tamper-evident structure of the kind used by Git and the InterPlanetary File System, IPFS).

Version 4 makes the CRDT property exact: immutable objects plus add-only signed records merge by set union in any order, and even the acyclicity edge case has a deterministic view rule (7.13).

Because identity, merge, and signed records are in the data model from day one, a deployment can begin at Tier A and migrate to Tier C without changing the data model.

PART 10. STIGMERGY: THE STORE GUIDES THE NEXT CONTRIBUTION

Stigmergy is coordination through the environment: agents leave traces in a shared medium, and the

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

state of the medium guides what other agents do next (an ant trail; a half-built wall that shows where the next brick goes).

The most distinctive property of the Causalontology commons is that its stigmergy needs no separate mechanism - the data structure's own partiality IS the signal.

The gap kinds, each a first-class, queryable fact (returned by the gaps operation with kind tags):

- missing_field.

A partial Causal Relation Object lacking its mechanism, temporal window, modality, or context - a literal "fill me in" marker.

The subsumption principle guarantees a steady supply: every claim imported from an external knowledge base arrives as a degenerate object.

Filling one is done with a refines object (4.6), and once a valid refinement exists the parent leaves the gap list - the gap visibly closes.

- dangling_reference.

A Causal Relation Object whose cause or effect names an occurrent nobody has defined, a realizable whose bearer is an undefined continuant, or a mechanism member absent from the store (7.7) - the red link that says "this page is wanted".

- empty_mechanism.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

A coarse relation not yet decomposed into finer ones - an invitation to explain HOW.

- inconsistent_hierarchy.

A mechanism that fails the reachability check of 7.7, or a record excluded by the cycle-breaking rule of 7.13 - an invitation to repair.

- conflict.

Two claims that satisfy the formal conflict test (7.6) - an invitation to bring new evidence, ideally intervention-grade.

- demand_supply.

A region of the graph that many queries traverse but where claims are sparse, unrefined, supported only by low-grade evidence, or thinned by retractions - the computed version of "most-requested articles": high demand, low supply, invest here.

Delivery: the gaps endpoint answers "give me the most valuable gaps near this topic"; a contribution dashboard renders the same signals for people; and an optional reputation or bounty layer can reward closures - and because every alias, link, claim, and refinement is a signed record, reputation can now be computed over EVERYTHING a source has contributed, not only its assertions.

Contributors - individual developers, laboratories, institutions - are guided by the emergent shape of the commons itself.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

And the endgame that makes Causalontology singular: gap-signals are just data, so autonomous minds read them too.

Mentova – or any Causalontology-conformant agent – can query the frontier, pick the most valuable gap, learn or reason out the missing structure, and contribute it back as a refines object with a signed, intervention-grade assertion.

The commons is then grown by human beings and synthetic minds together, guided by the same visible frontier.

PART 11. SAFETY: ABUSE RESISTANCE, CLAIMS OF CONSEQUENCE, TAKEDOWN BY TIER

A world-writable commons of CAUSAL claims carries higher stakes than a commons of trivia, and will attract the abuses every open system attracts.

The defenses are layered and, as of Version 4, uniform – because every datum has a signed author:

11.1 Abuse resistance.

• Forgery: impossible without a source's private key – every record of every kind is signed (8.2, 8.3).

The rejected "contributed_by" shortcut is the cautionary tale: unsigned attribution would have enabled framing (adding a poisoned entry in someone else's name); signed records make framing impossible.

• Spam and Sybil attacks (one attacker manufacturing many identities): keys are free to

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

mint, so trust NEVER attaches to key-count; it attaches to a consumer's chosen trust policy over specific sources (8.8).

Tier A adds account-based rate limits; reputation systems, where used, weight by history and corroboration, never by volume.

• Data poisoning (deliberately false claims, including claims aimed at misleading the artificial-intelligence agents that read the commons): signed provenance makes poison attributable; retraction gives honest sources a clean exit while the poisoner's history remains pinned to its key; consumers filtering by trusted sources and intervention-grade evidence are structurally resistant; conflicts surface rather than silently average.

• Vandalism of vocabulary (bad aliases, wrong taxonomy links): as of Version 4 every enrichment is a signed record, so vandalism is ATTRIBUTABLE AT EVERY TIER, suppressible by source key, and - when the author repents - removable by the author's own retraction.

Grow-only views mean vandalism can never DESTROY good data, only add bad data that policy filters out.

• Unsigned or unverifiable contributions: accepted, if at all, only into a clearly-marked QUARANTINE visibility tier that default queries exclude.

11.2 Claims of consequence.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Some causal claims can hurt people if believed wrongly - health and medical claims ("X cures Y"), safety-critical procedures, legal and financial claims.

Policy, from launch: • Consequential domains are tagged (a category discipline on the occurrents involved: health, safety, legal, financial).

• A consequential claim is served only with its provenance and confidence displayed - never as a bare statement of fact.

• Records on consequential claims MUST be signed; unsigned ones are quarantined without exception.

• Every interface that serves consequential claims carries a standing notice that the commons records who claims what, with what evidence - it does not dispense medical, legal, or financial advice.

11.3 Takedown, honestly, tier by tier.

"Remove" means different things at each deployment tier, and the specification says so rather than promising what content-addressing cannot deliver: • Tier A (central): the operator can hard-delete an object or record from the serving store.

Every takedown is itself publicly logged (actor, target identifier, ground) - the commons' equivalent of the constitutional guard's append-only veto log in the PrologAI Branch.

• Tier B (federated): the origin node deletes locally and publishes a signed SUPPRESSION notice on the synchronization channel; peer nodes SHOULD honor it; compliance is per-node policy and

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

visible (a node's suppression behavior is auditable).

• Tier C (decentralized): copies cannot be forcibly deleted from machines the network does not control.

Takedown means a signed TOMBSTONE record that all conformant default views honor by suppression, plus the public log.

This is the honest limit of decentralization, stated in advance.

11.4 Privacy.

The commons is for GENERAL causal knowledge (type-level, Part 3).

Claims about identifiable private individuals are out of scope, removable under 11.3, and - because the standard models types, not tokens - largely excluded by construction: there is no object kind for "what happened to this particular person on this particular day".

Contributors in jurisdictions with data-protection law (for example the European Union's General Data Protection Regulation, GDPR) should note that Tier A operators can honor erasure requests fully, and that Tier C operators can honor them only in the tombstone sense of 11.3 - a fact a deployer must weigh before choosing a tier.

PART 12. LICENSING: CODE, SPECIFICATION, AND DATA

Code and data are different things and carry different licenses:

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• The specification and all reference code: the Apache License 2.0 - permissive, patent-protective, and matching the PrologAI substrate.

• The DATA in the commons: Creative Commons Zero (CC0) - the public-domain dedication Wikidata uses.

Contributors dedicate their claims and records to the public domain on contribution, so the commons is maximally reusable and no downstream user needs to track thousands of attribution chains.

(Attribution survives anyway - not as a legal obligation but as provenance, which as of Version 4 is built into every datum.)

• External content imported under incompatible licenses is not imported; share-alike sources are linked, not copied.

PART 13. GOVERNANCE AND VERSIONING

• The specification is versioned with Semantic Versioning (a three-part MAJOR.MINOR.PATCH scheme in which a change to the first number signals an incompatible change).

Every binding and every store declares the specification version it implements.

• THE ENUMERATION RULE (new in Version 4, ending an ambiguity): a closed enumeration - the occurrent categories, the continuant categories, the modalities, the evidence types, the enrichment fields - is part of the validator contract and,

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

for the vocabulary categories, part of the identity guarantee itself.

EXTENDING ANY CLOSED ENUMERATION IS A MAJOR VERSION CHANGE, never MINOR, even though old data round-trips: an old validator would reject new data, and identity computed over new categories would not be comparable.

(Adding a wholly NEW object kind is likewise MAJOR for validators that enforce closed kind lists; this is exactly how the 2.0.0 release added the token tier and the stratum, bridge, port, conduit, and quality kinds - batched together as a single MAJOR step.)

• The conformance suite is versioned with the specification; "conformant" always means "conformant to a named version".

• Every binding is gated in Continuous Integration: a release that does not pass the vectors of its declared version is not published.

• GOVERNANCE.md defines a lightweight change process in the spirit of a W3C or IETF working group: a written proposal, review in the open, a conformance impact statement (which vectors change), and a decision recorded in the repository.

• The takedown procedure of Part 11.3 is part of governance; every takedown and every registry re-binding after key compromise (7.11) is publicly logged.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

PART 14. THE CONFORMANCE SUITE (ONE HUNDRED AND THIRTY-SEVEN VECTORS)

Each vector is a named test case with an input and an expected result; in the repository each is one file under conformance/vectors/.

An implementation is conformant if and only if it produces the expected result for every vector.

As of specification 4.0.0 the suite has ONE HUNDRED AND THIRTY-SEVEN vectors, V01 through V137. This listing is GENERATED from the machine vector files, so every vector number, name, group, and expected result below matches conformance/vectors/ exactly; each vector's full input, with its real sixty-four-character identifiers, is in its own file. Vectors V01 through V38 are the re-frozen original suite; V39 onward exercise the twenty-one-kind model in full - the token tier, the ordinal tick unit, the cross_stratal_seam, the conduit realized_by, and the attitude, predicted_occurrence, and prediction_error kinds (the last three in groups X, Y, and Z; the letter V is never used as a group letter).

GROUP A - SCHEMA VALIDITY

  V01 complete_cro_is_valid
      expect: schema_valid true, semantically_valid true
      note: A complete Causal Relation Object carries no strength and no provenance fields; both live in assertions.

  V02 degenerate_cro_is_valid_and_partial

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      expect: schema_valid true,
semantically_valid true, is_partial true, missing
[mechanism, temporal, modality, context]
      note: The subsumption case: an imported bare
edge as a degenerate object - a valid
            partial and a stigmergic gap.

  V03 missing_effects_is_invalid
      expect: schema_valid false, reason required
property 'effects' missing

  V04 empty_causes_is_invalid
      expect: schema_valid false, reason causes
must have at least one item

  V05 unknown_modality_is_invalid
      expect: schema_valid false, reason modality
not in enumeration

  V06 additional_property_is_invalid
      expect: schema_valid false, reason
additional property 'colour'
      note: Also rejects any attempt to put
aliases or other enrichment state inside a
            content object.

  V07 free_text_cause_is_invalid
      expect: schema_valid false, reason cause
must match ^occ:[0-9a-f]{64}$
      note: Words live on enrichment records, not
in claims.

  V08 occurrent_is_valid_and_pure
      expect: schema_valid true
      note: Nothing else is legal on an occurrent
- its aliases arrive as signed enrichment
            records.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V09 occurrent_missing_label_is_invalid
      expect: schema_valid false, reason required
property 'label' missing

  V10
occurrent_category_not_in_enumeration_is_invalid
      expect: schema_valid false, reason category
not in enumeration
      note: The closed enumeration makes
vocabulary identity deterministic; 'charge' would
            also be rejected in review as an
unqualified homonym.

  V11 assertion_is_valid
      expect: schema_valid true

  V12 assertion_confidence_out_of_range_is_invalid
      expect: schema_valid false, reason
confidence must be <= 1

  V13 enrichment_is_valid
      expect: schema_valid true,
semantically_valid true

GROUP B - SEMANTIC VALIDITY

  V14
reversed_temporal_window_is_semantically_invalid
      expect: schema_valid true,
semantically_valid false, reason dmin must be <=
dmax

  V15 cyclic_mechanism_is_semantically_invalid
      expect: schema_valid true,
semantically_valid false, reason mechanism must be
acyclic

  V16 self_refinement_is_semantically_invalid
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

expect: schema_valid true, semantically_valid false, reason refines must be acyclic

V17 refinement_changing_parent_field_is_not_a_refinement
expect: valid_refinement false, reason a refinement may not change a field the parent specified; this is a rival claim

V18 enrichment_field_illegal_for_kind_is_invalid
expect: schema_valid true, semantically_valid false, reason participants is not a legal field for a continuant (semantics rule 12)

V19 enrichment_entry_shape_mismatch_is_invalid
expect: semantically_valid false, reason an aliases entry must be a language-tagged text object

V20 subsumes_cycle_is_rejected_or_broken_deterministically
expect: enforcing_tier write rejected: would create a cycle in the materialized subsumes graph, decentralized_merge the conformant view excludes the cycle-completing record with the latest timestamp (ties by lexicographic record identifier) and surfaces it as a repair gap

GROUP C - TEMPORAL ADMISSIBILITY

V21 elapsed_time_inside_window_is_admissible
expect: admissible true

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V22
elapsed_time_outside_window_is_not_admissible
      expect: admissible false
      note: Together with V21: the clinician's
'how long ago did you eat?' - a candidate
            cause excluded purely on timing.

  V23 unit_conversion_is_fixed
      expect: admissible true
      note: 12 months = 31,556,952 s and 14 months
= 36,816,444 s by the fixed constants
            (month = 2,629,746 s). An
implementation using a different month length
fails
            this vector - which is the point.

GROUP D - IDENTITY, CANONICALIZATION, AND MERGE

  V24 key_order_does_not_change_identity
      expect: equal_identifiers true

  V25 number_formatting_does_not_change_identity
      expect: equal_identifiers true
      note: Guaranteed by RFC 8785 number
serialization.

  V26 identical_put_is_idempotent
      expect: objects_in_store 1, second_put
returns the same identifier and changes nothing

  V27 same_entry_two_sources_is_corroboration
      expect: enrichment_records 2,
materialized_alias_entries 1,
contributors_on_entry 2
      note: Corroboration, not duplication - and
each word has its authors.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

GROUP E - PROVENANCE, SIGNATURES, RETRACTION, SUCCESSION

V28 two_sources_one_claim_is_one_object_two_assertions
expect: causal_relation_objects 1, assertions_about_it 2, overwrites 0

V29 valid_signature_verifies
expect: verify true

V30 tampered_record_fails_verification
expect: verify false

V31 retraction_excludes_assertion_from_default_views
expect: default_view excludes the retracted assertion, history_view includes it, marked retracted, foreign_retraction a retraction signed by a source outside the lineage is semantically invalid and has no effect

V32 author_retracts_own_enrichment
expect: materialized_view no longer contains Bob's entry (unless another unretracted record contributes the same entry), history_view the record survives, marked retracted

V33 key_succession_preserves_source_lineage
expect: lineage_of_K2_contains_K1 true, retraction_valid true, trust granted to the lineage covers records under both keys

GROUP F - CONFLICT, HIERARCHY, RESOLVE, STIGMERGY

V34 preventive_conflicts_with_sufficient
expect: conflicts true

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

V35 contributory_does_not_conflict_with_sufficient
expect: conflicts false

V36 hierarchy_consistency_is_reachability
expect: with_m1_and_m2 consistent (path A -> B -> C exists), with_m2_replaced_by_D_to_C inconsistent; P surfaces as an inconsistent_hierarchy gap, with_m2_absent_from_store indeterminate; P surfaces as a dangling_reference gap

V37 resolve_minimum_is_exact_then_alias
expect: resolve('Press The Button', en) returns the occurrent (alias match, case-insensitive), resolve('press_button', en) returns it ranked first (canonical-label match), fuzzy_extras rank after both

V38 refined_parent_leaves_the_gap_list
expect: before_R gaps(kind missing_field) includes P, after_R gaps excludes P; R does not appear (it is complete)
note: The gap visibly closes - the heart of the stigmergy.

GROUP G - STRATUM

V39 valid_stratum
expect: schema_valid true
note: A stratum with scheme and ordinal is valid; lower ordinal is finer.

V40 stratum_without_scheme_is_invalid
expect: schema_valid false, hard true
note: scheme is REQUIRED (N3.1.1). HARD.

V41 two_strata_same_ordinal_same_scheme_is_valid

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      expect: schema_valid true
      note: A legitimate disagreement (two names
for one level); MUST NOT be rejected
            (N3.1.2).

  V42
cross_scheme_ordinal_comparison_is_scheme_mismatch
      expect: classification scheme_mismatch, hard
true
      note: Comparing ordinals across schemes is
forbidden (Rule 13). HARD.

  V43 noncontiguous_ordinals_are_valid
      expect: schema_valid true
      note: Ordinals need not be contiguous
(N3.1.3).

GROUP H - OCCURRENT STRATIFICATION

  V44 stratified_occurrent_is_valid
      expect: schema_valid true,
semantically_valid true
      note: An occurrent may carry an optional
identity-bearing stratum.

  V45
unstratified_occurrent_is_valid_and_surfaces_invit
ation
      expect: schema_valid true, gap
unstratified_occurrent, hard false
      note: The null stratum is legal (N3.3.3); an
invitation, not an error.

  V46
same_label_different_stratum_yields_different_iden
tity
      expect: distinct_identities true
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: Proves N3.3.2: stratum is identity-
bearing.

GROUP I - BRIDGE

  V47 valid_bridge_constitutes
      expect: schema_valid true, wellformed true
      note: constitutes: the fine occurrents ARE
the coarse one.

  V48 valid_bridge_aggregates
      expect: schema_valid true, wellformed true
      note: aggregates: a statistical/summary
property of interchangeable fine occurrents.

  V49 valid_bridge_realizes
      expect: schema_valid true, wellformed true
      note: realizes: multiple realizability; a
different fine set could implement the
            coarse one.

  V50 valid_bridge_supervenes_on
      expect: schema_valid true, wellformed true
      note: supervenes_on: the weakest relation;
the honest choice when the story is
            unknown.

  V51
bridge_with_coarse_ordinal_not_greater_is_malforme
d
      expect: wellformed false, gap
malformed_bridge, hard true
      note: N3.2.1(e): coarse ordinal MUST be
strictly greater than fine. HARD.

  V52
bridge_with_fine_members_in_different_strata_is_ma
lformed
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      expect: wellformed false, gap
malformed_bridge, hard true
      note: N3.2.1(c): all fine members share one
stratum. HARD.

  V53 bridge_cycle_is_invalid
      expect: gap bridge_cycle, hard true
      note: The bridge graph MUST be acyclic
(N3.2.3). HARD.

  V54 bridge_across_schemes_is_malformed
      expect: wellformed false, gap
malformed_bridge, hard true
      note: N3.2.1(d): coarse and fine strata must
share a scheme. HARD.

  V55
two_bridges_one_coarse_different_fine_are_both_val
id
      expect: both_valid true
      note: Multiple realizability is expected,
not a contradiction (N3.2.5).

GROUP J - BRIDGED REACHABILITY

  V56
coarse_causal_relation_object_with_fine_mechanism_
and_bridges_is_consistent
      expect: consistency consistent
      note: Amended Rule 7 / Algorithm B: the
coarse claim is mechanized finely, bridges
            carrying the translation.

