

QVTM Formal Foundations

Mathematical Foundations of Trust Continuity

Version: 1.0 **Status:** Final Release — Frozen **Language:** English **Document Class:** Public Formal Specification **Authors:** Fernando Guerrero; Patricio Guerrero **Date:** July 2026

Normative Dependencies

This document derives its mathematical model exclusively from the frozen first-generation QVTM documentation:

- QVTM Position Paper v1.0;
- QVTM White Paper v1.0;
- QVTM Protocol Specification v1.0;
- QVTM SDK Reference v1.0.

The QVTM Protocol Specification is authoritative for externally observable protocol behavior. The White Paper is authoritative for the reference architecture where the Protocol Specification does not define behavior. The SDK Reference is authoritative only for the public integration surface. The Position Paper supplies the scientific motivation and conceptual origin of Trust Continuity.

This document introduces no protocol operation, protocol state, transition, primitive, evaluation outcome, or implementation requirement not already established by those frozen documents.

Document Purpose

This specification provides the canonical mathematical model used to reason about QVTM. It formalizes identity, Trust State, Continuity Evidence, Continuity Evidence Packages (CEPs), Continuity Graphs, Continuity Paths, Protocol State, Protocol Operations, synchronization, recovery, invariants, safety, progress, and internal model correctness.

It does not prove protocol security, cryptographic security, implementation correctness, endpoint integrity, application authorization correctness, or resistance to an adversary. Those questions belong to the future QVTM Security Analysis and QVTM Cryptographic Analysis.

Part I — Mathematical Foundations

Part I establishes the notation, semantic boundary, and static mathematical structures used throughout the specification.

1. Introduction

Chapter Principle

This chapter fixes the purpose, authority, and method of the QVTM Formal Foundations.

1.1 Objective

The objective is to define a mathematical structure whose valid executions represent the observable semantics frozen by the first-generation QVTM documentation. The model shall be sufficiently precise to support later security, cryptographic, certification, compliance, and mechanized-verification work without exposing proprietary algorithms or redefining protocol behavior.

1.2 Formalization Boundary

The model distinguishes three levels:

1. a QVTM concept defined by frozen documentation;
2. its mathematical representation in this specification;
3. an implementation representation outside this specification.

An Observable Object represents an observable QVTM concept within the formal model. It is not the implementation object itself. Two implementation representations are semantically equivalent when they induce the same Observable Object and observable behavior.

1.3 Construction Order

The specification proceeds in the following order:

1. notation and semantic equivalence;

2. static objects and relations;
3. observable protocol state and evolution;
4. synchronization and recovery;
5. invariants, safety, progress, and correctness boundaries.

Definitions precede the statements that depend upon them. Proof dependencies are acyclic.

1.4 Proof Style

A **Proof** gives the complete argument required at the abstraction level of this specification. A **Proof Sketch** records the decisive mathematical dependencies and argument when a complete derivation would require a mechanized semantics, environmental assumptions, or lower-level protocol detail outside this document.

Neither form is a machine-checked proof. No result relies on Graph Puzzle scoring, proprietary correlation heuristics, undisclosed storage state, or cryptographic assumptions beyond those explicitly excluded by Chapter 19.

1.5 Reading Strategy

Readers concerned primarily with formal semantics should read Parts I through III in order. Readers concerned primarily with proof obligations should first review Chapters 2, 3, 9, 10, and 11, and then proceed to Part IV. Appendix A provides the notation reference, Appendices B and C index formal definitions and statements, Appendix D records documentary derivation and authority, and Appendix E records proof dependencies.

Chapter Summary

This chapter establishes the semantic and proof boundaries of the document.

2. Mathematical Notation

Chapter Principle

This chapter defines the notation used by the QVTM Formal Model.

2.1 Basic Notation

Notation	Meaning
$x \in X$	x is an element of X
$X \subseteq Y$	X is a subset of Y
$X \times Y$	Cartesian product
$\mathcal{P}(X)$	power set of X
X^*	finite sequences over X
X^ω	infinite sequences over X
$f: X \rightarrow Y$	total function
$f: X \rightharpoonup Y$	partial function
$R \subseteq X \times Y$	binary relation
$\text{dom}(f)$	domain of f
$\text{ran}(f)$	range of f
$\langle x_0, \dots, x_n \rangle$	finite sequence
$\bot$	absence of an optional result

2.2 Semantic Equivalence

Definition 2.1 (Semantic Equivalence)

For mathematical representations x and y , write

$$x \equiv y$$

when they carry the same normative meaning under the frozen QVTM documentation, even if their implementation encodings differ.

