Owning the Rails: Higher-Categorical Semantics of Authority, Privacy, and Settlement in Agentic Infrastructure

Anonymous researcher

2 November 2026 · Version 1.0

Contents

Abstract . . . 1
1. The question behind an agentic internet . . . 2
2. An ambient higher-topos semantics . . . 3
3. Authority is a quantified ability to intervene . . . 5
4. Certified workflows and compositional accounting . . . 7
5. Information flow through the entire workflow . . . 9
6. Retention, erasure, and the boundary of control . . . 12
7. Portability, descent, and higher coherence . . . 13
8. Settlement and application-level spending authority . . . 16
9. Decentralization as a question about veto domains . . . 17
10. A worked deployment with an explicit failure analysis . . . 18
11. Implementation, verification, and publication limits . . . 21
Authorship, automation, and conflicts . . . 23
References . . . 23

Abstract

Agentic software combines inference, personal memory, external tools, compute, and payment into workflows that act on a principal's behalf. Whether the principal can use such a workflow is a different question from whether the principal can control its future execution. We develop a semantics of that distinction in an ∞-topos. Resource configurations form types, policies form subobjects, executions form spans with explicit effect witnesses, and intervention strategies supply operational evidence of control. Within this framework we prove preservation and resource bounds for composed certified workflows, a conditional composition result for complete observer views, and a descent theorem for coherently compatible deployment certificates. Elementary counterexamples establish that input-output equivalence does not determine authority, separate private marginals do not imply private joint execution, and local existence does not supply a global control strategy. We also give an erasure obstruction for copies outside a principal's authority, a budget invariant for an abstract settlement machine, and a domain-cut criterion for infrastructural veto power. A worked deployment and executable finite checks make the assumptions inspectable. The contribution is an integrated specification method and a critical formal interpretation of an infrastructure essay, rather than a proof of its commercial or quantitative claims.

Keywords: agentic systems; higher topos theory; authority; noninterference; resource semantics; descent; payment protocols; operational sovereignty.

Publication status: manuscript prepared for Higher Categorical Semantics. This document has not been represented as accepted, reviewed, or published.

1. The question behind an agentic internet

An agent does more than return an answer. It can acquire information, construct a plan, call a model, run a program, record a memory, purchase a service, and alter an external resource. The resulting system has at least two interfaces. One interface describes what a user can request and observe. Another describes which principals can change the system's rules, inspect its intermediate state, prevent an execution, or continue it after a provider withdraws. These interfaces can agree on every successful demonstration while assigning radically different powers to their participants.

The source of this manuscript is Open Protocol Labs' essay *Own the Rails*, distributed through the X post supplied with the publication request [1]. The post resolves to an X article, rather than a short technical specification. Its central concern is the allocation of control over the components used by agents. We take that concern as a research question: which mathematical structures distinguish the ability to obtain a service from the ability to govern its execution, information flow, and continuation?

The source proposes a commercial infrastructure response. Our theory treats its proposed components as candidates for an architecture to be evaluated. It supplies conditions under which particular guarantees follow and countermodels in which superficially similar components provide weaker guarantees. No commercial system is certified merely by being named in this paper. In particular, we have not inspected a deployed network, audited its contracts, evaluated its cryptography, or measured its providers' independence.

This distinction is necessary for a publishable mathematical treatment. A theorem cannot turn a product description into evidence that a production implementation satisfies its hypotheses. A protocol can specify payment messages while leaving application budgets to another layer. A common request format can permit syntactic substitution while leaving a user's stored state inaccessible. A claim about infrastructure can be useful motivation even when the empirical evidence required to establish that claim is outside the present investigation.

1.1 Contribution and relation to existing mathematics

The mathematical ingredients are established ones: dependent types in slices, subobjects as predicates, pullback composition of correspondences, effective descent, and conditional probability kernels. Higher topos theory supplies the ambient language [2]. Higher categories of spans provide a related framework for compositional systems [3]. We use their structural perspective while giving every result below directly in terms of the objects, maps, and hypotheses it needs. We do not claim a new construction of all higher categories of correspondences.

The distinctive contribution is the joint specification of four questions that are often discussed separately. First, what observations can an execution produce? Second, what interventions can a principal force against the modeled environment? Third, how do privacy and resource certificates compose? Fourth, when do locally compatible deployment descriptions define a single global description? Bringing these questions together exposes mistakes that disappear when a workflow is reduced to its final answer.

Several results below are intentionally elementary. Their role is diagnostic. An elementary counterexample can invalidate an ambitious architectural inference more decisively than a sophisticated analogy can support it. Conversely, our positive results state sufficient hypotheses whose satisfaction could be investigated by an implementation, an auditor, or a later verification development. Mathematical originality is claimed only for the organization and application of the specification, subject to comparison with further prior work.

1.2 Evidence and source boundaries

The source was retrieved on 2 November 2026 through a public representation of the linked article. We inspected its article blocks and its embedded table and caption text. The accompanying source record gives the canonical post and article identifiers, retrieval method, counting convention, and a digest of the retrieved representation. The copyrighted article is not reproduced in the publication bundle.

The source includes many numerical and predictive claims. None is a premise of the theorems in this manuscript. We neither validate those numbers by repetition nor use them as constants in a model of actual market behavior. Our worked numbers are declared synthetic. The relevant technical payment reference is the protocol project's own specification [4]. Other references identify mathematical and security foundations rather than supplying empirical confirmation of the essay.

A recurring methodological rule is that we name an observational boundary before stating a guarantee. A system can conceal a prompt from one process while exposing its size to another. It can preserve a user's local file while retaining a derived representation elsewhere. It can execute a correct program while revealing the program's input. The phrase "the system is private" has no determinate mathematical content until the protected input, allowed disclosure, observer, execution horizon, and adversarial powers are fixed.

2. An ambient higher-topos semantics

Fix an ∞-topos $\mathcal{X}$ and a sufficiently large universe for the objects under discussion. We write $\mathcal{S}$ for the ∞-category of spaces. A context is an object Γ of $\mathcal{X}$. A type in that context is an object of the slice $\mathcal{X}_{/\Gamma}$. A term is a section, and substitution along $f : \Delta \to \Gamma$ is pullback. All pullbacks and equivalences are understood homotopically.

This interpretation distinguishes data from a presentation of data. A type can have nontrivial automorphisms, and a certificate can have different equivalent presentations. A change of provider may preserve a deployment only through a specified equivalence. Compatibility between two such changes requires further coherence. The higher structure records these identifications without forcing every comparison into literal equality of files, identifiers, or implementation syntax.