  V57
same_without_bridges_is_inconsistent_hierarchy
      expect: consistency inconsistent, gap
inconsistent_hierarchy
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: Without bridges the coarse endpoints
never reach the fine graph.

  V58
literal_reachability_only_implementation_fails_V56
      expect: literal_consistency_not consistent,
bridged_consistency consistent
      note: NEGATIVE VECTOR. A literal-
reachability implementation fails V56; only
bridged
            reachability passes. This is what V58
exists to catch (N12.2.4).

GROUP K - CLASSIFICATION AND SKIPS

  V59
intra_stratal_causal_relation_object_classified_co
rrectly
      expect: classification intra_stratal
      note: Algorithm C: all causes and effects
share one stratum.

  V60
adjacent_stratal_causal_relation_object_classified
_correctly
      expect: classification adjacent_stratal
      note: Algorithm C: ordinals differ by
exactly 1.

  V61
skipping_causal_relation_object_classified_correct
ly
      expect: classification skipping
      note: Algorithm C: ordinals differ by more
than 1.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V62
skipping_causal_relation_object_empty_mechanism_sk
ips_absent_surfaces_incomplete_mechanism
      expect: gaps [incomplete_mechanism]
      note: HALF OF THE ASYMMETRY (Algorithm D).
skips ABSENT + skipping + empty mechanism
            => incomplete_mechanism.

  V63
skipping_causal_relation_object_empty_mechanism_sk
ips_true_surfaces_NOTHING
      expect: gaps []
      note: THE OTHER HALF. skips TRUE + skipping
+ empty mechanism => NO GAP. The absence
            of mechanism is a FINDING. V62/V63
justify Proposal 7.

  V64
skips_true_with_mechanism_is_contradictory_skip
      expect: gaps [contradictory_skip], hard true
      note: Rule 16 clause 1. HARD.

  V65
skips_true_on_intra_stratal_causal_relation_object
_is_vacuous_skip
      expect: gaps [vacuous_skip]
      note: Rule 16 clause 2. Invitation.

  V66 skips_false_is_distinct_from_skips_absent
      expect: distinct_identities true
      note: N12.4.2: false is a positive claim;
absent makes no claim. Different bytes,
            different hashes.

  V67 mixed_stratal_endpoints_surfaces_warning
      expect: gap mixed_stratal_endpoints, hard
false
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: Causes or effects span multiple
strata; legal but usually an authoring error.

GROUP L - MODALITY

  V68 enabling_modality_is_valid
      expect: schema_valid true
      note: The fifth modality: causes make
effects POSSIBLE without making them occur.

  V69 enabling_is_compatible_with_sufficient
      expect: conflicts false
      note: Rule 6 amended: enabling is compatible
with the positive modalities.

  V70 enabling_is_opposed_by_preventive
      expect: conflicts true
      note: enabling and preventive are contraries
(one removes a barrier, the other erects
            one).

GROUP M - PORT AND CONDUIT

  V71 valid_port
      expect: schema_valid true
      note: A typed interface borne by a
continuant.

  V72
two_ports_same_bearer_differing_only_in_label_yiel
d_distinct_identities
      expect: distinct_identities true
      note: label is identity-bearing (N4.1.1);
this is the pattern that proves the
            realizable collision is repaired.

  V73 valid_transmissive_conduit_no_transform
      expect: schema_valid true, wellformed true
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: transform ABSENT: a pipe. A perfect
wire would substitute.

  V74 valid_computational_conduit_with_transform
      expect: schema_valid true, wellformed true
      note: transform PRESENT: a computer. It
possesses a faculty resident in neither
            endpoint.

  V75
conduit_carrying_unaccepted_occurrent_is_malformed
      expect: wellformed false, gap
malformed_conduit, hard true
      note: N4.2.1(c/d). HARD.

  V76 conduit_from_an_in_port_is_malformed
      expect: wellformed false, gap
malformed_conduit, hard true
      note: N4.2.1(a): from must be out or
bidirectional. HARD.

  V77
computational_conduit_may_emit_what_it_did_not_acc
ept
      expect: wellformed true
      note: Proves N4.2.2: a computational conduit
may emit its transform's effects.

GROUP N - REALIZABLE REPAIR

  V78
two_labelled_realizables_same_bearer_same_kind_yie
ld_distinct_identities
      expect: distinct_identities true
      note: The defect repair (4.3): with label
identity-bearing, a bearer may bear
            arbitrarily many realizables.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V79
unlabelled_realizable_remains_valid_and_hash_ident
ical_to_1_0_0
      expect: schema_valid true, identity_stable
true
      note: An unlabelled realizable keeps its
1.0.0 whole-word hash (label absent is not
            hashed).

GROUP O - OCCURRENT TAXONOMY AND MEREOLOGY

  V80 occurrent_subsumes_is_valid
      expect: semantically_valid true
      note: fires_action_potential is a KIND OF
fires (occurrent_subsumes).

  V81 occurrent_subsumes_cycle_is_invalid
      expect: gap occurrent_hierarchy_cycle, hard
true
      note: N6.1.1. HARD.

  V82 occurrent_part_of_is_valid
      expect: semantically_valid true
      note: chew is PART OF eat
(occurrent_part_of).

  V83 occurrent_part_of_does_not_imply_causation
      expect: implies_causation false
      note: N6.1.3: chew is part of eat; chew does
not CAUSE eat.

  V84 subsumption_across_strata_surfaces_warning
      expect: gap subsumption_crosses_strata, hard
false
      note: N6.1.4: cross-stratal subsumption is
almost always a misuse where a Bridge is
            meant.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
GROUP P - TOKEN TIER

  V85 valid_token_individual
      expect: schema_valid true
      note: THIS hippocampus, in THIS patient.

  V86 individual_without_instantiates_is_invalid
      expect: schema_valid false, hard true
      note: No free-floating tokens (P1 / Rule
18). HARD.

  V87
two_individuals_same_type_different_designators_yi
eld_distinct_identities
      expect: distinct_identities true
      note: designator is identity-bearing.

  V88 valid_token_occurrence
      expect: schema_valid true
      note: One instance of an occurrent type, at
absolute time.

  V89
token_interval_instantaneous_vs_ongoing_vs_bounded
_are_distinguishable
      expect: three_distinct_identities true
      note: N5.2.2: end present = bounded; end
absent = instantaneous; open true = ongoing.

  V90 token_with_roles_is_valid
      expect: schema_valid true
      note: Thematic roles (agent, patient,
instrument, ...) at token level (N5.2.3).

  V91 valid_quality_quantity_with_ucum_unit
      expect: schema_valid true
      note: A property type with a canonical UCUM
unit.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V92 valid_state_assertion_quantity
      expect: schema_valid true, state_gaps []
      note: THIS individual bore THIS quantity
value over THIS interval.

  V93 valid_state_assertion_categorical
      expect: schema_valid true, state_gaps []
      note: A categorical value.

  V94 valid_state_assertion_boolean
      expect: schema_valid true, state_gaps []
      note: A boolean value.

  V95
state_value_type_mismatching_quality_datatype_is_i
nvalid
      expect: state_gaps [value_type_mismatch],
hard true
      note: N5.3.1 / Rule 19. HARD.

  V96
state_unit_mismatching_quality_unit_is_invalid
      expect: state_gaps [unit_mismatch], hard
true
      note: N5.3.2 / Rule 19. HARD.

  V97 valid_token_causal_claim_with_covering_law
      expect: schema_valid true
      note: THIS particular causing is an instance
of THAT general law (N5.4.1).

  V98
token_causal_claim_with_no_covering_law_is_VALID_a
nd_surfaces_uncovered_causation
      expect: schema_valid true, gap
uncovered_causation, hard false
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: The most valuable gap: something
happened and we do not know why (N5.4.2).

  V99
token_causal_claim_actual_delay_within_covering_la
w_window_is_consistent
      expect: delay_within_window true
      note: Algorithm E: observed delay falls
within the law's temporal window.

  V100
token_causal_claim_actual_delay_outside_window_sur
faces_delay_outside_window
      expect: delay_within_window false, gap
delay_outside_window, hard false
      note: Either the law is wrong or the
instance is misclassified; both are science
            (N5.4.3).

  V101
token_causal_claim_with_cause_after_effect_is_retr
ocausal_and_invalid
      expect: retrocausal true, hard true
      note: Rule 21: causes do not follow their
effects. HARD.

  V102
token_causal_claim_whose_tokens_do_not_instantiate
_its_covering_law_surfaces_mismatch
      expect: gap covering_law_mismatch, hard
false
      note: Rule 20: the instance does not appear
to be an instance of the law it cites.

GROUP Q - PROVENANCE

  V103 assertion_about_a_token_is_valid
      expect: schema_valid true
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: about widened to the token kinds
(6.1).

  V104
assertion_with_evidenced_by_citing_interventional_
tokens
      expect: schema_valid true,
identity_includes_evidenced_by true
      note: The evidence hierarchy acquires a
mechanism (6.2).

  V105
evidence_type_simulation_is_valid_and_ranks_below_
observation
      expect: schema_valid true,
ranks_below_observation true
      note: simulation is real evidence but ranks
below observation, which ranks below
            intervention (6.2).

GROUP R - BACKWARD COMPATIBILITY

  V106
every_whole_word_1_0_0_record_retains_its_identity
_hash_under_2_0_0
      expect: identity_stable true, all_whole_word
true
      note: The formal proof of compatibility
within the whole-word baseline: replay all 38
            re-frozen inputs; every identity
procedure is deterministic and whole-word.

GROUP S - WHOLE-WORD NAMING (P7)

  V107
abbreviated_scheme_is_rejected_only_whole_word_sch
eme_is_valid
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      expect: abbreviated_rejected true,
whole_word_valid true, hard true
      note: NEGATIVE VECTOR. An abbreviated scheme
(the former cro: / str: form) fails
            schema validation; only the whole-word
scheme validates. Machine-enforces the
            re-mint. HARD.

GROUP T - ORDINAL TICK TEMPORAL UNIT (3.0.0)

  V108 tick_temporal_window_is_valid
      expect: schema_valid true,
semantically_valid true
      note: 3.0.0 Change One: a Causal Relation
Object whose temporal window is in ticks is
            schema- and semantically-valid; ticks
is an additive enum value.

  V109 tick_admissibility_is_integer_ordered
      expect: in_window_admissible true,
out_of_window_admissible false
      note: 3.0.0 Change One: a tick window is
ordered by INTEGER comparison of tick
            counts; an elapsed tick count inside
[min,max] is admissible, outside is not -
            no wall-clock mapping is used.

  V110 tick_and_wallclock_are_disjoint_dimensions
      expect: tick_delay_within_wallclock_window
false, tick_and_wallclock_windows_overlap false,
to_seconds_of_ticks_refused true
      note: 3.0.0 Change One: an ordinal (tick)
unit and a wall-clock unit are DIFFERENT
            DIMENSIONS - a tick delay is never
within a wall-clock window, their windows
            never overlap, and converting ticks to
seconds is a refused category error.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V111
tick_unit_is_identity_bearing_wallclock_id_unchang
ed
      expect: tick_id_differs_from_seconds_id
true, wallclock_id_unchanged_from_2_0_0 true
      note: 3.0.0 Change One: the temporal unit is
identity-bearing, so a tick-unit CRO has
            a distinct identifier from an
otherwise-identical seconds-unit CRO; and an
            existing wall-clock CRO's identifier
is unchanged under 3.0.0 (additive,
            identity-preserving).

GROUP U - MANAGED CROSS-STRATAL SEAM (3.0.0)

  V112 cross_stratal_seam_unmodeled_is_valid
      expect: schema_valid true,
semantically_valid true, wellformed true
      note: 3.0.0 Change Two: a Cross Stratal Seam
over NON-adjacent endpoints with
            mechanism_status 'unmodeled' is
schema-valid, semantically valid, and
            well-formed (the eighteenth kind).

  V113 seam_mechanism_status_is_identity_bearing
      expect: absent_is_valid true,
unmodeled_and_absent_have_distinct_identity true
      note: 3.0.0 Change Two: mechanism_status is
identity-bearing - a seam declaring
            'absent' (no intervening mechanism
exists) has a DISTINCT identifier from one
            declaring 'unmodeled' (a mechanism
exists but is not drawn). This is exactly
            the honest-ignorance distinction the
bare skips boolean could not make.
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

V114 seam_drawn_chain_wellformed_and_absent_chain_refused
expect: drawn_chain_wellformed true, absent_with_chain_refused true
note: 3.0.0 Change Two: a seam that DRAWS a chain of adjacent-stratum steps spanning its endpoints is well-formed; a seam carrying a chain but mechanism_status 'absent' is refused as contradictory_seam (a drawn mechanism is not absent), both by local semantics and by seam well-formedness.

V115 seam_home_is_the_coarsest_stratum
expect: home_is_max_ordinal_stratum true
note: 3.0.0 Change Two: THE HOME RULE - a Cross Stratal Seam belongs to the coarsest (greatest-ordinal) stratum it touches, so a layer-to-stratum binding can place and check it.

V116 adjacent_seam_is_malformed_and_seam_is_new_identity_space
expect: adjacent_endpoints_malformed true, co_stratal_endpoints_malformed true, seam_scheme_is_new true
note: 3.0.0 Change Two: a seam over ADJACENT (ordinal gap 1) or CO-stratal endpoints is malformed (a seam is for non-adjacent strata; use a Causal Relation Object or Bridge otherwise); and cross_stratal_seam is a new identity scheme that disturbs no existing record.

GROUP W - REALIZED_BY REFERENCE (3.0.0)

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V117 conduit_realized_by_is_valid
      expect: schema_valid true
      note: 3.0.0 Change Three: a conduit carrying
a realized_by reference (naming, by
            identity, the native law or signal
that realizes its transform) is
            schema-valid; it is invalid under
2.0.0 (additionalProperties false), correctly
            a 3.0.0 feature.

  V118
realized_by_is_identity_bearing_absent_id_unchange
d
      expect: realized_by_id_differs_from_unbound
true, unbound_conduit_id_unchanged_from_2_0_0 true
      note: 3.0.0 Change Three: realized_by is
identity-bearing, so a conduit that names it
            has a distinct identifier from one
that does not; and a conduit that carries no
            realized_by keeps its 2.0.0 identifier
byte-for-byte (additive,
            identity-preserving).

  V119
realized_by_unbound_is_legal_malformed_reference_r
ejected
      expect: unbound_is_valid true,
malformed_reference_rejected true
      note: 3.0.0 Change Three: ABSENT (unbound)
realized_by is legal - a transmissive or
            computational conduit need not name a
native realizer; a realized_by that is
            not a scheme-qualified identifier is
schema-rejected.

GROUP X - PREDICTION AND PREDICTION ERROR (4.0.0)
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
  V120
predicted_occurrence_with_tick_window_is_valid
      expect: schema_valid true,
semantically_valid true,
id_scheme_is_predicted_occurrence true,
id_differs_from_token_occurrence_over_comparable_f
ields true
      note: 4.0.0 Section A: a
predicted_occurrence over an ordinal tick window
is schema-
            and semantically valid; its identifier
carries the predicted_occurrence: scheme
            and DIFFERS from the token_occurrence
identifier minted over comparable fields
            - a forecast is not a report (Rule
24).

  V121
predicted_occurrence_with_wall_clock_window_and_st
rength_is_valid
      expect: schema_valid true,
semantically_valid true,
strength_is_identity_bearing true
      note: 4.0.0 Section A: a
predicted_occurrence over a wall-clock window
carrying a
            strength (the predicted probability)
is valid; strength is PREDICTED CONTENT
            and identity-bearing - the identifier
with strength differs from the identifier
            without (the narrow, deliberate 4.0.0
extension of Principle P4).

  V122
predicted_occurrence_missing_predictor_is_invalid
      expect: schema_valid false, hard true
      note: NEGATIVE VECTOR. A prediction without
a predictor is not a prediction:
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

predictor is REQUIRED. HARD.

V123 predicted_occurrence_with_both_dimensions_is_invalid
expect: schema_valid true, semantically_valid false, gap dimension_conflict, hard true
note: 4.0.0 Rule 24: a predicted interval carrying BOTH a wall-clock start and an ordinal start_tick is schema-valid but semantically invalid (dimension_conflict) - per Rule 23 the two dimensions never compare, so a window in both is incoherent. HARD.

V124 fulfilled_prediction_error_is_valid
expect: schema_valid true, semantically_valid true, pairing_mismatch false
note: 4.0.0 Section A: a prediction_error whose observed token_occurrence instantiates the SAME occurrent its predicted_occurrence instantiates, with discrepancy 0.0, is valid - the prediction was fulfilled (actual minus expected is zero).

V125 unfulfilled_prediction_error_with_absent_observed_is_valid
expect: schema_valid true, semantically_valid true, pairing_mismatch false
note: 4.0.0 Section A: a prediction_error with observed ABSENT and a negative discrepancy is valid - the predicted occurrence was not fulfilled by any recorded occurrence, and the surprise of nothing arriving is itself a

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

first-class record.

V126 prediction_error_missing_discrepancy_is_invalid
expect: schema_valid false, hard true
note: NEGATIVE VECTOR. An error without a magnitude is not an error: discrepancy is REQUIRED. HARD.

V127 prediction_error_pairing_mismatch_is_semantically_invalid
expect: schema_valid true, pairing_mismatch true, hard true
note: 4.0.0 Rule 24: a prediction_error whose observed token_occurrence instantiates a DIFFERENT occurrent than its predicted_occurrence surfaces pairing_mismatch - a prediction is answered only by an instance of the occurrent it predicted. HARD.

GROUP Y - ATTITUDE AND THEORY OF MIND (4.0.0)

V128 valid_belief_attitude
expect: schema_valid true, semantically_valid true
note: 4.0.0 Section B: an attitude whose holder is a token_individual and whose content is a state_assertion (a true belief) is schema- and semantically valid - the record of what a mind contains.

V129 false_belief_attitude_is_valid_and_quarantined
expect: schema_valid true, semantically_valid true, no_conflict_raised true

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
      note: 4.0.0 Rule 25 (the doxastic
quarantine): an attitude whose content contradicts
            the actual record is VALID and raises
NO conflict - the mismatch IS a false
            belief, first-class and shareable. The
contradiction between the believed claim
            and the actual claim is real (the
formal conflict test finds it between the
            CLAIMS), but the attitude never
asserts its content.

  V130 desire_attitude_is_valid
      expect: schema_valid true,
semantically_valid true
      note: 4.0.0 Section B: attitude_type desires
is a member of the closed enumeration -
            a mind may want what is not the case.

  V131 intention_attitude_is_valid
      expect: schema_valid true,
semantically_valid true
      note: 4.0.0 Section B: attitude_type intends
is a member of the closed enumeration -
            a mind may aim at what is not yet the
case.

  V132 nested_attitude_content_is_another_attitude
      expect: schema_valid true,
semantically_valid true,
outer_and_inner_have_distinct_identity true,
outer_content_is_inner_id true
      note: 4.0.0 Section B: an attitude's content
may be another attitude - A believes
            that B believes X. The two-level
structure is valid, the outer and inner
            attitudes have distinct identities,
and the outer content field carries the
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

inner attitude's identifier by reference (theory of mind, by composition).

V133 unknown_attitude_type_is_rejected
expect: schema_valid false, hard true
note: NEGATIVE VECTOR. The attitude_type enumeration (believes, desires, intends, knows, expects, fears) is CLOSED; extending it is a MAJOR change (Locked Decision 24). HARD.

V134 attitude_with_strength_is_rejected
expect: schema_valid false, hard true
note: NEGATIVE VECTOR. An attitude carries no strength (Principle P4): grading belongs to the provenance layer, and additionalProperties false rejects the field at the schema. HARD.

V135 attitude_is_assertable_through_provenance
expect: schema_valid true, verifies true, holder_distinct_from_source true
note: 4.0.0 Rule 25: an assertion ABOUT an attitude identifier is schema-valid (the about pattern widened in 4.0.0) and verifies; the attitude's HOLDER (a modeled agent) and the assertion's SOURCE (a signing key) are demonstrably different things.

GROUP Z - 4.0.0 IDENTITY WITNESSES

V136 three_zero_zero_identifiers_unchanged_under_4_0_0
expect: wallclock_causal_relation_object_id_unchanged true, unbound_conduit_id_unchanged true

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

note: 4.0.0 identity witness: the exact frozen 3.0.0 identifiers -

causal_relation_object:d8daf899daa3ee03caa6b1425cc6d4d33cef20d951e1203ffd35df29857aa43c (the V111 wall-clock record) and

conduit:dc4af3b1a24f0560d5ebcee488779f06ab3c78301cfb9d0c7edff80bc62e27a6 (the V118 unbound conduit) - are re-pinned unchanged under the 4.0.0 implementation. Every 3.0.0 record keeps its identifier byte-for-byte.

V137 abbreviated_schemes_for_new_kinds_are_rejected
expect: abbreviated_rejected true, whole_word_valid true, hard true
note: NEGATIVE VECTOR. Mirrors V107 for the three 4.0.0 kinds: an abbreviated scheme (an att: / prd: / err: form) fails schema validation; only the whole-word schemes attitude:, predicted_occurrence:, and prediction_error: validate (Principle P7). HARD.

PART 15. QUICKSTART: ONE CLAIM, FOUR LANGUAGES, END TO END

A worked example showing the whole life of one piece of causal knowledge.

(The identifiers are abbreviated; executable versions live in examples/.)

Step 1 - Alice (Python) defines the vocabulary and imports a bare edge.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Alice first calls resolve("press the button", en) - nothing exists yet - so she mints two pure occurrents:

```
    { "type":"occurrent", "label":"press_button",
"category":"action" }
    { "type":"occurrent", "label":"light_on",
"category":"state_change" }
```

Her Software Development Kit's mint_with_aliases sugar also emits her signed English-alias enrichment records for each.

An external knowledge base says "pressing the button causes the light" - so she posts the degenerate claim:

```
    { "type":"causal_relation_object",
"causes":["occurrent:press..."],
"effects":["occurrent:light..."] }
```

and one signed assertion: source ed25519:alice..., evidence_type imported, confidence 0.5.

The store now holds a partial claim, and GET /gaps?kind=missing_field lists it.

Step 2 - Bob (Java) is guided by the gap and enriches both claim and vocabulary.

Bob's dashboard shows the gap.