For ordinary discrete information we use 0-truncated objects. Principals, token amounts, artifact identifiers, and finite capability labels can therefore be modeled by internal sets. We use a natural number object $\mathbb{N}$ for bounded iteration and nonnegative resource units. Prices in the examples are integer units of a fixed asset. An application using multiple assets needs a vector of amounts or an explicit conversion model; a mathematical comparison between unlabeled currency quantities is not meaningful.

An ∞-topos has the standard slice, truncation, and descent structures used here [2]. We exploit those structures in specific constructions. We do not assume classical excluded middle or a global

choice principle. This matters operationally: an internal assertion that a suitable execution exists locally need not provide a globally executable algorithm that selects it.

2.1 State, policy, and evidence

A deployment state is an object $S \to \Gamma$. It can include a resource inventory, agent memory, license records, key references, a payment ledger, and a dependency description. A policy is a subobject $P \hookrightarrow S$, meaning a proposition about each state. Such propositions are (-1)-truncated. Satisfaction can be checked internally, but a policy's applicability to a physical machine still depends on an interpretation of the state's fields.

A proof-relevant certificate is a type $C \to S$, not necessarily a subobject. Different witnesses can certify the same state using different migration programs, hardware attestations, or provenance paths. Its propositional truncation expresses that some witness exists. These objects answer different questions. An actual migration term can be executed; a truncated assertion of its existence cannot automatically be extracted into a chosen program.

The distinction also prevents an abuse of higher language. Higher paths record identifications among witnesses already present in a model. They do not manufacture trustworthy evidence about omitted behavior. A hidden logging process remains hidden if the state model omits it, regardless of how much coherence is supplied for the modeled processes.

We maintain a separate interpretation map from physical observations and administrative records into semantic state. This map is part of the trusted boundary of an application. If a hardware claim, an operator identity, or a license record enters the model as an assumption, later reasoning preserves that assumption's conditional status. A proof of a consequence is not an independent measurement of its premise.

2.2 Resources as structured objects

A resource carries more structure than an endpoint. We associate with it an input type, output type, state type, principal type, admissible operations, observation maps, and a family of intervention predicates. Optional fields include location, model artifact, retention deadline, metering rules, and a declared trust domain. A resource schema is a finite diagram of such objects and maps, with explicit equations and policy factorizations.

The following distinctions guide the translation from infrastructure language.

Architectural concern	Semantic object	Evidence still required
Request access	A permitted execution witness	Authentication and availability
Control of execution	A strategy satisfying an intervention goal	Enforcement against the modeled environment
Portable memory	A state translation preserving selected observations	Export completeness and behavioral compatibility
Restricted disclosure	Factorization of the complete observer view	Coverage of logs, timing, errors, and payments
Data-use permission	A predicate on uses and recipients	Enforcement and an interpretation of the license
Independent providers	Distinct modeled failure or veto domains	Accurate operational dependency information
Successful payment	A ledger transition with evidence	Finality assumptions and application accounting

These are interfaces between a mathematical description and evidence. A hash can identify some bytes under a cryptographic assumption; it does not by itself establish their origin or permissible use. A signature can authenticate a statement under a key assumption; it does not establish that the statement is true. Separating these roles makes the resulting semantics usable without assigning magical properties to cryptographic notation.

3. Authority is a quantified ability to intervene

We now give an operational meaning to control. Fix a principal u, a resource r, a finite execution horizon, and an intervention objective I. Examples include rejecting a use, replacing a model, exporting selected state, limiting expenditure, or preventing a new disclosure. Let Σ_u be the type of strategies available to u and let Env be the type of admissible environment behaviors.

A strategy can depend on permitted execution history. Environment behaviors include provider responses, scheduler choices, failures, and adversarial actions allowed by the model. Let $\mathsf{Valid}(\sigma, e)$ express that the combined behavior respects the model, and let $\mathsf{Goal}_I(\sigma, e)$ express that it achieves the stated intervention. Evidence of control is a term of

$$\mathsf{Ctl}_u(r, I) = \sum_{\sigma : \Sigma_u} \prod_{e : \mathsf{Env}} (\mathsf{Valid}(\sigma, e) \Rightarrow \mathsf{Goal}_I(\sigma, e)).$$

The order of quantifiers is essential. A successful execution under one cooperative response proves much less than a strategy that works for every admitted response. Likewise, an abstract existence proposition proves less operationally than a named strategy with a correctness certificate. The model should require at least one valid environment behavior for each admitted strategy, so that control is not obtained from an empty environment or an impossible validity condition.

3.1 Four distinct kinds of power

We separate four fields that are often grouped under ownership. A permission field records what an asserted legal or organizational rule allows. A credential field records who can authenticate a request. An enforcement field records which state transitions a principal can actually prevent or require. A continuation field records whether a principal can maintain an acceptable workflow after a specified external service disappears.

These fields can vary independently. A principal can hold credentials for a rented API without possessing the model artifact. A person can be entitled to deletion while lacking direct control of a remote storage device. A locally stored model can support continuation while still depending on an external activation service. A dataset can be governed by a permission statement while having already been copied to a party whose subsequent behavior is outside the enforcement model.

We use operational sovereignty as a technical abbreviation for a specified family of certified intervention abilities. It is a relative, graded notion: relative to objectives, environment powers, protected resources, and time horizon. It is not a mathematical determination of legal ownership. An infrastructure can improve one coordinate of this family while weakening another.

This formulation also permits an explicit treatment of delegated authority. A delegate's strategies are restricted by a capability predicate, a budget, and an expiry condition. Revocation is a transition that changes the admissible future strategies. It does not imply that all information seen during earlier valid execution becomes unavailable to the delegate. Authority over future actions and erasure of past observations require different certificates.

3.2 Proposition: observation does not determine authority

Let Obs forget a deployment's intervention structure and retain only its successful request-response behavior. There exist deployments D_0, D_1 with identical retained behavior for which a fixed control objective holds in D_0 and fails in D_1.

Proof. Work with finite sets. Both deployments accept a bit x and return x. In D_0, principal u holds an executable copy of the program, and can choose the program that runs on a subsequent request. In D_1, the provider chooses that program, and the only strategy available to u is to submit a request. Let I be the objective of replacing the program with the constant-zero program for the next request, including input 1.

In D_0, choosing the replacement is a strategy achieving I against every environment admitted by this example. In D_1, admit an environment in which the provider keeps the identity program. The user's unique strategy then fails on input 1. The currently displayed request-response behavior is identical, but the intervention objective differs. Therefore no predicate of that behavior alone characterizes this objective. □

The example concerns equality of a specified observational interface, not an assertion that the deployments are equivalent as structured resources. If an interface includes authenticated control operations and their adversarial semantics, it retains more information and can distinguish them. The general lesson is to retain exactly the structure needed for the property one wants to infer.