He knows the mechanism is immediate, so he posts a refinement:

```
    { "type":"causal_relation_object",
"causes":[...same...], "effects":[...same...],
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

"temporal":{"minimum_delay":0,"maximum_delay":1,"unit":"seconds"},
"modality":"sufficient",
"refines":"causal_relation_object:<Alice's>" }

with his signed assertion (evidence_type observation, confidence 0.8).

The refinement is valid, so Alice's degenerate claim leaves the gap list - the gap has visibly closed.

Bob also contributes a signed enrichment record adding a Japanese alias to press_button; the materialized view now shows the occurrent trilingual, with each alias carrying its author.

Step 3 - Mentova (Prolog, the reference implementation) adds intervention-grade evidence.

Playing its environment, Mentova presses the button one hundred times; the light comes on ninety-eight times within a second.

It signs an assertion about Bob's refined claim: evidence_type intervention, strength 0.98, confidence 0.95.

Acting beats watching, and now the claim carries the strongest grade of evidence a causal commons can hold.

Step 4 - Alice retracts her superseded word.

Seeing Mentova's intervention-grade evidence, Alice formally retracts her own low-confidence imported assertion: a signed retraction naming it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Default views now show Bob's and Mentova's assertions only; Alice's remains in history for anyone who opts in.

(Had Bob mistyped his Japanese alias, he could retract that enrichment the same way - the same honest exit works for words and for claims.)

Step 5 - Carol (Swift) consumes with her own trust policy.

Carol's app fetches the claim, its live assertions, and the occurrent's materialized aliases with their contributors.

Her policy: trust intervention over observation; aggregate strength by confidence-weighted mean; honor retractions (the default).

She verifies the Ed25519 signatures locally, computes an effective strength close to 0.98, and - because the claim carries a temporal window - her app also uses temporal admissibility with the fixed constants: a light that came on an hour after the press is NOT attributed to the press.

Four languages, no Foreign Function Interface, no shared code - one data structure, one store, one visible frontier, an honest exit for a source that changed its mind, and an author on every word.

PART 16. THE MINIMUM VIABLE PRODUCT AND THE ROADMAP

A Minimum Viable Product (MVP) is the smallest version that proves the whole idea end to end.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Steps, in order:

1. Publish the repository with spec/ (the normative text, the seventeen schemas, the JSON-LD context, identity.md, semantics.md, provenance.md, store.md, safety.md) and conformance/ (the one hundred and seven vectors).

No new programming language yet.

OUTCOME - done (2026-07-13).

The repository is public at github.com/ai-university-aiu/causalontology.

It carries the complete specification, all seventeen schemas, the one hundred and seven conformance vectors, the governance rules, one license for everything ("The attribution always; no profit, no problem license."), and this document as its canon.

2. Write the second implementation: causalontology-py, passing every vector - the proof that the specification is genuinely language-independent (Prolog was the first implementation).

OUTCOME - done (2026-07-13).

causalontology-py lives in bindings/python/ and passes all one hundred and seven vectors - on its very first run.

It needs nothing installed beyond Python itself: the canonical byte format, the SHA-256 fingerprinting, and even the Ed25519 digital-

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

signature mathematics are written out in plain Python.

The signature code passes the official known-answer test - a worked example published inside the internet standard itself, with the exact expected answer - so we know the mathematics is right, not merely self-consistent.

3. Stand up the Tier A store: the HTTP binding of Part 9.2 over any RDF-capable or document store, with signature verification, materialized enrichment views, retraction handling, and quarantine.

OUTCOME - done (2026-07-13).

The reference store lives in store/server/ and starts with one command; it listens on port 8785, a small tribute to the canonicalization standard.

It verifies every signature on arrival, quarantines unsigned records, honors retractions in its default views while keeping honest history, checks retractions against key-succession lineage, refuses taxonomy cycles with a written reason, and serves every list with pagination.

Twenty of twenty end-to-end checks pass over real HTTP - including the whole quickstart story: mint the vocabulary, write the bare claim, watch the gap appear, refine it, and watch the gap close.

4. Ship the stigmergy read: GET /gaps with the six kinds, and the minimal dashboard.

From day one the commons guides its own growth.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

OUTCOME - done (2026-07-13).

All six gap kinds are live, including demand_supply, which is computed from real readership: the store counts how often each claim is actually read, and a claim that is read often but supported weakly becomes a visible invitation.

The gap list is value-ranked, so the first item is always, literally, the most valuable brick to lay next.

The store also serves a human dashboard at /dashboard - one self-contained page, live totals, the ranked gaps, the surfaced conflicts, a topic filter.

Eight of eight checks pass, including the one that matters philosophically: when intervention-grade evidence lands on a hungry claim, its gap closes by itself.

Acting beats watching, enforced by the frontier.

5. Then, in any order, as demand pulls: causalontology-java and causalontology-swift; the SPARQL endpoint and full JSON-LD publication; the Web Ontology Language mapping; the token tier (delivered); Tier B federation; the reputation and bounty layer (now computable over every kind of contribution); Tier C decentralization; and Mentova reading the gap list and contributing signed, intervention-grade refinements - the first synthetic mind gardening the commons.

OUTCOME - done (2026-07-13), every item; and in the 2.0.0 wave the token tier that was once

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

reserved is now delivered, leaving only the bounty layer for a future major release.

• causalontology-java (bindings/java/): a complete port, 3,232 lines, using only the Java standard library - all one hundred and seven vectors pass in Continuous Integration (CI), the public test machinery that re-runs every suite automatically on every change.

• causalontology-swift (bindings/swift/): a complete port, 2,816 lines - all one hundred and seven vectors pass in CI.

• A bonus third verified implementation, causalontology-js (bindings/javascript/), in JavaScript with zero dependencies - and it was proven byte-identical with the Python binding: the same key, the same identifier, the same signature, bit for bit.

• The SPARQL endpoint: the store answers a documented subset of SPARQL, the standard query language of the linked-data world, and can export its whole contents as N-Triples for any RDF toolchain.

• The Web Ontology Language mapping: spec/schema/causalontology.owl.ttl aligns the seventeen kinds to the Basic Formal Ontology, the Relation Ontology, and the W3C Provenance Ontology.

• Tier B federation: one store can pull another's contents, and the two converge by signed set union - eleven of eleven checks pass, including the beautiful case where two servers each hold one signed vote for the same alias, and after

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

synchronization both show one entry with two contributors.

• Tier C decentralization: replicate.py exchanges offline bundles by any transport at all, and the mathematics of the data model does the rest - merging in any order gives the same store, merging twice changes nothing, every replica breaks a knowledge cycle in exactly the same place, and a single tampered byte is caught by the content addresses and signatures alone.

Eight of eight checks pass.

• The reputation layer: GET /reputation computes a contributor's standing purely from their signed public history - contributions, corroborations, honest retractions, and the grade of their evidence - glass-box, like everything else.

• The token tier (token_individual, token_occurrence, state_assertion, token_causal_claim) shipped in the 2.0.0 major release; only the bounty layer remains reserved, exactly as the governance rules require - major-version material, and saying so plainly is part of the discipline.

• And the crown: Mentova gardening the commons (examples/gardener/).

A glass-box Prolog mind - every line carrying its English explanation - read the gap list over real HTTP, chose the most valuable gap, pressed the button one hundred times with its own hand, induced the missing time window and modality from the tally, minted the refinement, signed an

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

intervention-grade assertion with its own key, and confirmed the gap had left everyone's frontier.

Eleven of eleven checks pass, and the gardener now runs in CI on every change, forever.

The first synthetic mind has laid a brick in the shared wall.

6. The systems tier (added and completed 2026-07-13): Rust, WebAssembly, TypeScript, and Go.

OUTCOME - done.

• causalontology-rust (bindings/rust/): the fourth implementation verified on the authoring machine - all one hundred and seven vectors pass, and the crate compiled cleanly on its very first build.

• The WebAssembly core: because the Rust library carries its schemas inside itself and never touches a disk, it compiles unchanged to WebAssembly - the portable binary format that browsers, edge servers, and most modern platforms can run.

The result is one audited binary that any host can call, and six of six cross-checks prove its answers are byte-identical with the JavaScript binding's.

One core, every host.

• TypeScript: rather than write a fifth copy of the logic that could drift out of agreement, the verified JavaScript binding gained hand-written strict type declarations - TypeScript users get

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

full checking and autocompletion over the exact code that already passes the vectors.

Verified with the TypeScript compiler.

• causalontology-go (bindings/go/): a complete port with zero dependencies, because Go's standard library carries Ed25519 natively; its number-formatting code was tested against the Python reference on more than one million values before it ever ran.

All one hundred and seven vectors pass in CI, on the first attempt.

THE SCOREBOARD, AS BUILT.

Seven independent implementations in seven programming languages - Prolog, Python, JavaScript, Rust, Java, Swift, and Go - plus typed JavaScript and the WebAssembly core: nine implementations and targets, every one passing the same one hundred and seven vectors.

Eight CI gates re-prove all of it on every change, and the Prolog gardener tends the commons at the end of every run.

Programming-language-neutrality is no longer a claim.

It is a table.

7. The ceremony (completed 2026-07-13): the 1.0.0 vector freeze, and publication.

OUTCOME - done, with one honest remainder.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• The freeze: a small, deterministic tool (conformance/freeze_1_0_0.py) replaced every symbolic placeholder in the one hundred and seven vectors with concrete bytes - real sixty-four-character identifiers, real Ed25519 public keys, and, in the assertion vector, a real signature that genuinely verifies.

Run the tool twice and the second run changes nothing - the freeze is idempotent, which is the mathematician's word for "already done stays done."

• One deliberate subtlety, recorded rather than hidden: the two vectors that test self-reference keep well-formed mapped identifiers instead of true content addresses, because a genuinely content-addressed object cannot contain its own fingerprint - which is exactly why the rule they test exists.

• Every implementation re-verified against the frozen bytes: Python, JavaScript, and Rust on the authoring machine; Java, Swift, and Go in Continuous Integration; the WebAssembly core byte-identical as before.

• Every manifest now says 1.0.0, and the repository is tagged v1.0.0 with a GitHub Release carrying the built artifacts: the Python wheel, the npm tarball, the Rust crate, and the WebAssembly core.

• Publication, channel by channel, stated plainly:

Swift Package Manager - LIVE (it publishes by git tag; one dependency line fetches version 1.0.0).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Go modules - LIVE (the module path and tag are in place; one go get command fetches it).

GitHub Release - LIVE, artifacts attached.

crates.io, PyPI, npm, and Maven Central - ALL LIVE, published later the same day: the owner created each account and handed over a fresh token, the machinery did the rest, and every token was retired afterward - the keys stayed with the owner throughout, by design.

The wall is built, the door is open, and every nameplate is on the wall.

The full story of publication day and the announcement is recorded in the Version History appendix.

8. The nine-language wave (completed 2026-07-13, the same day): C#, Ruby, PHP, Dart, Elixir, Haskell, Perl, R, and Lua.

OUTCOME - done: sixteen implementations in sixteen languages.

• Nine parallel ports were built in one wave, each from the same reference, each judged by the same one hundred and seven frozen vectors.

• Four were verified on the authoring machine: C# (a pure port of the signature mathematics over big integers), Dart (pure-Dart hashing and signing), Perl (core modules only, the whole suite in under nine seconds), and Lua - the summit of the wave - where even the big-number arithmetic beneath the signatures was hand-built and cross-checked against the Python reference on two hundred

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

eighty-eight independent arithmetic cases before the curve was assembled.

• Five were written blind - no interpreter on the machine - using the same care-list discipline that carried Java, Swift, and Go through Continuous Integration on their first attempts: Ruby, PHP, Elixir, Haskell (which hand-built both hash functions AND the signature mathematics from first principles, generating its constants from the square and cube roots of primes), and R.

• Seventeen Continuous Integration gates now re-prove every implementation on every change.

• Registry manifests ship with every port; the nine new registries (NuGet, RubyGems, Packagist, pub.dev, Hex, Hackage, CPAN, CRAN, LuaRocks) await only the owner's accounts, exactly as the first four did - and CRAN is a human-review process, stated plainly.

Sixteen languages, one standard, zero shared code - the table speaks for itself.

9. Wave two (completed 2026-07-14): C++, Zig, Julia, and Kotlin/Native.

OUTCOME - done: twenty implementations in twenty languages, and this time every port was verified on the authoring machine before it shipped.

• C++: zero libraries - even the big-number arithmetic under the signatures is hand-built over machine words, validated on three hundred sixty-one arithmetic cross-checks against the Python reference, compiled with every warning enabled and none emitted.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Zig: the friendliest port of the wave - Zig's standard library carries the signature mathematics natively, and the port pinned its facts by reading the standard library's own source rather than guessing.

• Julia: pure standard library, with the signature mathematics ported over Julia's native big integers and proven byte-identical with Python's output.

• Kotlin/Native: the deep end - a language whose native standard library has no cryptography, no JSON, and no big integers at all; all three were hand-built, cross-checked against the reference on four hundred twenty-two arithmetic cases and one thousand two hundred thirty-two number-formatting cases, and the compiled binary passed all one hundred and seven vectors on its very first run.

• Twenty-one Continuous Integration gates now re-prove every implementation on every change.

Twenty languages, one standard, zero shared code.

10. The distribution campaign (2026-07-14): the implementations go to the registries.

OUTCOME - thirteen channels live; the rest in review or ripening.

• Live by token, each published minutes after the owner created an account and handed over a key (and every key retired or encrypted afterward): NuGet for C# (dotnet add package causalontology), RubyGems (gem install causalontology), Hex for Elixir and Erlang, LuaRocks, and Packagist for PHP

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

(composer require causalontology/causalontology) - joining PyPI, npm, crates.io, and Maven Central from release day.

• The Kotlin/Native library also reached Maven Central, hand-bundled and signed exactly like the Java one: io.github.ai-university-aiu:causalontology-kotlin:1.0.2.

• pub.dev for Dart went live with the owner's one Google sign-in click and the machine's publish (dart pub add causalontology).

• Three publish-by-workflow jobs now live in the repository's Continuous Integration: RubyGems, Hex, and LuaRocks were published by runners that RE-PROVED all one hundred and seven vectors before pushing - "never publish unproven bytes" is enforced by the pipeline itself.

• Three community pull requests are in review, each opened only after a real local build test: the Julia General registry (pull request 161292; new packages auto-merge after about three days if the automated checks stay green), vcpkg (pull request 52892, with an honest disclosure that the repository is days old against their six-month maturity guideline - their maintainers may ask us to wait), and Conan (pull request 30612; the owner signs their Contributor License Agreement and their review proceeds).

• Two registrations wait on other people: Hackage's confirmation email and the CPAN administrators' review of the PAUSE account - both recorded in their own TODO files.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• One registry is deliberately deferred: CRAN, whose human reviewers require package-shaping work (documentation pages, a NAMESPACE, their own checker run clean) - a quiet-week project, stated plainly.

• Housekeeping along the way: the C++ binding gained a production CMake package (find_package works; verified end to end), and two packaging-only tags (v1.0.1, v1.0.2) snapshot the growing tree - the specification itself never moved from 1.0.0.

The wall is built, the door is open, most of the nameplates are on the wall, and the last few are in other people's mailrooms.

PART 17. LOCKED DECISIONS

4.0.0 AMENDMENT (recorded here as five 4.0.0 decisions, each an extension, none an override): (1) the object-kind enumeration grows from eighteen to TWENTY-ONE: attitude, predicted_occurrence, and prediction_error, all token-tier (the token tier grows from four to seven, local by default) - each additive, identity-preserving, defined normatively in Part 23; (2) the attitude_type enumeration (believes, desires, intends, knows, expects, fears) is CLOSED, extensible only by a future major version; (3) the doxastic quarantine (Rule 25): an attitude's content is never treated as asserted, and the HOLDER of an attitude is a modeled agent, forever distinct from the cryptographic SOURCE that signs assertions about it; (4) a predicted_occurrence's strength is PREDICTED CONTENT - the grade of the expectation itself - and not a provenance evaluation, so Principle P4's

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

ban on content-tier strength stands for every other content kind and is deliberately, narrowly relaxed here by extension; (5) the assertion 'about' reference widens to the three new kinds (a closed-pattern extension, MAJOR by Decision 24, altering no record's bytes). The three 3.0.0 extension decisions and the thirty 2.0.0 decisions below stand unchanged.

3.0.0 AMENDMENT (recorded here as three 3.0.0 decisions, extending closed enumerations by the one instrument the locked decisions reserve for it - a MAJOR version - and overriding none): (1) the object-kind enumeration is extended from seventeen to eighteen, admitting the new cross_stratal_seam kind; (2) the temporal-window unit enumeration is extended by the ordinal value "ticks"; (3) the conduit gains the optional realized_by reference field. Each is additive and identity-preserving (no existing record's identifier changes), each is defined normatively in Part 22, and none disturbs the token tier, the required-fields minimalism, whole-word Principle P7, or content-addressed identity. The 2.0.0 decisions below stand unchanged.

Decided now, so building can proceed without reopening them:

1. Causalontology's canonical form is a specification plus a conformance suite; PrologAI is the reference implementation, not the definition.

2. The object model has twenty-one kinds: seventeen content kinds - ten type-tier (occurrent, Causal Relation Object, continuant, realizable entity, stratum, bridge,

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

cross_stratal_seam, port, conduit, quality) and seven token-tier (token_individual, token_occurrence, state_assertion, token_causal_claim, attitude, predicted_occurrence, prediction_error) - and four provenance kinds (assertion, enrichment, retraction, succession). (The 2.0.0 model had seventeen kinds in all; the eighteenth, cross_stratal_seam, is the 3.0.0 extension, and the nineteenth through twenty-first - attitude, predicted_occurrence, prediction_error - are the 4.0.0 extensions, each recorded in the amendments above.)

3. CONTENT IS SEPARATED FROM PROVENANCE, UNIFORMLY: content objects carry ONLY their identity-bearing fields and are immutable; every mutable datum in the commons - every alias, link, claim, withdrawal, and key rotation - is a signed, add-only record with an author.

Unsigned attribution is forbidden by design (it enables framing).

4. The standard models BOTH type-level (generic) causation and token-level (particular-occasion) causation through the token tier; token-tier objects are local by default for privacy. Only the optional bounty layer remains reserved.

5. Strength and confidence live in assertions; strength's formal meaning is the do-calculus reading of Part 7.5; strength and confidence are distinct.

6. In a Causal Relation Object only id, causes, and effects are required; partial and degenerate objects are first-class.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

7. Cause, effect, and context references are occurrent identifiers - never free text.

Vocabulary convergence is by canonical labels (English lowercase snake_case whole words, homonyms qualified), REQUIRED closed category enumerations, and the REQUIRED resolve-before-mint workflow.

8. Identifiers are content-addressed: SHA-256 over the RFC 8785 (JSON Canonicalization Scheme) form of the identity-bearing fields (the table in Part 4.4), written scheme:hexadecimal.

9. Merge is trivial by construction: content objects are immutable and idempotent; records are add-only and idempotent; replicas merge by set union; the store is a Conflict-free Replicated Data Type with a deterministic cycle-breaking view rule (7.13) as its only edge case.

10. Enrichment sets (aliases, participants, subsumes, part_of, realized_in) are DERIVED, materialized views over enrichment records, each entry carrying its contributors; the same entry from two sources is corroboration.

11. Enrichment of claims is by the refines link: acyclic, parent-preserving, add-only; gap queries exclude validly-refined parents.

12. Conflict is the formal test of Part 7.6; conflicts are surfaced, never auto-resolved.

13. Hierarchy consistency is the reachability test of Part 7.7 over the defined mechanism graph;

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

absent members make the check indeterminate, not failed.

14. Sources are Ed25519 public keys; all four record kinds are signed over their canonical identity-bearing bytes; unsigned material is at best quarantined.

15. A source may retract only its own records (or its lineage's) - assertions AND enrichments; default views exclude retracted records; history is never erased.

16. Key rotation is by succession records signed by the predecessor; one successor per key; recovery from key COMPROMISE is a governance action, stated honestly as the limit of cryptographic identity.

17. Temporal admissibility uses the fixed conversion constants of Part 7.4 (month = 2,629,746 seconds; year = 31,556,952 seconds); implementations that differ are non-conformant.

18. The canonical serialization is JSON, profiled as JSON-LD; Protocol Buffers and the Concise Binary Object Representation are optional lossless binary encodings.

19. The store contract is the abstract operation set of Part 9.1, the HTTP binding of Part 9.2 with limit-and-cursor pagination, the query-by-example minimum of Part 9.3 (SPARQL is a SHOULD), and the resolve minimum of Part 9.4 (exact matches MUST return, ranked first; fuzziness is free above the floor).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

20. Interoperability across languages is through shared data and the shared store protocol, never through a Foreign Function Interface.

21. Trust is by provenance and consumer-chosen policy, not forced consensus; evidence is graded with intervention above observation.

22. Stigmergy is intrinsic: the six gap kinds of Part 10 are first-class queryable facts, and closing them is done with refines objects and enrichment records.

23. Claims of consequence are always served with provenance and confidence, require signed records, and fall under the logged takedown procedure; takedown semantics are tiered honestly (delete at Tier A, suppression at Tier B, tombstone at Tier C).

24. Extending any closed enumeration is a MAJOR version change - the enumeration rule of Part 13.

25. Licenses: Apache License 2.0 for specification and code; Creative Commons Zero for the commons data.

26. Governance: Semantic Versioning of the specification, conformance-gated releases, a written open change process with conformance impact statements, and public logging of takedowns and registry re-bindings.

27. Naming: "Causalontology" is the proper noun in prose; "causalontology" is the identifier in every machine context.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

28. The 2.0.0 additions are locked: the stratum (level of description), the bridge (cross-stratal same-happening identity), the port and conduit (typed interface and typed connection, sorted by the Perfect Wire Test into pipe and computer), and the token tier (actual occurrence: token_individual, token_occurrence, state_assertion, token_causal_claim). A Causal Relation Object also carries a skips flag for layer-skipping causation, and a realizable carries an identity-bearing label.

29. Hierarchy consistency is checked over BRIDGED reachability (amended Rule 7, Part 7.7); a declared skip is a legitimate edge and a positive finding, never a gap (Part 7.14); the modality enumeration gains a fifth value, enabling; and an assertion may cite token evidence via evidenced_by, with the evidence ladder intervention > observation > simulation.

30. Principle P7: every identifier scheme and type value is a WHOLE WORD, never an abbreviation (occurrent, causal_relation_object, stratum, bridge, port, conduit, quality, token_individual, token_occurrence, state_assertion, token_causal_claim, assertion, enrichment, retraction, succession); external standard names (SHA-256, RFC 8785, RFC 3339, Ed25519, JSON, JSON-LD, BFO, RO, PROV, UCUM) are kept verbatim.

PART 18. ONE-PAGER CHEAT SHEET

The whole Causalontology Architecture Standard (CAS), at a glance.

NOTE: this cheat sheet, and the release sections that follow it (Parts 19 and 20), are a dated

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

snapshot of the 2.0.0 release and cite its figures - seventeen kinds and one hundred and seven vectors. The CURRENT standard is 4.0.0, with TWENTY-ONE kinds and ONE HUNDRED AND THIRTY-SEVEN vectors; for the current model see Parts 3, 6, 14, 22, and 23 and the identity table of Part 4.4.

THE IDEA, IN ONE LINE.

• A verb-first noun-hosting ontology - reality is what happens, and things are its participants - delivering reified causation as a programming-language-neutral standard and a shared, stigmergic, provenance-first commons.

THE SEVENTEEN KINDS.

Nine type-tier content kinds - pure, immutable, content-addressed:

• occurrent: occurrent - a process or event TYPE (a verb); the vocabulary of causes and effects.

• causal_relation_object: Causal Relation Object - THE fundamental unit: a reified causal claim (now with a skips flag for layer-skipping causation).

• continuant: continuant - an enduring thing (a noun); a thin, hosted backbone.

• realizable: realizable - disposition, function, or role; the hinge (now with an identity-bearing label, so one bearer holds many).

• stratum: stratum - a named LEVEL of description within a stratification scheme; says WHERE a claim sits.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• bridge: bridge – a cross-stratal identity: one coarse occurrent IS a set of finer ones; the SAME happening at two levels.

• port: port – a typed INTERFACE borne by a continuant.

• conduit: conduit – a typed CONNECTION from port to port; a pipe if it conveys, a computer if it transforms (the Perfect Wire Test).

• quality: quality – a PROPERTY TYPE a thing can bear (mass, temperature, colour).

Four token-tier content kinds – a particular history, LOCAL BY DEFAULT for privacy:

• token_individual: a PARTICULAR thing (THIS lamp).

• token_occurrence: a PARTICULAR happening at absolute time (THIS press, at 15:07).

• state_assertion: a PARTICULAR value over a PARTICULAR interval.

• token_causal_claim: a PARTICULAR causing, optionally under a covering law, with a token-level but-for counterfactual.

Four provenance kinds – Ed25519-signed, add-only:

• assertion: assertion – who claims it, on what evidence, how strongly (may cite token evidence via evidenced_by).

• enrichment: enrichment – who added one alias, participant, or taxonomy link; now also

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

occurrent_subsumes and occurrent_part_of; every word has an author.

• retraction: retraction - a source withdraws its OWN record; history is kept, marked.

• succession: succession - key rotation, with lineage.

• (The token tier is DELIVERED; only the optional bounty layer stays reserved.)

THE CRO'S FIELDS.

• Required: causes and effects - lists of occurrent identifiers, at least one each.

• Optional: mechanism (the finer CROs that realize it), temporal ({minimum_delay, maximum_delay, unit}), modality (necessary | sufficient | contributory | preventive | enabling), skips (a declared layer-skip: the cause crosses non-adjacent strata with no intervening mechanism), context (enabling occurrents), refines (lineage to the parent it completes).

• A degenerate CRO (causes and effects only) is legal: a partial claim is a standing invitation.

IDENTITY, IN ONE PROCEDURE.

• Take the identity-bearing fields (type injected) - serialize with RFC 8785 - hash with SHA-256 - identifier = prefix : sixty-four hex characters.

• Consequence: the same claim minted anywhere on Earth gets the same identifier; merging needs no coordinator.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

THE SEMANTIC RULES, IN ESSENCE.

• Temporal: minimum_delay <= maximum_delay; fixed constants (month = 2,629,746 seconds; year = 31,556,952 seconds); admissible iff the elapsed time falls inside the window.

• Acyclicity: mechanism and refines may not loop.

• Refinement: same causes and effects; adds only fields the parent left unspecified; changing a parent's field is a RIVAL CLAIM, not a refinement.

• Conflict: same cause and effect sets + compatible contexts + overlapping windows + preventive against a positive modality.

• Hierarchy: consistent iff every parent cause-to-effect pair is reachable through the mechanism members' edges; absent members make it indeterminate, never guessed.

• Enrichment legality: each field belongs to fixed kinds; each entry has a fixed shape (an alias is {lang, text}).

EVIDENCE, GRADED.

• intervention > observation > simulation > derivation > human_hint > imported.

• Acting beats watching: your own hand on the switch outranks any amount of staring.

THE STORE, IN SIX PROMISES.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Immutable, idempotent object writes; identical content is one object, from any number of sources.

• Signed records only; unsigned arrivals are quarantined, never silently trusted.

• Default views honor retractions; the history view keeps everything, marked.

• Only a record's own source (or its succession lineage) may retract it - framing is impossible.

• Taxonomy cycles: refused with a reason when enforcing; after decentralized merges, broken deterministically in the view (latest timestamp loses; ties by identifier) - every replica the same way.

• Tiers: A - one store over HTTP; B - federation by pull and signed set union; C - offline bundles; the data model is a grow-only-set CRDT, so any merge order converges.

STIGMERGY - THE STORE GUIDES ITS OWN GROWTH.

• Six gap kinds, value-ranked: conflict (5), inconsistent_hierarchy (4), missing_field (3), demand_supply (3 + live demand), dangling_reference (2), empty_mechanism (1).

• GET /gaps returns the most valuable brick first; close it with a refinement or a signed enrichment and it leaves everyone's frontier.

CONFORMANCE - THE ONE RULE.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• An implementation is Causalontology-conformant if and only if it passes all one hundred and seven vectors.

• Frozen at 2.0.0 (2026-07-16): the vectors carry concrete bytes - real identifiers, real keys, a real verifying signature; V01 through V38 re-frozen unaltered in meaning, V39 through V107 new.

• Distribution: thirteen channels live (PyPI, npm, crates.io, Maven jar and Kotlin klib, NuGet, RubyGems, Hex, LuaRocks, Packagist, pub.dev, plus SwiftPM, Go, and Zig by git tag); Julia, vcpkg, and Conan in community review.

• Proven in twenty languages (Prolog, Python, JavaScript, Rust, Java, Swift, Go, C#, Ruby, PHP, Dart, Elixir, Haskell, Perl, R, Lua, C++, Zig, Julia, Kotlin/Native), plus TypeScript typings and one WebAssembly core - byte-identical identifiers and signatures across implementations.

GOVERNANCE AND LICENSE.

• Thirty locked decisions; closed enumerations - extending one is a MAJOR version; the token tier delivered, only the optional bounty layer reserved, and said so plainly.

• One license for everything: "The attribution always; no profit, no problem license."

• Glass box, always: every claim traces to a named rule and a signed author, and "why?" always has a readable answer.

PART 21. COMMONS STORAGE AND FEDERATION DESIGN

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The thesis of this Part is short: the commons is a network, not a server. No single machine should ever hold all of it, and by design none has to. Everything the earlier Parts established already points this way - content-addressed identity (Part 4), coordination-free set-union merge (Part 4.5), signed add-only records (Part 8), and the Tier A, Tier B, and Tier C deployment tiers (Part 9.6). This Part makes the network explicit and shows how the store the standard already defines becomes a durable, federated, world-wide commons that stays free and open by construction rather than by charity.

21.1 Why the data WANTS to be a network.

Recall two facts from earlier in this document. First, every object's identifier is the Secure Hash Algorithm 256-bit (SHA-256) digest of its own canonical content, so the identifier IS the content (Part 4.1). Second, the store merges by set union with no coordinator, which is exactly what makes it a Conflict-free Replicated Data Type (CRDT - a data structure that merges concurrent edits deterministically with no central authority, Part 4.5). This is the same mathematics that IPFS (the InterPlanetary File System, a content-addressed distributed file system), Git (the content-addressed version-control system), and Dat (a content-addressed data-sharing protocol) are built on.

The consequence is that the commons does not have to live in one place. It can be a peer-to-peer federation. "Peer-to-peer" means the participating machines - the nodes - talk directly to one another as equals, with no central authority sitting in the middle. Many independent nodes each

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

hold some of the data, pass new objects to one another, and converge to the same whole no matter the order in which the objects arrive. Because content-addressed set union is order-independent and idempotent (merging twice changes nothing), convergence is guaranteed by the data model itself, not by any protocol agreement negotiated at run time.

21.2 Tiers of participation: full nodes, partial nodes, and light clients.

A federation works because not every participant has to carry the same load. There are three natural roles, and any deployment can mix them freely.

A FULL NODE holds everything and serves it. A university, a research institute, or the founding organization would run a full node - the anchors of the network, each a complete copy of the commons.

A PARTIAL NODE holds only a slice of the whole. Because identity is a hash, splitting the data into slices is trivial. To "shard" is to split a large store into slices held by different machines, and a shard here is just a rule - for example "keep every object whose identifier begins with the hexadecimal digits 0 through 3", or "keep the neuroscience strata and everything they touch". A hash-prefix rule divides the commons into even, non-overlapping slices automatically, with no registry of who-holds-what to maintain, because the identifier alone decides which shard an object falls into.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

A LIGHT CLIENT holds almost nothing. It keeps little or no data of its own and instead queries other nodes, then verifies locally whatever comes back. This is the crucial point: a light client need not TRUST the nodes it queries. Every content object self-certifies by its hash - recompute the SHA-256 digest of the returned bytes and check that it equals the identifier that was asked for - and every provenance record is signed with Ed25519 (the elliptic-curve signature scheme of Part 8), so its author can be checked against the signature. A lying or corrupted server is therefore caught by the mathematics, not by reputation. A light client on a phone or in a browser can consume the commons safely while storing none of it.

21.3 What bounds the size: the commons federates LAWS, not diaries.

A natural worry about any world-wide knowledge store is that it will grow without limit. Causalontology's type/token split (Part 3) bounds it by design. The TYPE tier - the general causal claims, the laws and vocabulary - is the shared, world-wide part. The TOKEN tier - a particular mind's particular history, what actually happened to these very things at these very times - is LOCAL BY DEFAULT, for privacy (Part 3 and spec/safety.md).

The intended pattern of use follows from this. A mind observes tokens privately, induces general laws from them, and contributes the LAWS. The private diary of experience stays home; the distilled, general causal knowledge is what enters the commons. So the global commons is nearer the scale of Wikidata - a single shared store of

© 2026-01-01 All Rights Reserved.

general statements - than the scale of "every experience of every mind ever". That is precisely what makes hosting the commons a normal-sized engineering problem rather than an impossible one.

21.4 The storage engine: what a single node actually is.

Because objects are immutable and content-addressed, three valuable properties come for free at the storage layer.

DEDUPLICATION: the same claim minted a million times, by a million contributors, is one object with one identifier, stored once.

INTEGRITY: the hash IS the checksum. There is no separate verification step to bolt on; recomputing the digest is the verification.

PERMANENT CACHING: an immutable object can sit behind any cache or any mirror forever, because it can never change, so there is no cache-invalidation problem to solve - the hardest problem in ordinary caching simply does not arise.

A single node, concretely, is two things bolted together. First, a content-addressed blob store for the immutable objects - an embedded key-value store such as SQLite, RocksDB, or LMDB running on one machine for ordinary volumes, or an open-source S3-compatible object store such as MinIO or Garage for bulk (S3 being the widely-imitated object-storage interface first popularized by Amazon; "S3-compatible" means any store that speaks the same protocol). Second, a locally-computed index for the mutable VIEWS - the materialized enrichment sets, the gap lists, the

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

reputation weights, and the provenance graph. Content is shared across the network; indexes each node rebuilds for itself from the content it holds.

That split mirrors the content-versus-provenance split of the standard exactly (Part 3): the immutable, shared blobs are content, and the locally-derived, re-computable views are each node's own reading of the provenance records. It also permits HOT/COLD TIERING - recent, high-demand objects on fast storage, and the long tail of rarely-touched objects on cheap storage - and the decision of what is hot and what is cold can be guided by the very stigmergy demand signal the store already computes (Part 10), so the storage layout follows real usage rather than guesswork.

21.5 Distribution and durability: signed snapshots.

The single strongest move for durability is to publish periodic SIGNED SNAPSHOTS of the commons as downloadable dumps - the way Wikidata ships regular database dumps, and the way Software Heritage and the Internet Archive preserve data by copying it widely rather than guarding one master copy. A "snapshot" is just a point-in-time bag of content-addressed objects, signed as a whole by the publisher.

Because the objects are content-addressed, a snapshot is trivially verifiable. Anyone can pull it over IPFS, BitTorrent (the peer-to-peer file-distribution protocol), or plain Hypertext Transfer Protocol Secure (HTTPS), verify every single byte against the hashes, and stand up a mirror that is provably identical to the original.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

This one move - treating the commons as a verifiable, downloadable dataset - gives the commons global redundancy BEFORE a single live peer has been recruited. Copies proliferate, each self-checking, and the data survives the loss of any one machine, any one institution, or any one country.

21.6 Reads and writes at scale.

Reads are the easy case. Because objects are immutable, reads are infinitely cacheable, so mirrors, a content-delivery network (CDN - a network of caching servers positioned near users so that data is served from a machine close to the reader), and light clients all absorb read load cheaply. Popular objects end up cached close to whoever wants them, and the origin node is spared.

Writes are almost as simple, because a write is only ever "mint a new immutable object and gossip it". "Gossip" here means each node passes new objects along to its peers, who pass them along in turn, so an object ripples outward across the whole federation without any central dispatcher. Since merges are set union, the order in which objects arrive never matters, and two nodes that received the same objects in different orders still end up identical. Behind the fast gossip runs anti-entropy synchronization, the Tier B mechanism (Part 9.6): "anti-entropy" is a background reconciliation process that periodically compares two replicas and ships whatever each is missing, so that any node which fell behind - because it was offline, or a gossip message was lost - is brought back into agreement. Nodes commonly compare their holdings efficiently using a Merkle structure (a tree of hashes in

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

which each parent hash summarizes its children, so two replicas can find exactly where they differ by comparing a handful of summary hashes instead of every object). The result is a write path with no global lock and no write bottleneck: everyone can mint at once, and the mathematics reconciles them.

21.7 Trust and safety in a federation.

The trust story is the same one the standard already tells, now read at network scale. Every record is signed, so trust is a CONSUMER-CHOSEN policy rather than a forced global consensus: a consumer decides which authors and which evidence grades it will believe, and no vote of nodes can override that private judgement. The token tier stays local by default, so publishing to the federation never leaks a private history unless its owner chooses to publish it. Retraction (Part 8) withdraws a claim without erasing history, so a mirror that syncs a retraction hides the claim from default views while the record of what happened survives. And tamper-evidence comes from the hashes and signatures alone, so even an entirely untrusted mirror cannot corrupt the data without detection - a single altered byte changes an identifier or breaks a signature, and the alteration announces itself. Safety in this federation is a property of the data, not a privilege of the operator.

21.8 The phased build.

The commons can be built up in four phases, each of which is useful on its own and none of which requires the next.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Phase one (DELIVERED – see 21.10): harden the Tier A store (Part 9.2) from an in-memory service into a persistent, content-addressed NODE – a durable blob store for the immutable objects plus a local index database for the views. This is the smallest step that makes the data survive a restart. This phase is now built and merged: the reference node persists across restarts, backed by SQLite in write-ahead-logging mode, with every endpoint and every byte unchanged.

Phase two (DELIVERED – see 21.11): stand up one modest "genesis" full node and begin publishing signed snapshot dumps (21.5), so the data survives the loss of any single machine. From this point on the commons exists as a verifiable dataset that anyone can mirror, independent of any live service. This phase is now built and merged: a persistent node can export a deterministic, content-addressed snapshot committed by a Merkle root and signed with the node's Ed25519 key, and anyone can verify every byte and stand up a mirror from the dump alone.

Phase three (DELIVERED – see 21.12): turn on the Tier B federation already designed (Part 9.6) – gossip new objects between nodes, run anti-entropy synchronization to reconcile stragglers – so that there is no longer any single point of failure. This phase is now built and merged: nodes gossip new records to their peers as they arrive, and a background anti-entropy pass reconciles any two replicas by comparing Merkle roots and exchanging only the delta, so the commons converges even when gossip is dropped, a peer restarts, or writes land out of order. The commons now lives across many cooperating machines, with no single point of failure and no coordinator.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Phase four (DELIVERED - see 21.13): add light clients and CDN-cached immutable objects for cheap global reads, and hash-prefix sharding for partial nodes, so the network scales SIDEWAYS as it grows - more participants means more capacity, not more strain on a central server. This phase is now built and merged: a light client holds almost nothing and verifies every object by hash and every record by signature; immutable content is served with immutable cache headers and an ETag equal to its identifier, cacheable forever behind any CDN, while mutable views are not; and a partial node holds only a hash-prefix slice, advertised in a shard map and resolvable across nodes, with Phase-three federation running per shard. With Phase four merged, the FOUR-PHASE design of Part 21 is COMPLETE.

21.9 Free and open by design.

Because the commons is a federation, its cost is spread across everyone who runs a node; there is no central bill for anyone to pay, and no single operator whose budget the whole enterprise depends on. Institutional full nodes - universities, institutes, archives - anchor durability, and the open license (Part 12) invites each of them to hold a piece, so that the whole is preserved even though no one carries all of it. Every layer of the stack named in this Part is open-source: the embedded stores, the object stores, the transfer protocols, and the reference node software alike.

The deepest point is that the same property makes the commons both mergeable and safe. Content-addressing is what lets independent contributions converge into one store with no coordinator, and

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

it is also what makes the commons cheap to mirror and impossible to quietly corrupt. So the commons stays free and open not as an act of charity that some benefactor might one day withdraw, but as a consequence of how the data is built - a network whose durability, its openness, and its integrity all fall out of the single decision to name every object by the hash of its content.

21.10 Phase one, delivered: the persistent Tier A node.

The first of the four phases is no longer a plan; it is built and merged. The reference Tier A store - the one whose HTTP binding the earlier Parts describe - has been changed from a service that kept everything in memory, and so forgot it all when the process stopped, into a durable NODE that remembers. The important discipline of this change is that it moved only WHERE the data lives, and touched nothing about what the store does: every endpoint, every response shape, every identity rule, and every merge semantic is byte-for-byte what it was. A reader of the store cannot tell that it ever restarted, except that its data survived.

The engine is SQLite in write-ahead-logging (WAL) mode. SQLite was chosen deliberately for a reference node: it ships with Python, so there is no dependency to install and nothing to operate; it is ACID (Atomic, Consistent, Isolated, Durable) and crash-safe, so an interrupted write never leaves a half-written store; and the whole database is a single portable file that anyone can copy, inspect, or back up. Write-ahead logging adds durable writes together with concurrent reads. The higher-scale engines named earlier in

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

this Part - RocksDB, LMDB, and the S3-compatible object stores - remain the right tools for later phases and larger volumes; phase one does not reach for them, because the smallest durable step is the one worth shipping first.

The table layout mirrors the standard's own content-versus-provenance split exactly. One table holds the immutable content objects, keyed by their content-addressed identifier. A second holds the signed, add-only provenance records, keyed by identifier, with each record's Ed25519 signature kept alongside it. A third holds quarantined, unverifiable records, out of the default views. Above these sit the derived tables - the materialized object views, the gap registry, and the reputation weights - and the cardinal rule is that these are NOT sources of truth: they are a re-computable reading of the two tables beneath them, rebuilt from content and provenance alone whenever they are absent or stale. This is the same separation the whole standard rests on, now made physical in the storage layer: shared, immutable content in one place; each node's own re-derivable views in another.

Two properties are enforced at the moment of writing. First, INTEGRITY: a content object is stored only if its identifier equals the scheme followed by the SHA-256 (Secure Hash Algorithm 256-bit) digest of its canonical identity-bearing bytes - the very identity rule of Part 4 - so a mismatched or tampered object is refused at the door and never persisted. Second, IDEMPOTENCE by content address: writing an identifier that already exists is a no-op, exactly as the set-union merge semantics require, so the same claim minted a thousand times is stored once. The

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

database file's location is configurable, defaulting to a git-ignored data directory in the repository so that a real store is never committed by accident, and a node starting cold against an existing database serves everything it holds with no re-ingest step.

What this delivers is the smallest whole thing: one durable node. It does not yet federate, publish snapshots, shard, or serve light clients - those are Phases two, three, and four. Phase two is delivered in 21.11, Phase three in 21.12, and Phase four in 21.13. But the data now survives a restart, and the commons has its first permanent floor to stand on.

21.11 Phase two, delivered: the genesis node and signed snapshots.

The second of the four phases is now built and merged as well. A persistent Phase-one node, run as the durable anchor of the network, can now publish SIGNED SNAPSHOT DUMPS of the commons - and with that, the commons stops depending on any single running machine. It becomes a verifiable dataset that anyone can hold.

A snapshot is two things: a BODY and a MANIFEST. The body is every content object and every provenance record, each written as its exact canonical bytes, one per line, sorted by identifier - a stable, streamable encoding so that a large snapshot flows through without ever being loaded whole into memory, and so that the SAME store always produces a BYTE-IDENTICAL body. The manifest is a small signed header carrying the snapshot-format version, the specification version (2.0.0), a created-at timestamp in Coordinated

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Universal Time (UTC), the counts of content objects and provenance records, and a MERKLE ROOT computed over the sorted body. A Merkle root is a single hash standing at the top of a tree of hashes, in which each parent summarizes its children, so that a change to any one item changes the root; here it commits the whole body to one value. The manifest is then SIGNED with the publishing node's Ed25519 (Edwards-curve Digital Signature Algorithm) key, and the created-at field - the one part that cannot be reproduced - is deliberately EXCLUDED from the Merkle root, so the root stays content-only and the same store reproduces the same root forever, while each publication still carries an honest timestamp.

The discipline is that a snapshot is SELF-VERIFYING and REPRODUCIBLE. Every content object proves itself by its own identifier, which is the hash of its canonical bytes; every provenance record proves itself by its own signature; the body as a whole is committed by the Merkle root; and the root, inside the signed manifest, is authenticated by the manifest signature. Nothing in the snapshot has to be trusted on anyone's word.

Three tools make this concrete. An EXPORT tool reads a persistent node and writes the artifact: the body, the signed manifest, a detached SHA-256 (Secure Hash Algorithm 256-bit) checksum file, and a detached signature - so a downloader can check the bytes and the signature with ordinary tools, without ever running the store. A VERIFY tool re-checks a downloaded dump end to end: the manifest signature (optionally pinned to a publisher the downloader chooses to trust), the Merkle root recomputed from the delivered bytes, every

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

object's content address, and every record's signature. An IMPORT tool runs that same verification and then merges the snapshot into a target store by SET UNION - and because every check runs before any write, a tampered snapshot ABORTS the import and never partially loads, while importing the same valid snapshot twice changes nothing (idempotent, by content address). Verify, then union-merge: exactly the Tier C replica discipline of Part 9.6, now packaged as a portable dump.

Privacy is built into the default. A snapshot MUST NOT include token-tier records - the token individuals, token occurrences, state assertions, and token causal claims that are a particular mind's particular history - unless the operator explicitly opts in. The default snapshot is the shareable commons: the type tier (the laws and vocabulary) plus the provenance about it, and nothing that names a private particular. This is the local-by-default rule of Part 3 and spec/safety.md made physical in the dump format: the snapshot is exactly the "laws, not diaries" boundary, drawn where content-addressing's own unforgetting nature demands it be drawn.

Standing up a mirror needs only the dump and the verify-and-import tool. Pull the four files by any means - a web download, a shared drive, a memory stick - verify every byte, and import into an empty node; the result is provably identical to the origin. Distribution over IPFS, BitTorrent, or plain HTTPS is left as an operational choice and is not automated in this phase; what this phase delivers is the artifact and its proofs, not a hosting pipeline. Real dumps are kept out of the repository; a small worked example is committed so

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

the format is exercised by the tests. With this, the commons has its second permanent floor: copies can proliferate, each self-checking, and the data survives the loss of any one machine, institution, or country - the global redundancy of 21.5, achieved BEFORE a single live peer has been recruited. Phase three, the live Tier B federation with gossip and anti-entropy, is delivered in 21.12, and Phase four, light clients with CDN-cached reads and hash-prefix sharding, in 21.13 - the last of the four.

21.12 Phase three, delivered: live Tier B federation.

The third of the four phases is now built and merged as well. The commons stops being a set of nodes that merely CAN be reconciled by hand and becomes a living network that reconciles ITSELF: nodes gossip new records to their peers as they arrive, and a background pass heals whatever gossip missed, so many machines hold one shared whole with no coordinator, no leader, and no global ordering. The manual Tier B primitive of Part 9.6 - two live stores converging by a signed set-union pull - is not replaced but built upon: this phase makes that convergence AUTOMATIC and EFFICIENT.

A node carries a configurable set of PEERS, each addressed by a base URL (from an environment list or a command-line flag, added or removed without downtime). A peer never has to be trusted to be synced with, because every record it offers is self-authenticating by hash and signature; and because writes are never blocked on a peer, an unreachable peer is retried with backoff rather than allowed to stall the node.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The first mechanism is GOSSIP - push on write. When a node accepts a new type-tier object or provenance record, whether it minted it locally or received it from a peer, it announces it onward to its peers. Three disciplines keep gossip healthy. It is BATCHED, so a burst of writes becomes one message rather than a storm. It is IDEMPOTENT, so a peer that already holds an identifier does nothing - the set-union merge makes a repeat a no-op. And it is LOOP-FREE, because a node gossips onward ONLY the records that were genuinely new to it; once every peer holds an item, the next announcement adds nothing new to announce, and the ripple dies out instead of circling a ring of peers forever.

The second mechanism is ANTI-ENTROPY - scheduled reconciliation, the background repair that catches anything gossip dropped: a lost message, a peer that was offline, two writes that arrived in different orders on different nodes. On a configurable interval a node reconciles with each peer. It first compares MERKLE ROOTS over the type-tier set - the very root the Phase-two snapshot manifest already computes (21.11), reused unchanged. If the roots are equal, the two replicas are already identical and the pass is over, having cost one small request; this is the common case, and it is deliberately cheap. Only when the roots DIFFER does the node do more: it exchanges the ordered set of identifiers, computes the DELTA - the identifiers each side is missing - and transfers ONLY those objects and records, in bounded batches so a large difference streams through rather than loading whole into memory. A handful of differing items moves a handful of items, never the entire store. Reconciliation is

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

order-independent and idempotent, so it can run in any direction, in any order, as often as one likes, and only ever moves the commons toward agreement.

The safety story is the one the standard has always told, now enforced at every hop of the network. INBOUND VERIFICATION IS MANDATORY: before any received item is merged, a content object is checked against its own identifier - the identifier must equal the scheme followed by the SHA-256 digest of the object's canonical identity-bearing bytes - and a provenance record is checked against its own Ed25519 signature. An item that fails is rejected and never merged, while the rest of the same batch merges normally, so a single forged or corrupt item cannot poison an exchange and a hostile or broken peer can never corrupt a store or force a bad record in. Federation is ADD-ONLY: it never deletes; retraction and succession travel as the signed records they are, so a mirror that syncs a retraction hides the claim from default views while the record of what happened survives - history is never erased by a peer. There is NO GLOBAL CONSENSUS and NO ORDERING: no leader, no vote, no total order; contradictory claims coexist, each with its signed provenance, exactly as they do in a single store, and trust remains a local, consumer-chosen policy, untouched by the network.

Privacy is preserved by default, exactly as in the snapshots of 21.11. Gossip and anti-entropy operate over the SHAREABLE COMMONS only - the type tier (the laws and vocabulary) plus the provenance about it - which is the identical set a default snapshot carries. Token-tier records, and any provenance that references them, do NOT federate

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

unless the operator explicitly opts in. The "laws, not diaries" boundary of 21.3 is thus drawn a second time, here at the live network's edge: a node can join the federation and share the commons without ever leaking a private particular history.

Why this is enough, with no consensus protocol bolted on, is the same reason the whole store needs none: the data model is a grow-only-set Conflict-free Replicated Data Type (CRDT). Content is immutable and content-addressed, provenance is signed and add-only, and every disagreement lives in provenance rather than in a contested cell, so merge is set union - commutative, associative, and idempotent. Any set of peers, gossiping and reconciling in any order, with any messages dropped or duplicated, converges to the identical type-tier set. What was a mathematical guarantee about a single store is now a mathematical guarantee about the whole network. With this, the commons has its third permanent floor: it is a living, self-healing network of peers. Phase four, the last, is delivered next in 21.13 - light clients and content-delivery-network-cached immutable objects for cheap global reads, and hash-prefix sharding for partial nodes - so the network scales sideways as it grows.

21.13 Phase four, delivered: light clients, CDN caching, and hash-prefix sharding.

The fourth and last phase is now built and merged as well, and with it the four-phase design of this Part is COMPLETE. The first three phases made the commons durable (a node that remembers), redundant (a signed snapshot anyone can mirror), and self-healing (a live network that reconciles itself). This phase makes it scale SIDEWAYS: it can now be

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

read by clients that hold none of it, cached at the edge for cheap global reads, and hosted a slice at a time by partial nodes, so that more participants mean more capacity rather than more strain on any one machine. The guiding rule throughout is the one the whole standard rests on: SCALE WITHOUT SACRIFICING VERIFIABILITY. Nothing here asks anyone to trust a server; every object still proves itself by its hash and every record by its signature, no matter who served it.

The first delivery is the LIGHT CLIENT. A light client holds almost nothing - at most a small optional local cache - and answers a query by fetching from full or partial nodes over the Hypertext Transfer Protocol (HTTP), then verifying locally whatever comes back. The verification is the whole point. For every content object it receives, it recomputes the identifier from the returned canonical bytes and checks that it equals the identifier it asked for; a single altered byte changes the hash, so a tampered object is caught and rejected. For every provenance record, it checks the Ed25519 signature; a forged or altered record fails and is rejected. This holds regardless of WHICH node or cache served the bytes, so a light client on a phone or in a browser can safely consume a commons served by strangers' mirrors and edge caches it has no reason to trust. And because verification is local, so is JUDGEMENT: the client applies its OWN chosen trust policy - which sources, which evidence grades, which confidence floor - and re-derives retraction and succession honoring from the verified record history itself, rather than trusting any server's filtered view. The server moves bytes; the client decides what they mean.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The second delivery is CDN-FRIENDLY CACHING of immutable content. A content object and a provenance record never change - their name IS their hash - so they can be cached forever, and the hardest problem in ordinary caching, invalidation, simply does not arise. The immutable read path serves them with a long-lived immutable cache directive and an entity tag (ETag) equal to the object's identifier, so any content-delivery network (CDN) edge or reverse proxy may keep a copy indefinitely and answer a conditional revalidation for free. The safety of putting an untrusted cache in front of a node rests entirely on the light client verifying every object by hash regardless of who served it: a stale or even hostile cache cannot deceive a verifying client, because wrong bytes do not hash to the right identifier. Mutable and derived responses - the materialized enrichment view of an object, the gap registry, reputation, any listing that changes as records arrive - are deliberately NOT cached as if immutable; they carry a no-cache directive, so an edge never serves a stale view as though it were settled fact. Immutable content is cacheable forever because its name is its hash; a mutable view is not, and is not treated as if it were.

The third delivery is HASH-PREFIX SHARDING, which lets a PARTIAL node hold only a slice of the commons. Because identity is a hash, splitting the data into slices is trivial: a shard is just a rule over the hash prefix - for example "hold every identifier whose hash begins with the hexadecimal digits 0 through 3" - so assignment needs no coordinator, no registry of who-holds-what, and no rebalancing dance; a node changes the prefixes it declares and it is done, and because the hash is uniformly distributed, equal prefix

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

widths give equal-sized slices for free. A node declares its coverage by configuration or environment variable. Nodes advertise their coverage in a SHARD MAP - a new read endpoint reports a node's own slice plus the coverage of the peers it knows - so a light client or a peer can find which node holds a given identifier. A request to a node for an identifier OUTSIDE its slice does not return a false "not found"; it returns a clear POINTER to a node that does hold it, which the client then fetches from and verifies itself. The COVERAGE INVARIANT is that the union of all shards must span the whole identifier space; the tooling surfaces any gap - a prefix no node holds - as the misconfiguration it is, while overlap between shards is simply harmless redundancy. And crucially, the Phase-three gossip and anti-entropy now operate PER SHARD: a node reconciles only the prefixes it holds, with the other nodes that hold them, comparing a Merkle root over its slice rather than over the whole commons. So federation itself scales sideways - a partial node is never forced to hold, gossip, or reconcile a prefix outside its coverage.

The privacy boundary is drawn here a final time. The token tier - a particular mind's particular history - remains LOCAL BY DEFAULT: it is not served over the public light-client or CDN path, and it is not placed into public shards, unless the operator explicitly opts in. The "laws, not diaries" rule of 21.3, already enforced at the snapshot boundary (21.11) and the live-federation boundary (21.12), now holds at the edge and in the shard map too: a node can serve light clients, sit behind a CDN, and host a slice of the commons without ever exposing a private particular.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

With Phase four merged, the storage-and-federation design of Part 21 is FULLY REALIZED. The commons is a durable, self-healing, horizontally scalable peer network: anyone can read it from a light client that trusts no one, cache it at the edge because its names are hashes, or hold a slice of it as a partial node - free and open by construction, its durability, its openness, and its integrity all falling out of the single decision to name every object by the hash of its content. What remains is not a build phase but ADOPTION: standing up genesis and partner nodes, and growing the commons. The architecture is finished; what remains is to fill it.

GLOSSARY OF ACRONYMS AND TERMS

• Application Programming Interface (API) - a defined set of operations one program exposes for others to call.

• Artificial General Intelligence (AGI) - machine intelligence with broad, human-level competence.

• Assertion - a signed statement by a source about a content object, carrying evidence type, strength estimate, confidence, and timestamp.

• Basic Formal Ontology (BFO) - the upper-level ontology from which Causalontology takes the continuant/occurrent division.

• Best Current Practice 47 (BCP 47) - the Internet Engineering Task Force standard for language tags such as en, es, ja.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Bridge - a cross-stratal identity map: one COARSE occurrent IS a particular SET of finer ones - the same happening described at two levels; the machinery behind the amended Rule 7.

• Bridged reachability - the amended Rule 7 (2.0.0): hierarchy consistency checked after bridges merge same-happening occurrents across strata, enabling multi-scale mechanism-consistency checking.

• Causal Relation Object (CRO) - Causalontology's fundamental unit: a reified causal claim linking cause occurrents to effect occurrents, with optional mechanism, temporal window, modality, context, and refines lineage.

• Concise Binary Object Representation (CBOR) - a compact binary data format mirroring JSON.

• Conduit - a typed CONNECTION from one port to another; a PIPE if it merely conveys, a COMPUTER if it transforms in transit, decided by the Perfect Wire Test.

• Conflict-free Replicated Data Type (CRDT) - a data structure that merges concurrent edits deterministically with no central coordinator; the Causalontology store is one by construction (immutable objects, add-only records, set-union merge).

• Continuant - a thing that endures through time (a noun).

• Continuous Integration (CI) - automatically building and testing every change.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Coordinated Universal Time (UTC) – the global time standard used in timestamps.

• Corroboration – the same enrichment entry contributed independently by more than one source: one displayed entry, several signed contributors.

• Creative Commons Zero (CC0) – a public-domain dedication; the data license of the commons (and of Wikidata).

• Cursor – an opaque continuation token for fetching the next page of a list.

• Degenerate Causal Relation Object – a valid claim carrying only the required fields; a partial, and a stigmergic gap.

• Directed Acyclic Graph (DAG) – a graph with directed edges and no cycles.

• do-calculus – Judea Pearl's formalization of intervention; do(X) means X is brought about by action rather than observed; the foundation of the formal meaning of strength.

• Edwards-curve Digital Signature Algorithm over Curve25519 (Ed25519) – the deterministic signature scheme (RFC 8032) used for source identity and record signing.

• Enabling modality – a Causal Relation Object modality (the fifth value, 2.0.0): a cause that makes an effect POSSIBLE without bringing it about.

© 2026-01-01 All Rights Reserved.

• Enrichment record – a signed statement adding one entry to one enrichment field of one content object; Version 4's guarantee that every word and link in the commons has an author.

• Foreign Function Interface (FFI) – a mechanism for code in one language to call code in another; deliberately NOT used here.

• Framing attack – contributing malicious content attributed to someone else; made impossible here because attribution exists only as signatures.

• General Data Protection Regulation (GDPR) – the European Union's data-protection law; relevant to takedown semantics by tier.

• Hypertext Transfer Protocol (HTTP) – the protocol of the web.

• International Organization for Standardization (ISO) 8601 – the standard date-and-time format used in timestamps.

• Internet Engineering Task Force (IETF) – a body that publishes internet standards, including RFC 8785 and RFC 8032.

• InterPlanetary File System (IPFS) – a content-addressed, distributed file system; a model for Tier C.

• JavaScript Object Notation (JSON) – the canonical wire format.

• JSON Canonicalization Scheme (JCS) – RFC 8785; the deterministic byte form used for hashing and signing.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• JSON for Linked Data (JSON-LD) - a way to write JSON that is also Resource Description Framework linked data.

• Lineage - the chain closure of a source's keys under succession records.

• Materialized view - an object's enrichment sets as served by the store: derived from unretracted enrichment records, each entry with its contributors.

• Merkle Directed Acyclic Graph - a hash-linked, tamper-evident graph structure (as in Git and IPFS).

• Minimum Viable Product (MVP) - the smallest version that proves the whole idea.

• Occurrent - a process or event TYPE that unfolds through time (a verb); the vocabulary of causes and effects.

• Open Biological and Biomedical Ontologies (OBO) - the ontology family that publishes the Relation Ontology.

• Perfect Wire Test - the test that sorts a conduit into a PIPE or a COMPUTER: ask whether a perfect, lossless wire could stand in its place with no change to the outcome; if yes it is a pipe, if the transformation matters it is a computer.

• Persistent Uniform Resource Locator (PURL) - a stable, redirectable web address.

© 2026-01-01 All Rights Reserved.

• Port – a typed INTERFACE borne by a continuant; the point at which things connect.

• Principle P7 – whole words, never abbreviations: every identifier scheme and type value is a readable word (occurrent, causal_relation_object, stratum, ...), while external standard names (SHA-256, RFC 8785, RFC 3339, Ed25519, JSON, JSON-LD, BFO, RO, PROV, UCUM) are kept verbatim.

• Protocol Buffers (Protobuf) – a compact, typed binary serialization format.

• Provenance – the record of who contributed what, on what evidence, with what confidence; carried in signed records – in Version 4, for every datum.

• PROV (the W3C Provenance Ontology) – the web standard vocabulary for provenance, aligned in the JSON-LD context.

• Quality – a PROPERTY TYPE a thing can bear (mass, temperature, colour, charge); the anchor for token-tier state assertions.

• Quarantine – the visibility tier for unsigned or policy-flagged material, excluded from default queries.

• Realizable entity – the hinge: a disposition, function, or role inhering in a continuant yet realized in occurrents.

• Reification – turning a relation (here, causation) into a first-class object that can itself be described and reasoned about.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Relation Ontology (RO) – the Open Biological and Biomedical Ontologies' standard relation vocabulary, alignment target for the causal edge.

• Request for Comments (RFC) – the Internet Engineering Task Force's publication series; RFC 8785 defines canonicalization, RFC 8032 defines Ed25519.

• Resource Description Framework (RDF) – the standard model for linked data as a graph of subject-predicate-object statements.

• RDF-star – the extension of RDF for making statements about statements (reification); the natural home of the Causal Relation Object.

• Retraction – a source's signed withdrawal of its own earlier assertion or enrichment; honored by default views; never erases history.

• Secure Hash Algorithm 256-bit (SHA-256) – the cryptographic hash behind content-addressed identifiers.

• Semantic Versioning – a MAJOR.MINOR.PATCH scheme in which a MAJOR change signals incompatibility; closed-enumeration extensions are MAJOR by rule.

• Skip semantics – the meaning of a Causal Relation Object's skips flag: the cause crosses NON-ADJACENT strata with no mechanism at the intervening levels (layer-skipping causation, e.g. a stress hormone crossing roughly ten levels to bind DNA directly); the absence of a mechanism is a positive finding, not a gap.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Software Development Kit (SDK) – a per-language library implementing the abstract operations.

• SPARQL Protocol and RDF Query Language (SPARQL) – the standard query language for RDF data.

• State assertion – a token-tier kind: a PARTICULAR value of a quality, borne by a token individual, over a PARTICULAR interval of absolute time.

• Stigmergy – coordination through the shared environment: the state of the medium guides the next action.

• Stratum – a named LEVEL of description within a stated stratification scheme (molecular, cellular, psychological); lets a claim say WHERE it sits.

• Subsumption principle – an external causal claim is a degenerate Causal Relation Object, so Causalontology absorbs and enriches it without loss.

• Succession – a signed key-rotation record; the basis of source lineage.

• Sybil attack – one attacker manufacturing many identities; defeated by attaching trust to chosen sources, never to identity count.

• Token individual – a PARTICULAR thing (THIS lamp), instantiating a continuant type; a token-tier kind, local by default.

• Token occurrence – a PARTICULAR happening at absolute time (THIS press, at 15:07), instantiating an occurrent type.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

• Token causal claim – a PARTICULAR causing ("THIS press caused THIS light-on"), optionally under a covering law, optionally carrying a token-level but-for counterfactual.

• Token-level causation – causation between particular occurrences; DELIVERED by the token tier (token_individual, token_occurrence, state_assertion, token_causal_claim), local by default for privacy.

• Tombstone – a signed suppression record; the honest meaning of takedown at the decentralized tier.

• Type-level causation – causation between kinds of occurrents; the scope of this standard.

• Uniform Resource Identifier (URI) – a standard identifier for a resource.

• Verb-first noun-hosting ontology – the name of Causalontology's paradigm: the occurrent (the happening) is the atomic building block and causation is the reified master verb, while things are hosted as first-class participants – the ontology governs the verbs and hosts the nouns, joined by the realizable hinge.

• Web Ontology Language (OWL) – the World Wide Web Consortium's ontology language; a fuller mapping is a governance-track deliverable.

• World Wide Web Consortium (W3C) – a body that publishes web standards.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

CONCLUSION

Causalontology's purest form is a specification, not a program: twenty-one object kinds - seventeen for content (ten type-tier, seven token-tier) and four for provenance - canonicalized by an internet standard and content-addressed; content separated from provenance UNIFORMLY, so that content objects are pure and immutable and every mutable datum in the commons, down to a single alias in a single language, is a signed, add-only, retractable record with an author; a merge story that is therefore trivial - set union - making the whole commons a Conflict-free Replicated Data Type with one deterministic edge rule; a conformance suite of one hundred and thirty-seven vectors that makes "correct" mean the same thing in Prolog, Python, Java, and Swift, down to the length of a month and the ranking of a resolve; a declared scope covering both type-level and token-level causation, the token tier delivered and local by default rather than reserved or fudged; a trust model of graded, signed evidence in which acting beats watching, disagreement is surfaced rather than averaged away, framing is impossible, and an honest source always has an exit; takedown semantics stated honestly for each deployment tier; and a stigmergy that needs no extra machinery, because the structure's own gaps - closable, visibly, by refinement and enrichment - are the invitation that guides every next contributor, human or synthetic.

PrologAI remains the reference implementation and Mentova the flagship mind; the standard belongs to everyone.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Every known inconsistency, gap, and soft spot of Versions 1 through 3 is repaired here; the data model has no remaining known weakness; what remains is to build.

This is the edition to marry, and the edition to take into battle.

PART 22. THE CROSS-STRATAL SEAM, THE ORDINAL TICK, AND THE CONDUIT'S REALIZED_BY

This Part defines three parts of the object model - the ordinal tick temporal unit, the managed cross-stratal seam, and the conduit's realized_by reference - each in the same normative register as Part 6 (the schemas) and Part 7 (the semantics). Each is ADDITIVE and IDENTITY-PRESERVING. Every identifier introduced is a whole English word (Principle P7). Content addressing is unchanged: an object's identifier is the Secure Hash Algorithm 256-bit (SHA-256) digest over the RFC 8785 (JSON Canonicalization Scheme) form of EXACTLY its identity-bearing fields (Part 4.4), so a field that is ABSENT is not serialized and cannot change a hash.

22.1 THE ORDINAL TICK TEMPORAL UNIT.

Definition. The temporal window's closed 'unit' enumeration (Part 6.1, Part 7.4) gains one value, "ticks". A tick is a discrete, dimensionless ORDINAL step - a count, not a duration - with NO wall-clock mapping. The amended enumeration is exactly:

```
    "unit": { "enum":
["instant","seconds","minutes","hours",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

"days","weeks","months","years","ticks"] }

A tick window's minimum_delay and maximum_delay are non-negative INTEGERS (a tick is a whole step; a fractional tick is semantically invalid), and, as for every window, minimum_delay must be less than or equal to maximum_delay (Part 7.1), compared by integer ordering.

The dimension rule. A tick has NO conversion constant, and none is permitted: the fixed constants of Part 7.4 map the eight wall-clock units to seconds and DO NOT extend to ticks. The wall-clock units (instant, seconds, minutes, hours, days, weeks, months, years) form ONE dimension - real elapsed time, measured in seconds - and ticks form a SECOND, DISJOINT dimension - ordinal steps, counted. The two never mix:

• Temporal admissibility (Part 7.4). A tick window is admissible only against an elapsed count expressed in ticks, compared by integer ordering of tick counts; a wall-clock window is admissible only against an elapsed time in seconds. An elapsed time of one dimension is NEVER within a window of the other.

• Conflict and overlap (Part 7.6). A tick window and a wall-clock window are of disjoint dimension and therefore NEVER overlap; they cannot conflict on timing.

• Conversion is refused. Converting ticks to seconds, or seconds to ticks, is a CATEGORY ERROR that a conformant implementation MUST refuse, not a number it may compute. A tick is an order, not a length.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Rationale, in one line. This lets a timing-critical process whose only meaningful clock is a step counter - a cerebellar inter-stimulus interval, a settling-step count, a discrete simulation tick - be recorded NATIVELY, rather than distorted into "seconds" it does not have.

Content-addressed identity policy. The addition is an ADDITIVE enum value; it changes no identity rule. The temporal window (and thus its unit) is already an identity-bearing part of the Causal Relation Object (Part 4.4), so a Causal Relation Object whose window carries unit "ticks" hashes to a DIFFERENT identifier than an otherwise identical Causal Relation Object with unit "seconds" - the unit is identity-bearing, as it always was. No 2.0.0 record is disturbed, because none carried "ticks": a record whose window uses a wall-clock unit hashes byte-for-byte as before (witness V111). Vectors V108 through V111.