3.3 Why this requires more than a service category

Objects and request arrows alone can describe which services compose. They do not necessarily describe who can alter those arrows. Our resource semantics therefore retains maps between implementations, maps between certificates, and coherent comparisons between alternative translations. The resulting perspective is higher-categorical because the compatibility data between representations are part of the specification.

Noninvertible refinements must also be retained where appropriate. A translation that discards capabilities is not an equivalence merely because it preserves one observation. It may preserve a safety property while destroying a continuation strategy. A complete theory of an implementation should specify the direction in which each certificate can be transported, rather than declaring every successful integration an equivalence.

4. Certified workflows and compositional accounting

A bounded workflow from state type A to state type B is a span

$$A \xleftarrow{s} E \xrightarrow{t} B.$$

An element of E is an execution witness. It can contain intermediate states, tool responses, errors, cancellations, retries, and a terminal outcome. The source and target maps expose its boundary states. A nondeterministic workflow can have several witnesses with the same source. A deterministic workflow is represented by a special case in which its witnesses are determined by its input, possibly after randomness has been included explicitly.

The span alone is a space of possibilities. It neither supplies a scheduler nor certifies that every possibility is physically realizable. If we require local totality, we can require s to be an effective epimorphism. A chosen executable implementation supplies more structure, such as a selected witness or a transition program. Section 7 explains why this difference cannot be ignored in a higher-topos interpretation.

Let $P \hookrightarrow A$ and $Q \hookrightarrow B$ be policies. Write $E_P = E \times_A P$. A preservation certificate is a factorization of the restricted target map through Q:

$$E_P \longrightarrow Q \hookrightarrow B.$$

This says that every represented execution starting in an admissible state ends in a state satisfying the postcondition. The certificate concerns all represented witnesses, including represented failures. If failure states are omitted, the statement says nothing about them. If retries are hidden inside one witness, that witness's metering and information effects must include them.

4.1 Theorem: preservation under sequential composition

Suppose $A \leftarrow E \rightarrow B$ is certified from P to Q, and $B \leftarrow F \rightarrow C$ is certified from Q to R. Their composite has witness type $E \times_B F$. It is certified from P to R.

Proof. Restrict the composite to input states in P. Projection to E then lands in E_P, and the first certificate supplies a map to Q. The matching boundary condition in the pullback identifies this

state with the source of the F witness. Consequently projection to F lifts to F_Q. The second certificate supplies a map $F_Q \to R$. Composing these maps gives the required factorization to R. Every step uses a pullback or a supplied factorization and is therefore valid in each slice of $\mathcal{X}$. □

Repeated composition gives an inductive certificate for a finite workflow. The identity span has the identity preservation certificate. Associativity is mediated by the canonical equivalences between the different iterated pullbacks, with their usual coherence. These comparisons are relevant when the same workflow is compiled into differently grouped execution plans. They justify transporting the certificate between equivalent compositions without asserting literal equality of every intermediate representation.

There is a significant limitation. An empty execution type satisfies a universal preservation assertion. It does not deliver service. Safety must therefore be accompanied by a separate availability or progress assertion when the application requires one. Under the specific condition that both source maps are effective epimorphisms, the composite source is again an effective epimorphism: it is a composite of a base change of the second source map with the first. Even this provides local existence, not an executable global choice or a termination bound.

4.2 Resource annotations

Fix a finite number d of resource dimensions and let $M = \mathbb{N}^d$ with componentwise addition and order. A workflow has a cost map $c : E \to M$. Dimensions can count payments in one asset, bounded tool calls, or a conservative unit of work. They should not silently combine incomparable resources.

A bound m means $c(e) \leq m$ on all executions under consideration. For consecutive witnesses e, f, define composite cost by

$$c_{\text{seq}}(e, f) = c_E(e) + c_F(f).$$

If $c_E \leq m$ and $c_F \leq n$, then $c_{\text{seq}} \leq m + n$. This is an immediate consequence of monotonicity of addition, but its operational consequence is useful: a compiler can propagate a conservative budget through a certified finite workflow. A bound that covers only successful execution cannot be used as a bound covering failure and retry paths.

Different resources need different composition laws. Elapsed time for independent parallel work is not generally the sum of branch durations. Peak memory is not total allocated memory. Monetary reservations are not settled debits. A real implementation should choose the corresponding ordered algebra explicitly and prove the relevant composition rule. Our finite checker uses additive payment units only.

Parallel composition uses products of independent state and witness objects, with additive monetary cost. Independence here includes disjoint or correctly partitioned spend authority. Two branches that each believe they own the same remaining balance are not a product of independent wallets. An external reservation protocol is needed before parallel composition can justify a shared budget; Section 8 gives one abstract protocol.

4.3 Effect records and refinement

Besides cost, an execution can carry an effect record listing requested capabilities, recipients of disclosed information, retained artifacts, and external trust domains. These records are conservative summaries. Composition takes unions for set-valued exposure and dependency summaries, while accumulating numerical budgets. A data-use policy is checked against the resulting sequence of uses, rather than inferred from the presence of a payment.

A map $h : E \to E'$ over the same boundaries is a refinement witness relating executions. If E' has a preservation factorization, then its composite with h certifies E. A resource bound also transfers when the annotation on E is bounded above by the pulled-back annotation on E'. These conditions must be supplied. A map of unlabeled spans alone cannot establish a claim about metering or disclosure.

Refinement can remove unwanted behaviors. It can also remove every behavior. Thus a safety-preserving refinement need not preserve progress. Similarly, an implementation that returns fewer details may preserve one privacy goal while losing information required for audit. Keeping the direction of each comparison visible prevents several incompatible ideas from being hidden under a single word such as "compatible."

4.4 The compiler obligation

The semantic description can serve as a target for a future verified compiler. Each source operation would elaborate into a span, preservation witnesses, resource annotations, and explicit environmental assumptions. Sequential and parallel syntax would invoke the composition rules just established. Dynamic operations would require a bounded contract or a runtime check maintaining an invariant.

This manuscript does not contain such a compiler. Its executable companion checks finite instances of selected laws and countermodels. It does not turn an arbitrary natural-language objective into a verified agent, and it does not justify running generated code outside its modeled authority. The contribution is a precise target contract against which such an implementation could be developed.

5. Information flow through the entire workflow

Privacy must concern the observer's complete view. Fix a private input type X, an allowed disclosure map $a : X \to A$, and an observer view $v : X \to V$ in a deterministic model. The view can contain tool arguments, network metadata, output lengths, failure codes, payment amounts, retained memory, and final responses. Which fields are included depends on the stated observer.

An exact noninterference certificate relative to a is a factorization

$$v = \bar{v} \circ a.$$

The observer learns no distinction that the allowed disclosure has not already made. If a is the identity, the condition permits complete disclosure and is correspondingly weak. If a is constant, the condition says the entire view is independent of the protected input. The scientific content lies in the choice and justification of a, not in writing the factorization.

This use of noninterference follows a longstanding security perspective [5]. Our application makes the observer boundary include the financial and orchestration components of an agent. A privacy

argument that treats the inference response alone can miss a disclosure made by the service selected, the price paid, or the fact that a particular retry occurred.

5.1 A deterministic factorization criterion

Suppose X, A, V are 0-truncated objects of $\mathcal{X}$, and $a : X \to A$ is an effective epimorphism. Then v factors uniquely through a if and only if it agrees on the kernel pair:

$$v\pi_1 = v\pi_2 : X \times_A X \longrightarrow V.$$

Proof. Any factorization has this equality because $a\pi_1 = a\pi_2$. Conversely, effective epimorphisms between these internal sets are effective quotients of their kernel pairs. The displayed equality supplies a map from the quotient into V. Uniqueness follows from epimorphicity. This is the ordinary effective-quotient argument inside the truncated part of the topos. □

For proof-relevant codomains, one must use the corresponding coherent descent data rather than replace all compatibility by a single unqualified equality. Section 7 states the higher version in a form suited to deployment certificates. Our finite examples use sets, where the elementary criterion is sufficient.

5.2 Counterexample: individually private stages can disclose jointly

Let X be a secret bit and R a fresh uniform random bit. Two stages reveal $V_1 = R$ and $V_2 = R \oplus X$. Each marginal distribution is uniform for each fixed value of X. An observer seeing only one stage cannot distinguish the two inputs.

The observer seeing both stages computes $X = V_1 \oplus V_2$. The joint distributions are supported on disjoint pairs for the two secret values. Thus marginal noninterference at each stage does not imply joint noninterference of the composed workflow. This example remains relevant when information is split among logs, requests, and payment records that are later joined.

The problem is correlation. It cannot be repaired by declaring both stages randomized or by measuring their marginal output statistics more carefully. The certificate must describe conditional behavior given the history already available to the observer, or otherwise account for the complete joint distribution.

5.3 Theorem: conditional finite composition

We now work with finite probability spaces, a concrete semantic model separate from any claim that stochastic kernels form an ∞-topos. Let $h_i = (v_1, \dots, v_{i-1})$ be the observer's previous view. Suppose, for each stage $i \le n$, there is a kernel Q_i such that the actual conditional output law is

$$K_i(v_i \mid x, h_i) = Q_i(v_i \mid a(x), h_i).$$

Assume these conditional laws describe every observer-visible event, including the terminal outcome. Then the joint law of the complete view depends on x only through $a(x)$.

Proof. The law of a particular finite transcript is the product of its successive conditional probabilities:

$$\Pr_x(v_1, \dots, v_n) = \prod_{i=1}^{n} Q_i(v_i \mid a(x), v_1, \dots, v_{i-1}).$$

The expression on the right has no further dependence on x. Consequently inputs with equal allowed disclosure produce identical transcript laws. A stopping decision depending only on the allowed disclosure and the current observer history can be represented by additional output symbols, including a terminal symbol. Padding a bounded run with that symbol reduces bounded stopping to the same product argument. □

The hypothesis concerns conditional kernels, not separate unconditional summaries. It excludes the preceding correlated-release counterexample: after R has been observed, the conditional law of $R \oplus X$ depends on the secret. It also requires that a controller deciding whether to retry cannot secretly choose different retry behavior based on protected information unless that dependence is explicitly allowed.

Markov categories provide an established language for probability kernels and conditional independence [6]. We use finite kernels here to make the exact argument executable and to avoid assuming that a chosen probability construction exists internally in every ambient topos. A later internal probabilistic semantics would need its own distribution object, measurability conditions, and composition theorem.

5.4 Approximate disclosure bounds

For finite distributions define total variation by

$$\mathrm{TV}(\mu, \nu) = \tfrac{1}{2} \sum_z |\mu(z) - \nu(z)|.$$

Suppose inputs x, x' have the same allowed disclosure and, uniformly over every relevant history h, corresponding stage kernels differ by at most ε_i in total variation. Then the complete transcript laws differ by at most $\min(1, \sum_i \varepsilon_i)$.

Proof. Compare hybrid executions that replace the stage kernels one at a time. In two adjacent hybrids, the prefix distribution is identical and only one conditional kernel changes. Averaging the uniform stage bound over that prefix gives distance at most ε_i. Applying the common remaining stochastic computation cannot increase total variation. The triangle inequality bounds the total distance by the sum. Every total-variation distance is also at most one. □

Uniformity over histories matters because adaptive selection can concentrate execution on unusual branches. A bound established only for a conveniently averaged history law does not supply the stated hypothesis. Infinite execution likewise requires an additional argument; our theorem and checker concern bounded finite transcripts.

5.5 Correctness, confidentiality, and location

A proof that a program evaluated a specified function does not prove that the evaluator kept its input secret. An evaluator can compute the correct answer and copy the input into an auxiliary log. Unless the verified relation constrains that log or the observer cannot access it under independently justified assumptions, functional correctness and confidentiality remain separate.

Likewise, physical locality and logical confidentiality are distinct specifications. A remote confidential-computing service receives data at a remote physical location even if a protected execution boundary is intended to hide plaintext from an operator. A useful certificate must state the observer excluded by that boundary and the hardware and software assumptions supporting it.

The same discipline applies to pricing. If the amount settled depends on a secret classification, a public payment record can disclose that classification even when the model request is encrypted. A constant-price route can avoid that particular channel at some economic cost. Alternatively, the policy can declare the classification part of the allowed disclosure. The semantics makes that trade visible without deciding it for the principal.

6. Retention, erasure, and the boundary of control

Deletion is not a single operation on a single universally shared state. A distributed execution can create local files, remote logs, backups, derived features, and model updates. An erasure objective must specify which of these are covered, which observers are considered, and what residual information is permitted.

Let a complete post-execution state be (L, R), where L is controlled by principal u and R belongs to an external environment. A local deletion command can alter L. It cannot be assumed to alter R unless the model supplies an enforceable transition on R. This elementary separation is enough to limit what a deletion certificate can establish.

6.1 Proposition: an uncontrolled-copy obstruction

Suppose an admitted environment can retain a bit distinguishing private inputs 0 and 1 in R, and can preserve that bit through every action available to u. No strategy restricted to those actions guarantees exact post-deletion noninterference for an observer able to read that bit, when allowed disclosure is constant.