22.2 THE MANAGED CROSS-STRATAL SEAM (the eighteenth kind).

The kind. cross_stratal_seam (identifier prefix cross_stratal_seam:) is the eighteenth object kind and a TYPE-TIER content kind - pure, immutable, and content-addressed like the other type-tier kinds. It is a first-class, MANAGED record of a legitimate causal jump across NON-ADJACENT strata: where the bare skips flag (Part 7.14) merely asserts that such a leap occurs, a cross_stratal_seam NAMES the leap, records what is and is not known about the mechanism between its ends, and may draw the intervening pathway.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Schema (spec/schema/cross_stratal_seam.schema.json):

```
    {
      "$schema": "https://json-
schema.org/draft/2020-12/schema",
      "$id":
"https://causalontology.org/schema/cross_stratal_s
eam.schema.json",
      "title": "Cross-Stratal Seam",
      "type": "object",
      "required": ["id", "source", "target",
"mechanism_status"],
      "additionalProperties": false,
      "properties": {
        "id":     { "type": "string", "pattern":
"^cross_stratal_seam:[0-9a-f]{64}$" },
        "type":   { "const": "cross_stratal_seam"
},
        "source": { "type": "string", "pattern":
"^occurrent:[0-9a-f]{64}$",
                    "description": "the endpoint
occurrent at one end of the jump" },
        "target": { "type": "string", "pattern":
"^occurrent:[0-9a-f]{64}$",
                    "description": "the endpoint
occurrent at the other end of the jump" },
        "mechanism_status": { "enum":
["unmodeled","absent"],
                    "description": "unmodeled = an
intervening mechanism EXISTS but is deliberately
not drawn here; absent = there is genuinely NO
intervening mechanism" },
        "chain":  { "type": "array",
                    "items": { "type": "string",
"pattern": "^occurrent:[0-9a-f]{64}$" },
                    "description": "OPTIONAL
ordered intervening occurrents that DRAW the
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
pathway; each strictly between the endpoint
ordinals, one scheme, strictly monotone; a non-
empty chain forbids mechanism_status absent
(Algorithm F)" }
      }
    }
```

The fields.

• source and target are the two ENDPOINT occurrents. Each endpoint is placed at a stratum within a stratification scheme (Part 3, the stratum kind), and the ordinal of that stratum within its scheme is the endpoint's ORDINAL. The two ordinals are the two ends of the jump.

• mechanism_status is the HONEST-IGNORANCE distinction the boolean skips flag could never make. "unmodeled" says an intervening mechanism EXISTS but is deliberately not drawn here - a modeling choice, the seam standing as a placeholder for work not yet done. "absent" says there is genuinely NO intervening mechanism - a positive claim about the world, the same species of positively-expressed negative knowledge as the preventive modality (Part 7.8).

• chain is an OPTIONAL, ordered list of intervening occurrents that DRAWS the pathway between the ends. It is governed by Algorithm F below.

Rule 22 - seam well-formedness (Algorithm F). This is the semantics rule that extends Part 7 for the new kind; JSON Schema checks the shapes above, and Algorithm F checks what only a rule that walks the strata can check. A cross_stratal_seam S is

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

SEMANTICALLY VALID if and only if Algorithm F accepts it:

1. RESOLVE. source and target are present and each resolves to a stratum in ONE stratification scheme; let o_source and o_target be their stratum ordinals. If either endpoint (or, in step 4, any chain member) is absent from the store, the check is INDETERMINATE and surfaces as a dangling_reference gap (as in Part 7.7), never a failure.

2. NON-ADJACENCY. The endpoints lie on NON-ADJACENT strata: the absolute ordinal difference is at least two, so at least one stratum lies strictly between them. A jump between adjacent strata is an ordinary mechanism, not a seam.

3. STATUS. mechanism_status is exactly one of "unmodeled" or "absent".

4. CHAIN. If chain is present and non-empty, then every member resolves to a stratum in the SAME scheme whose ordinal is STRICTLY BETWEEN min(o_source, o_target) and max(o_source, o_target); the members are STRICTLY MONOTONE in ordinal (strictly increasing or strictly decreasing, so no two members share an ordinal and none coincides with an endpoint); AND mechanism_status is NOT "absent". A drawn chain with mechanism_status "absent" is semantically invalid, because a DRAWN mechanism is not an ABSENT one.

5. HOME. The HOME STRATUM of S is the stratum of its GREATER-ORDINAL endpoint - the COARSEST stratum S touches. A layer-to-stratum binding places and checks the seam at its home stratum.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The home rule, stated. A cross_stratal_seam belongs to the COARSEST stratum it touches - the greater-ordinal endpoint. This gives every seam a single, well-defined home in the ladder of description, so that a stratum-organized store, or a layer-to-stratum binding that assigns each construct to exactly one level, can place, index, and check the seam without ambiguity.

Subordination of the skips flag. The Causal Relation Object's boolean skips (Part 7.14) is SUBORDINATED to - not replaced by - the cross_stratal_seam. A bare Causal Relation Object with skips true REMAINS valid and is the COARSE, mechanism-status-UNSPECIFIED form of the same phenomenon: it asserts a leap across non-adjacent strata without committing to whether an intervening mechanism is unmodeled or genuinely absent. A cross_stratal_seam is the RICHER form of that same claim - it commits, through mechanism_status, and may draw the pathway, through chain. The two coexist under the open-world reading (Part 7.8): a bare skip may be enriched into a seam by supplying its mechanism_status; a seam does not invalidate a bare skip; and neither raises an empty_mechanism gap (a seam with mechanism_status "absent" and no chain is the seam-level statement of exactly the honest leap the skip asserts). A rival claim that a mechanism DOES exist where a seam says "absent" is a normal conflict (Part 7.6), settled by evidence, never by mere absence.

Content-addressed identity policy. A new kind is a NEW identity scheme, so it disturbs NO existing record. A cross_stratal_seam's identifier is SHA-256 over the RFC 8785 canonical form of its

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

identity-bearing fields - type, source, target, mechanism_status, and chain - written cross_stratal_seam:<sixty-four lowercase hexadecimal characters>. Every field present is identity-bearing: a seam with mechanism_status "unmodeled" has a DIFFERENT identifier than one identical to it but for "absent"; a seam that carries a chain has a different identifier than one without; and because chain is an ordered array, its order is part of the canonical form (Algorithm F's strict-monotonicity requirement fixes that order). The identity-bearing-fields table of Part 4.4 gains the row:

cross_stratal_seam	type, source, target, mechanism_status, chain

Vectors V112 through V116.

22.3 THE CONDUIT'S REALIZED_BY REFERENCE.

Definition. The conduit kind (Part 3, Part 6.13) gains one OPTIONAL field, 'realized_by': a scheme-qualified reference, BY IDENTITY, to the native law or signal that realizes the conduit's transform. For a COMPUTER conduit (transforms true) it typically references the native rate law the conduit stands for - a causal_relation_object; for a PIPE conduit (transforms false) it typically references the runtime signal the conduit conveys - an occurrent. It is a REFERENCE, not an embedded law: the native dynamics stay native, and the standard records only the BINDING between the typed connection and the dynamics that realize it.

Schema (amending spec/schema/conduit.schema.json). The required fields are unchanged; one optional property is added:

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
    "realized_by": { "type": "string",
                     "pattern":
"^(causal_relation_object|occurrent):[0-9a-
f]{64}$",
                     "description": "OPTIONAL
scheme-qualified reference, by identity, to the
native law (a causal_relation_object) or signal
(an occurrent) that realizes this conduit's
transform; a reference, not an embedded law;
absent means UNBOUND" }
```

If realized_by is present but its referent is absent from the store, the reference surfaces as a dangling_reference gap (the discipline of Part 7.7), not a failure - the open-world reading (Part 7.8) applies. A conduit with no realized_by is UNBOUND, which is legal and complete: unboundedness is unspecified, not empty.

Content-addressed identity policy. The addition is an ADDITIVE OPTIONAL field, identity-bearing WHEN PRESENT. Because content addressing serializes and hashes only the fields PRESENT (RFC 8785 over the identity-bearing fields, Part 4.4), a conduit that OMITS realized_by hashes BYTE-FOR-BYTE as it did under 2.0.0 - its identifier is unchanged (witness V118). A conduit that CARRIES realized_by hashes to a different, new identifier, because the binding is part of what that conduit IS. The identity-bearing-fields row for the conduit (Part 4.4) extends to:

```
    conduit   type, label, conduit_type,
from_port, to_port, transforms, realized_by
```

Vectors V117 through V119.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

PART 23. THE ATTITUDE, THE PREDICTED OCCURRENCE, AND THE PREDICTION ERROR

This Part defines three token-tier kinds - attitude, predicted_occurrence, and prediction_error - which bring the token tier to SEVEN kinds and the object model to TWENTY-ONE, with the assertion's 'about' reference reaching them (Part 6.5). Each is ADDITIVE and IDENTITY-PRESERVING, and each is defined here in the same normative register as Part 6 (the schemas), Part 7 (the semantics), and Part 22 (the cross-stratal seam, the ordinal tick, and the conduit's realized_by). Every identifier introduced is a whole English word (Principle P7), and vector V137 machine-enforces it: an abbreviated scheme for any of the three kinds is rejected, exactly as vector V107 rejects the old abbreviated forms. Content addressing is unchanged: an object's identifier is the Secure Hash Algorithm 256-bit (SHA-256) digest over the RFC 8785 (JSON Canonicalization Scheme) form of EXACTLY its identity-bearing fields (Part 4.4), so a field that is ABSENT is not serialized and cannot change a hash - and every earlier record keeps its identifier byte-for-byte, witnessed by vector V136, which re-pins the exact frozen bytes of the V111 wall-clock Causal Relation Object and the V118 unbound conduit. All three kinds are TOKEN-TIER and therefore LOCAL BY DEFAULT (Principle P2; safety rule R1 of Part 11, spec/safety.md): a conformant store publishes none of them unless its operator explicitly chooses to. The release of these three kinds is part of the record: Sections B and A of the change order that introduced them were released for implementation by the konnectome Requirements Ledger's findings Wall-1 (Section B, agent attitude and theory of mind) and Wall-2 (Section A, prediction,

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

expectation, and prediction error), both of 2026-07-22 and both real findings from a real konnectome build, with the owner's written authorization of 2026-07-22; Sections C, D, and E of that change order remain HELD, and no part of them appears in this specification.

23.1 THE ATTITUDE (the nineteenth kind).

The kind. attitude (identifier prefix attitude:) is a TOKEN-TIER content kind - pure, immutable, and content-addressed like every content kind. It is a first-class record of a PROPOSITIONAL ATTITUDE - a mental stance a particular agent holds toward some content: believing it, desiring it, intending it, knowing it, expecting it, fearing it. An attitude records what the holder's mind CONTAINS, never what is true: the content may be FALSE, and recording the attitude asserts nothing about the world beyond the containing itself.

Schema (spec/schema/attitude.schema.json, reproduced exactly; also listed in the Part 6 catalogue as 6.20):

```
    {
      "$schema": "https://json-
schema.org/draft/2020-12/schema",
      "$id":
"https://causalontology.org/schema/attitude.schema
.json",
      "title": "Attitude",
      "description": "4.0.0. A first-class record
of a PROPOSITIONAL ATTITUDE: what a holder's mind
CONTAINS, never what is true. The holder is a
MODELED agent inside the ontology (a continuant
type or a particular token individual), never a
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
cryptographic key; the content is a reference by
identity to ANY content object - which may be
FALSE, and may itself be another attitude
(nesting). An attitude is NOT an assertion and
carries no strength (Principle P4): it is
asserted, graded, and retracted through the
ordinary provenance layer, and the SOURCE that
signs an assertion ABOUT an attitude is a
different thing from the HOLDER of the attitude
(Rule 25).",
      "type": "object",
      "required": [
        "id",
        "type",
        "holder",
        "attitude_type",
        "content"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^attitude:[0-9a-f]{64}$"
        },
        "type": {
          "const": "attitude"
        },
        "holder": {
          "type": "string",
          "pattern":
"^(continuant|token_individual):[0-9a-f]{64}$",
          "description": "the agent whose mind
contains the content: a MODELED agent inside the
ontology (a continuant type or a particular token
individual), never a cryptographic key - the
signer of an assertion about this attitude is a
SOURCE, and the two are different things (Rule
25)"
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        },
        "attitude_type": {
          "enum": [
            "believes",
            "desires",
            "intends",
            "knows",
            "expects",
            "fears"
          ],
          "description": "CLOSED enumeration;
extending it is a MAJOR change (Locked Decision
24)"
        },
        "content": {
          "type": "string",
          "pattern":
"^(occurrent|causal_relation_object|continuant|rea
lizable|stratum|bridge|cross_stratal_seam|port|con
duit|quality|token_individual|token_occurrence|sta
te_assertion|token_causal_claim|attitude|predicted
_occurrence|prediction_error):[0-9a-f]{64}$",
          "description": "a reference by identity
to ANY content object, INCLUDING another attitude
(nesting: A believes that B believes X). The
content records what the holder's mind contains,
not what is true - it may be FALSE, and a content
that contradicts the actual record raises NO
conflict (Rule 25: that mismatch IS a false
belief, first-class and shareable)"
        }
      }
    }
```

The fields.

• holder - the agent whose mind contains the content: a MODELED agent inside the ontology, a

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

continuant type or a particular token individual, NEVER a cryptographic key. The key that signs an assertion ABOUT an attitude is a SOURCE (Part 8); the holder OF the attitude is a thing in the model; Rule 25 keeps the two forever distinct, and vector V135 demonstrates the distinction end to end.

• attitude_type – one of six values in a CLOSED enumeration: believes, desires, intends, knows, expects, fears. Extending it is a MAJOR version change (the enumeration rule, Locked Decision 24, restated as the second 4.0.0 decision of Part 17); an unknown attitude type is schema-rejected (vector V133).

• content – a reference BY IDENTITY to ANY content object – an occurrent, a causal claim, a state assertion, a seam, a prediction – INCLUDING another attitude, so attitudes NEST: "A believes that B believes X" is an outer attitude whose content field carries the inner attitude's identifier, and the two have distinct identities (theory of mind, by composition; vector V132).

Rule 25 – AN ATTITUDE'S CONTENT IS QUARANTINED (the doxastic rule; "doxastic" means having to do with belief). An attitude records the content of a holder's mind, not a fact about the world. No reasoning, conflict, reachability, or classification rule may treat an attitude's content as asserted, and content that contradicts the actual record raises NO conflict – that mismatch IS a false belief, first-class and shareable. Vector V129 pins the quarantine exactly: the formal conflict test still finds the contradiction between the CLAIMS themselves, but the attitude never asserts its content, so holding

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

the false belief is valid and raises nothing. An attitude carries no strength (Principle P4): grading belongs to the provenance layer - an attitude is asserted, graded, and retracted through ordinary signed records, like any other content object - and a strength field on the attitude itself is schema-rejected (vector V134). One sentence of safety, carried normatively in spec/safety.md: an attitude is an attribution of a mental state to an agent, which makes it among the most sensitive records this standard can carry - share deliberately, grade honestly, retract cleanly.

Content-addressed identity policy. A new kind is a NEW identity scheme, so it disturbs NO existing record. An attitude's identifier is SHA-256 over the RFC 8785 canonical form of its identity-bearing fields - type, holder, attitude_type, and content - written attitude:<sixty-four lowercase hexadecimal characters>; id is excluded (it IS the hash), and all four fields are required, so an attitude's identity covers everything it says: the same holder bearing two different attitude types toward one content is two objects, and the outer and inner attitudes of a nesting are two objects (vector V132). The identity-bearing-fields table of Part 4.4 gains the row:

    attitude   type, holder, attitude_type, content

Vectors V128 through V135.

23.2 THE PREDICTED OCCURRENCE (the twentieth kind).

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The kind. predicted_occurrence (identifier prefix predicted_occurrence:) is a TOKEN-TIER content kind: a first-class record of an EXPECTATION - an occurrent type predicted, by a stated predictor, to occur over a stated window. It is the sibling of the token_occurrence that has not (yet) happened, and the sibling relation is exact: like a token occurrence it instantiates an occurrent type and carries an interval; unlike a token occurrence it reports NOTHING, because a prediction is not an assertion that the thing happened, and it satisfies no rule that requires a token_occurrence (Rule 24). Vector V120 makes the distinction literal at the level of the bytes: the identifier a prediction mints carries the predicted_occurrence: scheme and DIFFERS from the token_occurrence identifier minted over comparable fields - a forecast is not a report.

Schema (spec/schema/predicted_occurrence.schema.json, reproduced exactly; also listed in the Part 6 catalogue as 6.21):