Proof. Choose the admitted environment that preserves the bit. Any user strategy changes only components that do not remove this distinction. The observer's post-deletion view is therefore different on the two private inputs. A constant allowed disclosure cannot factor that view. Since a control strategy must succeed against every admitted valid environment, no such strategy achieves the specified erasure goal. □

The conclusion is conditional on the admitted copy and preservation behavior. A stronger enforcement boundary can exclude that behavior and permit a stronger result. The theorem does not assert that every remote service retains a copy, or that deletion is impossible in every architecture. It identifies precisely why an unrestricted external-copy model is incompatible with an unconditional erasure guarantee.

6.2 Erasure as a counterfactual comparison

A useful exact erasure target compares the remaining observer view with a reference execution in which the deleted information was never supplied, except for explicitly allowed residual information. If a records those residual disclosures, one can require the post-erasure view to factor through a. An approximate target can instead bound a distance between the corresponding distributions.

The reference execution must be specified carefully. Deleting a purchased record may leave an authorized invoice. Deleting a message may leave a nonsecret aggregate count. Deleting training examples may leave a model whose distribution still depends on those examples. Each case permits different residual information. A statement about the disappearance of an original file is insufficient to establish a statement about every derivative.

Nor does a signature on a deletion receipt establish physical erasure by itself. The signature can authenticate the issuer's assertion. Whether the assertion is supported by a verified storage mechanism, an audit, or contractual trust is a separate question. An evidence type should therefore record both the asserted event and the assumption connecting that event to the intended observer model.

6.3 Licensing without an enforcement fiction

We model an asserted license as a predicate on a use event. A use event includes the data identifier, recipient, purpose, time, and any further fields relevant to the permission. A certified workflow can check that each represented use satisfies the predicate. The certificate's force depends on whether all actual uses are represented and whether unlicensed transitions are preventable.

Purchasing a dataset and recording a token transfer do not logically imply that no further copy exists. An ownership token, a content identifier, and a use permission are different types of record. A system may bind them through additional mechanisms, but the binding is a hypothesis to state and test. Our semantics neither interprets a payment as universal permission nor interprets a permission record as physical exclusivity.

This approach supports a more precise allocation of responsibility. The data principal chooses an allowed-use predicate. The runtime enforces the modeled capability boundary. An auditor examines whether the interpretation includes all relevant recipients and storage systems. A legal or institutional process may address behavior outside that boundary. Combining these roles in a deployment is possible; confusing them weakens the resulting guarantee.

7. Portability, descent, and higher coherence

Portability is a property of a structured translation. Suppose two deployments have state types S and S', selected observation maps $o : S \to O$ and $o' : S' \to O$, and a migration map $m : S \to S'$. Observation preservation requires a specified homotopy $o'm \simeq o$. This condition alone says nothing about permissions, control strategies, future transitions, resource bounds, or retained copies.

For those properties, the migration contract must include corresponding maps and compatibility. A memory migration can preserve a conversation transcript while dropping tool credentials or changing the interpretation of a stored preference. A model migration can preserve input syntax while changing the distribution of outputs. A payment migration can preserve an account identifier while changing which principal can authorize a debit.

7.1 Transporting an intervention strategy

Consider two finite interaction models. Suppose we have equivalences between their user-strategy types and their environment-behavior types, together with an equivalence of execution outcomes that preserves validity and the chosen intervention goal. Then a winning intervention strategy transports from one model to the other.

Proof. Map the original user strategy to the second strategy type. Given any valid environment behavior in the second model, use the inverse environment equivalence to obtain a behavior in the first. Validity preservation allows application of the original control certificate. Goal preservation carries the resulting witness back to the second outcome. This constructs the required dependent function over second-model environments. □

This is a deliberately strong sufficient condition. Real migrations can use simulations with weaker hypotheses, but those hypotheses must still justify transport of the specific universal claim. A common API establishes none of these equivalences automatically. The equivalence of environment powers is particularly important: migration into a provider with an additional unilateral veto can destroy a previously available strategy.

7.2 Families of deployment certificates

Let $\mathsf{Dep}(\Gamma)$ be the ∞-category of deployment structures over context Γ. For the theorem below, a deployment structure consists of a fixed finite diagram of objects and maps in $\mathcal{X}_{/\Gamma}$, specified equations, selected subobject factorizations, and designated equivalences. Parameters are pulled back along context changes. Morphisms preserve the declared structure, and their higher comparisons are those inherited from the slice.

Examples include the finite workflow spans of Section 4, their preservation factorizations, named resource maps, and selected finite-horizon strategy certificates presented as maps. The word "named" matters: an actual compatible witness is included. Merely asserting the local existence of some witness is a weaker specification.

We require every imposed condition to be invariant under equivalence and local for the chosen effective-epimorphism covers. Conditions involving an unmodeled global registry, an external undecidable oracle, or an unexplained global choice do not enter this theorem by default. One can add further structure, but its descent behavior must then be established separately.

7.3 Theorem: descent of coherent deployment structures

Let $q : U \to \Gamma$ be an effective epimorphism. Let $U_\bullet$ be its Čech nerve, with

$$U_n = \underbrace{U \times_\Gamma \cdots \times_\Gamma U}_{n+1 \text{ factors}}.$$

For deployment structures satisfying the preceding definition, restriction induces an equivalence

$$\mathsf{Dep}(\Gamma) \simeq \lim_{[n]\in\Delta} \mathsf{Dep}(U_n).$$

Consequently a fully coherent family of local workflow structures and their specified certificates glues to a global structure, uniquely up to the equivalence expressed by this limit.

Proof. Effective descent in an ∞-topos identifies the slice over Γ with the limit of the slices over the Čech nerve. Apply this equivalence to every vertex and arrow of the fixed diagram. Taking functor categories for the fixed diagram shape preserves the relevant limit description, so coherent local diagrams descend.

The specified equations descend because mapping spaces in the slice participate in the same equivalence. A factorization is extra diagram data and therefore descends together with its maps. Whether a designated arrow is an equivalence can be checked after effective-epimorphism base change. The assumed local conditions defining the deployment subcategories are consequently satisfied by the descended diagram.

Finally the same construction applies to structure-preserving morphisms and their higher identifications. It therefore produces an equivalence of the stated ∞-categories, rather than a correspondence only between isomorphism classes of objects. □

The proof specializes a standard higher-topos descent principle [2]. Its application here is that operational evidence must be included in what descends. Higher coherence is neither decorative nor a substitute for implementation evidence. It is the compatibility required when certificates are represented in overlapping contexts and compared through several translations.

7.4 Local existence need not provide global control

There is an especially clear counterexample in a topos of group actions. Let G be the two-element group and consider $\mathcal{X} = \mathsf{Fun}(BG, \mathcal{S})$. Regard G as a discrete object with its regular left action. The map $G \to 1$ is an effective epimorphism, since its underlying map of spaces is surjective on components.

A global section would choose a G-fixed element of the regular action. No such element exists: the nonidentity group element exchanges the two elements. Thus the object has local inhabitants but no global term. Pulling it back along itself supplies local choices, but those choices do not come with the compatibility required to descend to a section over 1.

Operationally, this separates a context-dependent access witness from a globally selected intervention program. The example does not model a particular identity provider. It shows that the inference from local availability to a global strategy is invalid in a valid ambient semantics. A theorem claiming that users can always choose among locally available execution providers must either construct the choice or state an additional choice hypothesis.

7.5 What higher structure contributes in practice

A deployment may be exported to a local workstation, re-encoded for a hosted service, and translated again for an alternate model. Each translation can preserve selected observations only up to an explicit equivalence. Two routes through these translations should agree through a comparison, and comparisons along longer routes should themselves be compatible.

In an ordinary set-based implementation, this often appears as versioned schemas, migration tests, identity mappings, and reconciliation rules. Higher semantics provides a unified language for the compatibility problem. It does not imply that every data format requires an elaborate homotopy calculation. The benefit is to expose what must be preserved and which compatibility evidence is missing when several independently reasonable migrations fail to agree globally.

8. Settlement and application-level spending authority

Payment is an action inside a larger protocol, not a synonym for successful service. The x402 project's core specification describes payment requirements, payloads, verification, and settlement interfaces. Its stated scope excludes client budget management [4]. We therefore give a separate abstract accounting machine rather than ascribe the following invariant to every x402 deployment.

Fix a single asset and an initial budget $B_0 \in \mathbb{N}$. A state contains available balance B, a finite partial map R from reservation identifiers to amounts, a set N of consumed identifiers, and cumulative settled debits D. We impose

$$B + \sum_{j \in \mathrm{dom}(R)} R(j) + D = B_0.$$

All amounts are nonnegative integers. Reservation identifiers are scoped to the relevant principal and budget. The use of a single asset avoids introducing a hidden exchange-rate assumption.

8.1 State transitions

A reservation of amount c requires an unused identifier j, no current reservation for j, and $c \leq B$. It changes B to $B - c$ and records $R(j) = c$. A cancellation of an unsettled reservation removes its entry and restores its amount to B. Cancellation marks the identifier consumed in our machine, so a delayed settlement cannot reinterpret it as a newly created reservation.

Settlement of an existing reservation removes its entry, adds its amount to D, and records the identifier in N. An identifier already consumed cannot settle again. Attempts that fail these conditions leave the accounting state unchanged. The permitted external settlement mechanism and its evidence must be connected to this abstract transition by an implementation proof.

A canceled reservation and a settled reservation are different cases. Returning the available amount after settlement would violate the invariant unless a separate, justified refund transition also changed the accounting model. Our machine contains no refund transition. An application requiring refunds must add it, state how credited amounts are funded, and prove an extended conservation rule.

8.2 Theorem: budget preservation

Starting from $(B, R, N, D) = (B_0, \emptyset, \emptyset, 0)$, every finite sequence of the specified transitions preserves the displayed invariant and satisfies $D \leq B_0$.

Proof. Initially the equality is immediate. Reserving c subtracts c from the available component and adds c to the reservation sum. Canceling performs the opposite changes. Settling subtracts c from the reservation sum and adds c to the debit component. A rejected action changes nothing. Each transition therefore preserves the sum, and induction handles any finite execution. Since the other components remain nonnegative, $D \leq B_0$. □

This theorem proves an accounting property. It does not prove that a provider delivered a useful answer, that a chain reached a particular finality threshold, or that an authorized payment was economically rational. Those are separate postconditions. A provider can receive payment and then fail to deliver while the budget invariant remains perfectly true.

8.3 Atomicity and retries

The reservation check and its update must be atomic with respect to competing spend authority. Consider a budget of five units and two concurrent four-unit requests. Each process can observe five before either writes. If both independently accept, their combined commitment is eight. The proof above does not apply because the implementation has not produced a sequence of the specified atomic transitions.

A database transaction, a serialized actor, or another verified coordination mechanism can provide the required linearization. Merely recording both events after the fact cannot repair a prior promise that overspent the account. A distributed implementation must also determine which component is authoritative during a partition, rather than treat several divergent balances as simultaneously spendable.

Retries require separate treatment of job identity and payment identity. Repeating a settlement with the same consumed identifier should be rejected or recognized without another debit. Creating a fresh identifier for each retry can legitimately incur additional charges, but those charges must remain within the reserved budget. A valid receipt for a payment also does not imply that the corresponding computation completed exactly once.

These distinctions explain why agent autonomy requires an explicit capability boundary around spending. The question is not simply whether software can construct a transaction. It is whether the software can construct only transactions authorized by the principal's policy, whether concurrent branches share a sound budget, and whether failure paths remain within that policy.

9. Decentralization as a question about veto domains

Counting endpoints does not measure independent control. Several model providers can depend on one gateway, one identity service, one settlement component, or one underlying compute operator. Conversely, a locally executable alternative can remove an external veto even if it attracts few requests in ordinary operation.

Fix a finite set $\mathcal{D}$ of external trust or failure domains and a finite family $\mathcal{F}$ of currently feasible complete routes. Each route p has a set $\delta(p) \subseteq \mathcal{D}$ of domains whose cooperation it requires. "Feasible" means that the route has already passed the selected type, policy, resource, and compatibility checks. A hypothetical route violating the budget cannot be counted as an available escape path.

For a coalition $C \subseteq \mathcal{D}$, remove every route requiring a domain in C. The coalition has a veto exactly when

$$\forall p \in \mathcal{F}, \qquad C \cap \delta(p) \neq \emptyset.$$

Define $\kappa(\mathcal{F})$ as the minimum size of such a coalition. Set $\kappa = 0$ when no route is feasible, and $\kappa = \infty$ when some feasible route needs no external domain. This is a finite hitting-set formulation. If routes come from a dependency graph, it becomes a cut problem with shared domain labels.

9.1 Proposition: a bounded coalition criterion

For nonempty $\mathcal{F}$ and a nonnegative integer k, every coalition of at most k modeled domains leaves a feasible route available if and only if $\kappa(\mathcal{F}) > k$.

Proof. A coalition removes every route precisely when it intersects every route's required-domain set. Such coalitions are exactly the hitting sets in the definition of κ. There is no such coalition of size at most k precisely when the minimum hitting-set size is greater than k. The convention for an external-domain-free route gives the same result. □

The proposition is a characterization, not a new general theorem of network theory. Its value is the question it forces an evaluation to answer: what are the actual shared domains of control? Replacing one endpoint by many endpoints behind the same mandatory gateway leaves a one-domain veto. Distinct brands or wallet addresses are not evidence of independence.

This model concerns removal of cooperation. It does not yet model capacity, correlated economic incentives, service degradation, or strategic price changes. A route that remains syntactically present but exceeds its capacity cannot satisfy a strong availability objective. A practical extension should recompute the feasible family after each relevant change, or enrich routes with quantitative availability conditions.