```
    {
      "$schema": "https://json-
schema.org/draft/2020-12/schema",
      "$id":
"https://causalontology.org/schema/predicted_occur
rence.schema.json",
      "title": "Predicted Occurrence",
      "description": "4.0.0. A first-class record
of an EXPECTATION: an occurrent type predicted, by
a stated predictor, to occur over a stated window.
A prediction is NOT an assertion that the thing
happened - it is the sibling of the
token_occurrence that has not (yet) happened, and
it satisfies no rule that requires a
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
token_occurrence (Rule 24). Its interval MUST
carry exactly ONE temporal dimension: a wall-clock
start (with optional end) or an ordinal start_tick
(with optional end_tick), never both and never
neither. The optional strength is the predicted
probability - part of the PREDICTED CONTENT
itself, identity-bearing when present, not a
provenance evaluation (the narrow, deliberate
4.0.0 extension of Principle P4).",
      "type": "object",
      "required": [
        "id",
        "type",
        "instantiates",
        "interval",
        "predictor"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^predicted_occurrence:[0-9a-
f]{64}$"
        },
        "type": {
          "const": "predicted_occurrence"
        },
        "instantiates": {
          "type": "string",
          "pattern": "^occurrent:[0-9a-f]{64}$",
          "description": "the occurrent TYPE
predicted to occur; MANDATORY (P1)"
        },
        "interval": {
          "$ref": "#/$defs/predicted_interval"
        },
        "predictor": {
          "type": "string",
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
          "pattern":
"^(continuant|token_individual):[0-9a-f]{64}$",
          "description": "the predicting source -
a MODELED agent or construct inside the ontology
(a continuant type or a particular token
individual), never a cryptographic key"
        },
        "strength": {
          "type": "number",
          "description": "OPTIONAL, identity-
bearing when present: the predicted probability,
part of the predicted content itself, NOT a
provenance evaluation (4.0.0 Locked Decision
extension of Principle P4)"
        }
      },
      "$defs": {
        "predicted_interval": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "start": {
              "type": "string",
              "format": "date-time",
              "description": "RFC 3339, UTC,
mandatory Z suffix - the wall-clock dimension"
            },
            "end": {
              "type": "string",
              "format": "date-time"
            },
            "start_tick": {
              "type": "integer",
              "minimum": 0,
              "description": "the ordinal
dimension: a discrete tick with NO wall-clock
mapping (Rule 23)"
            },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
            "end_tick": {
              "type": "integer",
              "minimum": 0
            }
          },
          "description": "the predicted window.
Rule 24 enforces exactly ONE temporal dimension:
start (wall-clock, optional end) or start_tick
(ordinal, optional end_tick), never both
(dimension_conflict, HARD) and never neither
(missing_dimension, HARD)"
        }
      }
    }
```

The fields.

• instantiates - the occurrent TYPE predicted to occur; MANDATORY (Principle P1): there are no free-floating predictions, any more than there are free-floating tokens.

• interval - the predicted window, governed by Rule 24. It may carry the WALL-CLOCK dimension - a start (an RFC 3339 date-and-time in Coordinated Universal Time, with the mandatory Z suffix) and an optional end - or the ORDINAL dimension - a start_tick (a non-negative integer count of the dimensionless ordinal tick of Part 22.1) and an optional end_tick. It MUST carry exactly ONE of the two dimensions: both at once is dimension_conflict, neither is missing_dimension, and each is HARD - per Rule 23 (the disjoint-dimension rule of Part 22.1) the two dimensions never compare, so a window written in both is incoherent, not merely redundant (vector V123). The schema deliberately requires NOTHING inside the interval; the exactly-one-dimension law is

© 2026-01-01 All Rights Reserved.

semantic, checked by the rule that sees the whole interval at once.

• predictor – the predicting source: a MODELED agent or construct inside the ontology, a continuant type or a particular token individual, never a cryptographic key. It is REQUIRED, because a prediction without a predictor is not a prediction (vector V122) – and it is deliberately distinct from the SOURCE that signs assertions about the prediction, exactly as an attitude's holder is distinct from the sources that assert it.

• strength – OPTIONAL, and the careful one: the PREDICTED PROBABILITY. It is part of the predicted content itself – the grade of the expectation, what the predictor forecast – and NOT a provenance evaluation of the record; how sure a SOURCE is that this prediction was made, and on what evidence, lives in assertions, exactly as always. This is the deliberate, narrow 4.0.0 extension of Principle P4, recorded as the fourth 4.0.0 decision of Part 17: the ban on content-tier strength stands for every other content kind and is relaxed here alone, because here the strength IS the content. An honest note on its shape: strength is a bare number – the schema states no minimum and no maximum, unlike the assertion's strength, which is bounded zero to one – so "the predicted probability" names the field's intent, not a machine-enforced range.

Content-addressed identity policy. A new kind is a NEW identity scheme, so it disturbs NO existing record. A predicted occurrence's identifier is SHA-256 over the RFC 8785 canonical form of its identity-bearing fields – type, instantiates,

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

interval, predictor, and strength WHEN PRESENT - written predicted_occurrence:<sixty-four lowercase hexadecimal characters>. Because content addressing serializes only the fields PRESENT, a prediction that omits strength is complete and hashes without it, while a prediction that carries strength hashes to a DIFFERENT identifier: the strength is identity-bearing, because a graded forecast and an ungraded one - or two forecasts of different grades - are different expectations (vector V121). A tick-window prediction and a wall-clock prediction are likewise distinct objects, and no prediction ever shares an identifier with a token occurrence (vector V120). The identity-bearing-fields table of Part 4.4 gains the row:

predicted_occurrence type, instantiates, interval, predictor, strength

Vectors V120 through V123.

23.3 THE PREDICTION ERROR (the twenty-first kind).

The kind. prediction_error (identifier prefix prediction_error:) is a TOKEN-TIER content kind: the first-class record of how a prediction met the world - the signed, graded discrepancy between a predicted_occurrence and what was, or was not, observed. The predicted occurrence (23.2) records the expectation before the world answers; the prediction error records the answer - fulfilled, exceeded, fallen short, or nothing at all - so a mind's expectations and their fates are both permanent, inspectable, shareable records.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Schema (spec/schema/prediction_error.schema.json, reproduced exactly; also listed in the Part 6 catalogue as 6.22):

```
    {
      "$schema": "https://json-
schema.org/draft/2020-12/schema",
      "$id":
"https://causalontology.org/schema/prediction_erro
r.schema.json",
      "title": "Prediction Error",
      "description": "4.0.0. The first-class
record of how a prediction met the world: the
graded discrepancy between a predicted_occurrence
and what was - or was not - observed. When
observed is present it names the token_occurrence
that answered the prediction and MUST instantiate
the same occurrent the prediction instantiates
(Rule 24, pairing_mismatch, HARD); when observed
is ABSENT the predicted occurrence was not
fulfilled by any recorded occurrence. discrepancy
reads actual minus expected: positive = exceeded
expectation, negative = fell short, zero =
fulfilled.",
      "type": "object",
      "required": [
        "id",
        "type",
        "predicted",
        "discrepancy"
      ],
      "additionalProperties": false,
      "properties": {
        "id": {
          "type": "string",
          "pattern": "^prediction_error:[0-9a-
f]{64}$"
        },
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

```
        "type": {
          "const": "prediction_error"
        },
        "predicted": {
          "type": "string",
          "pattern": "^predicted_occurrence:[0-9a-
f]{64}$",
          "description": "the prediction this
error grades; MANDATORY - an error without a
prediction is not an error"
        },
        "observed": {
          "type": "string",
          "pattern": "^token_occurrence:[0-9a-
f]{64}$",
          "description": "OPTIONAL: the token
occurrence that answered the prediction. ABSENT
means the predicted occurrence was not fulfilled
by any recorded occurrence - the surprise of
nothing arriving"
        },
        "discrepancy": {
          "type": "number",
          "description": "signed, graded; actual
minus expected - positive = exceeded expectation,
negative = fell short, zero = fulfilled (the
comparator sign convention)"
        }
      }
    }
```

The fields.

• predicted - REQUIRED: the predicted_occurrence this error grades, referenced by identity. An error without a prediction is not an error.

© 2026-01-01 All Rights Reserved.

• observed – OPTIONAL: the token_occurrence that answered the prediction. When PRESENT, it MUST instantiate the SAME occurrent the prediction instantiates – pairing_mismatch, HARD (Rule 24; vector V127): a prediction is answered only by an instance of the occurrent it predicted, never by whatever happened to occur nearby. When ABSENT, the predicted occurrence was not fulfilled by any recorded occurrence – the surprise of nothing arriving, itself a first-class record (vector V125).

• discrepancy – REQUIRED: a signed, graded number reading ACTUAL MINUS EXPECTED – positive means the world exceeded the expectation, negative means it fell short, zero means the prediction was fulfilled (the comparator sign convention; vector V124 pins the fulfilled case at a discrepancy of zero). An error without a magnitude is not an error (vector V126).

Where the pairing check runs, stated honestly. Like the covering-law coherence check of Rule 20 (vector V102), the pairing check of Rule 24 compares objects that the record under test only REFERENCES: deciding pairing_mismatch requires the predicted_occurrence and the observed token_occurrence themselves, not the prediction_error alone. A conformant implementation therefore exposes the check as its own operation over the resolved objects – the Python reference binding exports the helper prediction_pairing_mismatch(error, predicted, observed), mirroring the covering_law_mismatch helper of Rule 20 – rather than pretending a single-object validation could see across three records.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Content-addressed identity policy. A new kind is a NEW identity scheme, so it disturbs NO existing record. A prediction error's identifier is SHA-256 over the RFC 8785 canonical form of its identity-bearing fields - type, predicted, observed WHEN PRESENT, and discrepancy - written prediction_error:<sixty-four lowercase hexadecimal characters>. An absent observed is not serialized and is not hashed, so an unfulfilled error commits, in its very identity, to naming no observation - and a fulfilled error naming its observation is a distinct object from an unfulfilled one, exactly as it should be. The identity-bearing-fields table of Part 4.4 gains the row:

    prediction_error   type, predicted, observed, discrepancy

Vectors V124 through V127; and, for all three kinds at once, vector V136 witnesses that the 4.0.0 implementation disturbs no frozen 3.0.0 byte, and vector V137 witnesses that only the whole-word schemes attitude:, predicted_occurrence:, and prediction_error: are ever accepted (Principle P7).

23.4 WORKED EXAMPLES.

These examples show the token-tier kinds of this Part working together with the rest of the standard.

Each object is written in the document's instance style; a real record carries a full sixty-four-character hexadecimal digest where these show a shortened placeholder ending in an ellipsis.

© 2026-01-01 All Rights Reserved.

Worked example one: theory of mind, a false and nested attitude.

An analyst, modeled as a token individual token_individual:4b1a..., actually records that a market FELL - a token occurrence token_occurrence:fall... that instantiates the occurrent "market_falls".

A second agent, the trader token_individual:7c88..., is modeled as BELIEVING the opposite - that the market rose.

The trader's belief is captured as an attitude whose content is a claim that contradicts the record; and, one level out, the analyst holds an attitude ABOUT the trader's belief, so the outer attitude's content is the inner attitude's own identifier.

The inner attitude - the trader believes the market rose:

```
{ "type": "attitude",
  "holder": "token_individual:7c88...",
  "attitude_type": "believes",
  "content": "token_occurrence:rise..." }
```

The outer attitude - the analyst believes that the trader believes it:

```
{ "type": "attitude",
  "holder": "token_individual:4b1a...",
  "attitude_type": "believes",
  "content": "attitude:<the inner attitude's
identifier>" }
```

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Rule 25 raises NO conflict here, even though the trader's content (the market rose) contradicts the analyst's actual record (the market fell).

That is exactly the point: the mismatch IS a false belief, held first-class and shareable, not an error in the store.

The HOLDER in each record is a MODELED agent - a token individual inside the ontology - which is a different thing from the cryptographic SOURCE that signs the assertions carrying these attitudes into the commons (Rule 25).

Worked example two: the prediction loop over an ordinal tick window.

A predictor - the controller token_individual:c7d0... - predicts that a settling event will occur within a window measured in the dimensionless ORDINAL TICK, and carries an optional strength as the predicted probability.

The predicted occurrence:

```
    { "type": "predicted_occurrence",
      "instantiates": "occurrent:settles...",
      "interval": { "start_tick": 0, "end_tick": 8
},
      "predictor": "token_individual:c7d0...",
      "strength": 0.9 }
```

The interval carries the ORDINAL dimension only - start_tick and end_tick, with no wall-clock start - exactly as Rule 24 requires: exactly one temporal dimension, never both and never neither.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The fulfilled case. The settling event actually occurs and is recorded as a token occurrence token_occurrence:settled... that instantiates the SAME occurrent the prediction instantiates.

The prediction error pairs the two and reads a discrepancy of zero - fulfilled:

```
    { "type": "prediction_error",
      "predicted": "predicted_occurrence:<the
prediction's identifier>",
      "observed": "token_occurrence:settled...",
      "discrepancy": 0 }
```

Because observed instantiates the same occurrent the prediction instantiates, Rule 24's pairing check passes; a discrepancy of zero is the comparator's sign convention for a prediction the world fulfilled exactly.

The unfulfilled case. Had no settling event been recorded within the window, the prediction error would simply OMIT observed:

```
    { "type": "prediction_error",
      "predicted": "predicted_occurrence:<the
prediction's identifier>",
      "discrepancy": -1 }
```

An absent observed is not serialized and not hashed, so this error commits, in its very identity, to naming no observation - the surprise of nothing arriving, itself a first-class, permanent record.

APPENDIX: VERSION HISTORY

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

This document defines Causalontology as ONE INTEGRATED WHOLE at specification version 4.0.0.

The object model, the schemas, the semantics, and the conformance suite of the Parts above are the standard as it now stands, not a stack of editions to be read in sequence.

This appendix is the ONE place where the versions are enumerated, so that the history is recorded honestly without letting it fragment the specification itself.

Every version named here is ADDITIVE and IDENTITY-PRESERVING, a discipline explained in full at the end of this appendix.

- Version 1.0.0 (the genesis).

The first sealed core defined EIGHT kinds of object and THIRTY-EIGHT conformance vectors.

Its central move was to REIFY causation - to turn the sentence "A causes B" into a first-class object, the Causal Relation Object, that a machine can hold, compare, refine, sign, and explain.

Identity was content-addressed from the start: an object's name is the Secure Hash Algorithm 256-bit (SHA-256) digest of its own canonical content, so the same claim minted anywhere on Earth carries the same identifier and merges with no coordinator.

And every mutable word was a signed provenance record: who asserts a claim, with what evidence, at what confidence, was itself data, carried by an Ed25519 signature.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Reified causation, content-addressed identity, and signed provenance are the three pillars every later version rests on unchanged.

• Version 2.0.0 (the whole-word re-mint).

This version re-minted the canon to SEVENTEEN kinds and ONE HUNDRED AND SEVEN vectors, and it delivered the token tier rather than merely reserving it.

It answered four capabilities that the eight-kind core could not:

◦ what LEVEL of description a claim lives at - answered by strata, the ordered levels of description;

◦ whether two records are the SAME HAPPENING seen at different levels - answered by bridges, which identify one happening across levels;

◦ whether two things are CONNECTED - answered by ports and conduits, the typed connection whose legitimacy is decided by the Perfect Wire Test;

◦ and whether something DID ACTUALLY OCCUR - answered by the token tier, a history rather than only a book of laws, local by default so that a mind's particular record of what happened stays private unless its operator chooses to share it.

It also added the ENABLING modality - a cause that makes an effect possible without forcing it - amended Rule 7 to bridged reachability so that reachability follows bridges across levels, and

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

introduced skip semantics for a legitimate leap across non-adjacent levels.

And it fixed Principle P7 - whole words, never abbreviations - as the naming law every identifier scheme has obeyed since.

Vectors V01 through V38 were re-frozen unaltered in meaning under the whole-word schemes; vectors V39 through V107 were new.

- Version 3.0.0.

This version grew the model to EIGHTEEN kinds and ONE HUNDRED AND NINETEEN vectors with three additions, each defined in full in Part 22.

The first was the ORDINAL TICK temporal unit: a discrete, dimensionless count of steps that carries NO wall-clock mapping and forms a dimension DISJOINT from every unit of elapsed time.

A tick window and a seconds window never overlap, a delay measured in one dimension is never within a window of the other, and converting ticks to seconds is a refused category error, not a computed number - so a timing-critical process whose only meaningful clock is a step counter is recorded natively instead of being distorted into seconds it does not have.

The second was the managed CROSS-STRATAL SEAM: a first-class record of a legitimate jump across non-adjacent levels, carrying a mechanism_status that distinguishes "unmodeled" (an intervening mechanism EXISTS but is deliberately not drawn

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

here) from "absent" (there is genuinely NO intervening mechanism).

That distinction turns honest ignorance into a positive finding the bare skip flag could never express; its well-formedness is checked by Algorithm F, and every seam has a single home at the coarsest level it touches (the home rule).

The third was the conduit's REALIZED_BY reference: an optional binding, by identity, from a typed connection to the native law or signal that realizes its transform - a reference, not an embedded law, so the native dynamics stay native and the standard records only the binding between structure and dynamics.

• Version 4.0.0 (released by a gate).

This version grew the model to TWENTY-ONE kinds and ONE HUNDRED AND THIRTY-SEVEN vectors, adding three token-tier kinds defined in full in Part 23.

The ATTITUDE records what a mind CONTAINS: one of a closed set of attitude types - believes, desires, intends, knows, expects, fears - held by a MODELED agent toward any content object, which may be FALSE and may itself be another attitude, so that "A believes that B believes X" nests by composition.

Rule 25, the doxastic quarantine, keeps an attitude's content from ever being read as asserted, so a belief that contradicts the record raises no conflict - because that mismatch IS a false belief, first-class and shareable - and the HOLDER of an attitude is forever distinct from the cryptographic key that signs assertions about it.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

The PREDICTED OCCURRENCE records an EXPECTATION: an occurrent type predicted, by a stated predictor, over exactly ONE temporal dimension, optionally carrying an identity-bearing predicted strength.

Rule 24 keeps a prediction from ever being mistaken for a report: a predicted occurrence satisfies no rule that requires a token occurrence.

The PREDICTION ERROR records the signed discrepancy - actual minus expected - between a prediction and what was, or was not, observed.

The release was GATED, and the gate is part of the record.

A held change order forbade adding ANY of this until the konnectome glass-box build hit a real wall recorded in its Requirements Ledger.

◦ Wall-1 - a false, nested belief could be held privately by the build but could not be represented or shared in the commons - released Section B (the attitude and theory of mind), on 2026-07-22.

◦ Wall-2 - a prediction minted as a token_occurrence was byte-identical to a claim of fact - released Section A (prediction, expectation, and prediction error), on 2026-07-22.

Both findings passed the consumer-versus-ontology test - the gap was in the ontology itself, not in one consumer of it - and the owner authorized the release in writing on 2026-07-22.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

Sections C, D, and E of that change order - imagination quarantine, quantitative sensitivity, and multi-language expression - remain HELD, each awaiting its own recorded finding, and no part of them appears in this specification.

The assertion's 'about' reference widened to reach the three new kinds, and in the same stroke it repaired a genuine earlier omission: the cross-stratal seam had been ABSENT from the 'about' pattern, so until this repair an assertion could not name a seam.

• The discipline, stated once.

Every version named above is ADDITIVE and IDENTITY-PRESERVING.

Content addressing hashes ONLY the fields PRESENT, so a field that is absent is never serialized and cannot change a hash; every earlier record therefore keeps its content-addressed identifier byte-for-byte, and every new kind simply opens a new identity scheme that disturbs no existing record.

This is witnessed at version 4.0.0 by two vectors: V136 re-pins the exact frozen 3.0.0 bytes of the V111 wall-clock Causal Relation Object and the V118 unbound conduit and confirms them unchanged, and V137 enforces whole-word Principle P7 for the three new schemes.

Locked decisions are never silently overridden; they are amended only BY EXTENSION, in the open, exactly as the enumeration rule permits a closed enumeration to be extended by a major version.

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

That is why the standard can grow without ever breaking a record that came before it.

END OF STANDALONE BRANCH, VERSION 27

© 2026-01-01 All Rights Reserved. https://www.linkedin.com/in/d-r-dison

BACK COVER MATTER

Dear Reader,

Thank you for reading:

Causalontology: THE SPEC

This is an evolving work. (As this is an evolving field!)

If you are aware of emerging alternative information that could further the field of Artificial Intelligence Engineering into the future, then please let me know so that the project may be improved in the next edition! :

https://www.linkedin.com/in/d-r-dison

I hope you enjoyed this book!

Positive reviews are the lifeblood of independent authors.

Praise is always welcome in the form of a review, as it helps other readers to find the work.

Constructive criticisms are welcome privately at my LinkedIn so that I can keep improving.

Copyright © 2026 by D. R. Dison (LinkedIn) All rights reserved.

No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, D. R. Dison, except as provided by U.S. Copyright Law or for the use of brief quotations in a book review. Acknowledgment: This work was produced with the aid of various technological tools, including Word Processing Tools (WPT), Desktop Publishing Tools (DPT), Image Processing Tools (IPT), and Artificial Intelligence Tools (AIT).

(International Standard Book Number) ISBN: 979-8-9950671-0-X

Published in the United States of America.

Printer's Key Control Number 10 9 8 7 6 5 4 3 2 1