The domain assignment also carries epistemic uncertainty. If two apparently separate providers secretly share a controlling operator, the calculated resilience can be overstated. One can represent several possible assignments and require a property under all of them. That conservative extension replaces a single attractive diagram with a family of diagrams justified by the evidence actually available.

9.2 Prices and economic interpretation

A zero usage fee does not logically imply that information is being sold. A donated local service and a service funded through data monetization can display the same price while having different information flows. Conversely, charging a fee does not establish that a service refrains from secondary use. An information-flow certificate must come from the execution and enforcement model.

Nor does a transparent payment establish competitive pricing. Several providers can publish their prices while sharing a bottleneck or following the same incentive. A claim about competition needs a game or market model with its own assumptions. Our resource semantics can record prices and veto domains, but it does not derive an equilibrium or a welfare conclusion from those records alone.

This restraint improves the usefulness of the original infrastructure question. It permits direct comparison of architectures on explicitly selected dimensions: intervention rights, disclosure, cost, substitutability, and external vetoes. Those dimensions can support an economic analysis later, without smuggling that analysis into definitions of openness or ownership.

10. A worked deployment with an explicit failure analysis

Consider an agent arranging a meeting from private calendar records. This is a synthetic example, not a benchmark of any named service. The user wants an acceptable suggestion, permits disclosure of one availability bit, and authorizes a total of eight payment units. The user does not permit disclosure of an internal note associated with the calendar.

Represent a private record as $x = (a, h) \in \{0, 1\}^2$. The bit a encodes the permitted availability summary. The bit h encodes the protected note. Allowed disclosure is the projection $(a, h) \mapsto$

a. This is a small model, but it contains the essential difference between disclosing task-relevant information and disclosing the entire resource from which that information was extracted.

10.1 Four candidate routes

The candidate planner has the following declared routes. All payment amounts are synthetic totals including any modeled fees.

Route	Data sent beyond the user boundary	Total units	External domains	Eligibility
Local	None	9	None	Exceeds the selected budget
Raw hosted	Both calendar bits	4	Hosted operator	Violates the disclosure policy
Minimized A	Availability bit only	6	Gateway, model A, settler A	Eligible under stated assumptions
Minimized B	Availability bit only	8	Gateway, model B, settler A	Eligible under stated assumptions

This table does not claim that a local computation must be more expensive. The numbers are chosen to expose the interaction between constraints. A route can improve one property while failing another, so a planner cannot rank all candidates by price alone. The locally controlled route would become eligible under a larger budget, and the raw hosted route would become eligible only under a changed disclosure policy or a stronger confidentiality mechanism.

For each minimized route, the declared confidentiality assumption says that every remote observer-visible event depends only on a and public history. The model must include any selected service identifier, request length, timing category, error, and settled amount that the observer can see. If the implementation cannot support that assumption, the route loses the corresponding certificate.

10.2 A complete finite transcript

Choose Minimized A before inspecting the protected note. A local step extracts a. The remote step returns a recommendation bit $1 - a$. Settlement records a constant six-unit debit. In the simplified deterministic observer model, the complete view is

$$v(a, h) = (a, 1 - a, 6).$$

It factors through a by the map $a \mapsto (a, 1-a, 6)$. The private note is absent from the entire modeled view. This is stronger than checking that the final recommendation omits the note, because it also checks the request and the financial record.

Now change the amount to $6 + h$, keeping all text fields unchanged. The view no longer factors through a: for fixed availability, an observer distinguishes the private note from the charge. This

is a concrete financial side channel. The finite checker includes both cases, so a reader can see a passing factorization and a nearby failing design.

A similar failure occurs if the planner selects Minimized A when $h = 0$ and Minimized B when $h = 1$, and the observer can see the provider or price. The local extraction step can be correct while the orchestration policy leaks. Therefore the controller's routing and retry choices belong inside the privacy theorem's conditional hypothesis.

10.3 State and accounting certificates

An admissible initial state has budget eight, an empty reservation map, the calendar under local authority, and a valid permission to send the summary. The local extraction span preserves the budget and establishes that the outbound payload equals a. The remote span assumes that its modeled uses remain within the asserted permission and produces the recommendation.

The payment component reserves six units before dispatch. The remaining available balance becomes two. Upon settlement, the reservation disappears and cumulative settled debits become six. If dispatch fails before settlement, cancellation restores eight. If computation fails after settlement, the account can remain charged six; the application must handle that outcome without pretending that conservation proves successful delivery.

Every represented branch satisfies the budget invariant. A retry using a new six-unit reservation is rejected after the first debit because only two units remain. A retry with a consumed payment identifier is rejected by the consumed-identifier check. A separate policy could authorize an additional budget, but that would be a new authority event, not a consequence of the original eight-unit grant.

The workflow's postcondition should therefore have a tagged outcome. A success tag carries a recommendation and the relevant certificates. A failure tag carries its observed cost and an error report. Both tags preserve spending authority and the stated disclosure condition. A specification requiring success on every environment behavior would need stronger service and scheduling assumptions than this example supplies.

10.4 The hidden common gateway

The two eligible routes both require the gateway and settler A. Removing either shared domain removes every eligible route. Hence the domain-cut value is one, despite the presence of two model names. The rejected nine-unit local route cannot repair this fact while the eight-unit budget remains in force.

Now replace Minimized B with a direct route through model B and an independent settler B, retaining its eight-unit total and the same privacy contract. Its external-domain set is disjoint from Minimized A's set. No single domain intersects both routes, while choosing one domain from each route blocks both. The cut value becomes two.

The example shows why decentralization must be evaluated after policy and resource filtering. It also shows how a design change can have an exact stated effect. Under the modeled domain assignment and availability assumptions, the second design survives one external-domain withdrawal. It does not establish resilience against two withdrawals, hidden common ownership, congestion, or a privacy failure in either route.

10.5 Migration and continued operation

Suppose the user wants to move the calendar parser and stored memory to another deployment. A file export containing the calendar is only one component. The migration must preserve the meaning of the availability extractor, the allowed-disclosure map, the budget principal, the unspent balance, and the interpretation of existing reservations and consumed identifiers.

Duplicating an unspent balance into two independently spendable deployments would violate the intended authority model. A sound migration either moves the authoritative ledger, partitions the remaining budget, or maintains a single shared spending authority. Similarly, exporting the calendar while forgetting its disclosure predicate can produce a deployment that retains useful data but loses its privacy certificate.

The strategy-transport result in Section 7 states a sufficient mathematical condition for preserving the selected intervention abilities. In practice, that condition can be decomposed into explicit migration obligations and checked against the particular implementation. The higher-categorical viewpoint is useful because several migrations may preserve the same information through different representations, while their compatibility still requires evidence.

11. Implementation, verification, and publication limits

A practical implementation should retain a typed intermediate representation with state boundaries, terminal outcomes, effect records, resource units, and referenced assumptions. The intermediate representation should distinguish a proved fact, a runtime-checked fact, an externally attested claim, and an unverified assumption. Treating those four statuses as the same Boolean field would make the apparent certificate stronger than its evidence.

An initial implementation can use finite state spaces and bounded traces. This makes it possible to enumerate executions, inspect every terminal branch, and compare observer views exactly. More scalable versions can replace exhaustive enumeration with symbolic reasoning, abstract interpretation, or proof assistants. Each replacement needs a soundness argument connecting the implementation's approximation to the semantic condition it claims to establish.

11.1 The executable companion

The accompanying standard-library Python program checks a finite preservation example, deterministic factorization, the correlated-marginal counterexample, exact conditional composition, a concrete approximate-composition bound, the uncontrolled-copy obstruction, the regular-action section obstruction, reservation accounting, and domain cuts. It also evaluates the synthetic meeting routes.

The checks use exact rational probabilities. A passing result therefore does not depend on sampling tolerance or a favorable random seed. Accounting exploration enumerates bounded action sequences. Domain cuts are found by exhaustive subset enumeration, which is suitable for the small examples and deliberately not presented as an efficient general optimizer.

These checks have a limited evidential role. They detect inconsistencies in the finite illustrations and make counterexamples reproducible. They do not prove the general higher-topos theorems, verify a compiler, validate hardware assumptions, or establish that an external implementation matches the model. The mathematical proofs remain written proofs. A later Lean or other proof-assistant development would be a separate artifact requiring its own kernel and assumption audit.

11.2 What a runtime monitor can and cannot establish

A monitor can reject a request exceeding a remaining budget when it sees all relevant spend actions and controls their execution. It can reject an outbound payload violating an inspectable schema. It can check that a required certificate is present. These are useful enforcement mechanisms, but they rely on complete mediation of the actions they govern.

A monitor observing one execution cannot generally establish a relation between all executions with different secret inputs. Noninterference is such a relation. The correlated-marginal example demonstrates why checking each visible event in isolation is inadequate. One needs a suitable static proof, a sound restrictive runtime discipline, or another justified verification method for the stated model.

The distinction between policy and enforcement mechanism is well established in security work, including the characterization of policies enforced by execution monitoring [7]. Our deployment contracts should state whether a condition is enforced directly, inherited from a proved component, or assumed about an external environment. This makes a guarantee auditable instead of merely attaching a reassuring label to a service.

11.3 Open technical work

The first extension is a proof-assistant formalization of the finite contract language and its preservation theorem. A suitable development would separate pure syntax, finite-state denotation, resource annotations, and a connection to executable runtimes. Proof erasure must not remove data needed to enforce capabilities or reservations.

The second extension is a rigorous internal probabilistic semantics. The finite kernel layer used here should be related to a chosen distribution construction in a specified higher topos, with explicit conditions for conditioning and sequential composition. It would be incorrect to assume that every probability construction automatically commutes with every pullback or supports every desired disintegration.

The third extension is temporal and strategic. Infinite-horizon availability, fairness, eventual deletion, and recovery after network partitions require stronger models than bounded witnesses. A coinductive or temporal semantics should distinguish the existence of a favorable infinite run from a strategy that guarantees progress under the declared scheduler assumptions.

The fourth extension is an evidence model for infrastructure independence. Domain labels currently enter as assumptions. A practical system should retain provenance for those labels, acknowledge uncertain shared control, and recompute resilience under plausible correlated failures. That work connects the semantic theory to operational measurements without confusing a declared graph with the physical and institutional network it represents.

11.4 The resulting thesis

The precise thesis is that access, control, privacy, and continuation are different properties of structured executions. The appropriate semantics retains the principals who can intervene, the observations that cross a boundary, the resources consumed along every represented branch, and the compatibility witnesses needed to move between deployments.

Within that semantics, certified composition preserves the policies it actually covers. Conditional observer-view certificates support finite privacy composition. Coherent local structures descend,

while unrelated local choices do not supply a global strategy. Uncontrolled information copies obstruct the strongest erasure goals. Spending requires a separate invariant, and independent continuation requires routes that avoid shared external vetoes.

These statements provide criteria for evaluating a proposed agentic infrastructure. Whether a particular network satisfies them is an empirical and implementation question. The theory makes that question precise enough to investigate.

Authorship, automation, and conflicts

This manuscript was prepared at a human researcher's direction with generative AI assistance for research synthesis, mathematical exposition, artifact construction, and finite-example checking. The anonymous byline does not claim an affiliation, an independent referee report, or an external audit. The submitting researcher remains responsible for reviewing the manuscript, its assumptions, and its publication terms.

The source organization is the subject of a critical formal interpretation and has not endorsed this manuscript. No affiliation, compensation, or investment position is asserted. The examples are synthetic. No live purchases, financial transactions, confidential-data transfers, or production deployment experiments were performed for this work.

References

1. Open Protocol Labs. *Own the Rails.* X article associated with post 2117325212597567961, 2 November 2026. Canonical post; article. Retrieved through a public post representation. This source supplies the research motivation; its numerical and commercial claims are not theorem premises.

2. Jacob Lurie. *Higher Topos Theory.* Annals of Mathematics Studies 170, Princeton University Press, 2009. arXiv:math/0608040; author's full text. Background for slices, truncation, effective epimorphisms, and higher descent.

3. Rune Haugseng. *Iterated spans and classical topological field theories.* Mathematische Zeitschrift 289, 1427-1488, 2018. arXiv:1409.0837. Background on higher correspondence constructions; its field-theoretic results are not claims about agentic infrastructure.

4. x402 project contributors. *X402 Protocol Specification*, version 2. Official specification repository, accessed 2 November 2026. A mutable technical specification; the abstract accounting machine in Section 8 is an application-level proposal, not an attributed protocol guarantee.

5. Joseph A. Goguen and José Meseguer. *Security Policies and Security Models.* Proceedings of the 1982 IEEE Symposium on Security and Privacy, 11-20, 1982. Full text hosted by Purdue University. Background for explicit policy and noninterference reasoning.

6. Tobias Fritz. *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics.* Advances in Mathematics 370, 107239, 2020. arXiv:1908.07021. Background for categorical probability; this manuscript proves only the stated finite-kernel composition results.

7. Fred B. Schneider. *Enforceable Security Policies.* ACM Transactions on Information and System Security 3(1), 30-50, 2000. Author's full text. Background for distinguishing specified policies from the power of execution-monitoring mechanisms.