Mathematical equality $x=y$ denotes equality inside the formal model. Semantic equivalence is the representation-independent relation required by the Protocol Specification.

2.3 Finite Observable Collections

Unless a definition states otherwise, the evidence, CEP, Trust State, relationship, and evaluation collections contained in one observable protocol state are finite. This restriction models a bounded observation at one protocol instant; it does not impose a finite lifetime on a deployment.

2.4 Undefinedness

A partial function is undefined outside its domain. The model does not use null values or implementation sentinel values. Optional observable results use $\bot$ only when absence itself is part of the modeled result.

2.5 Trust Terminology

The following frozen terms are related but non-interchangeable.

Term	Formal meaning in this specification
Trust	The general architectural subject expressed through evidence-supported continuity semantics; it is not introduced as a separate scalar or score.
Trust State	An observable trust condition $t \in \mathcal{T}$ represented in the Continuity Graph.
Trust Evaluation	A Protocol Operation whose observable result $v \in \mathcal{V}$ has one outcome in $\mathcal{Y}$ and remains separate from application authorization.

Whenever this document uses “trust” without qualification, the surrounding formal object determines whether the reference is to a Trust State, a continuity relationship, or a Trust Evaluation result.

Chapter Summary

This chapter defines equality, semantic equivalence, collections, functions, relations, and undefinedness.

3. QVTM Formal Model

Chapter Principle

This chapter defines the formal model and its observable boundary.

Definition 3.1 (QVTM Formal Model)

The QVTM Formal Model is the structure

$$\mathcal{M}_{\mathrm{QVTM}} = (\mathcal{O}, \mathcal{R}, \mathcal{F}, \mathcal{S}, \mathcal{A}, \delta, \mathcal{S}_0),$$

where:

- $\mathcal{O}$ is the family of Observable Object spaces;
- $\mathcal{R}$ is the family of relations over those objects;
- $\mathcal{F}$ is the family of non-transition functions;
- $\mathcal{S}$ is the observable state space;
- $\mathcal{A}$ is the observable action space;
- δ is the partial transition function;
- $\mathcal{S}_0 \subseteq \mathcal{S}$ is the set of valid initial states.

Definition 3.2 (Observable Object)

An Observable Object is an element of a space in $\mathcal{O}$ representing a concept or result visible at a protocol boundary. “Mathematical object” is used only when discussing the metamodel or mathematical representation generally.

Definition 3.3 (Model Interpretation)

An implementation interpretation is a mapping

$$\llbracket \cdot \rrbracket : \text{Implementation Representations} \rightarrow \mathcal{O} \cup \mathcal{S} \cup \mathcal{A}.$$

Two implementations are observationally equivalent for this specification when semantically equivalent inputs induce semantically equivalent states and results.

3.1 Model Independence

The model is independent of programming language, storage layout, serialization, transport, deployment topology, cryptographic provider, and internal Graph Puzzle algorithm. Independence does not remove normative profile constraints; it removes dependence on one implementation of those constraints.

Chapter Summary

This chapter defines the single formal structure used by all later chapters.

4. Formal Identity Model

Chapter Principle

This chapter defines identity only to the extent required by frozen QVTM semantics.

Definition 4.1 (Identity Space)

Let $\mathcal{I}$ be the set of identities that may participate in QVTM continuity relationships. An identity is an element $i \in \mathcal{I}$.

The space may contain users, operators, devices, applications, services, autonomous agents, or other profile-defined participants. This definition does not impose a universal identity provider, credential format, or authentication ceremony.

Definition 4.2 (Identity Context)

Let $\mathcal{K}$ be the set of defined protocol contexts. An identity context relation is

$$\text{inContext} \subseteq \mathcal{I} \times \mathcal{K}.$$

Identity equality is ordinary mathematical equality. Credential replacement, device replacement, and recovery do not imply identity equality or inequality by themselves; their continuity significance is represented by evidence and continuity relationships.

Chapter Summary

Identity is a mathematical participant reference scoped by observable protocol context, not a credential implementation.

Part II — Continuity Structures

Part II formalizes Trust States, Continuity Evidence, CEPs, Continuity Graphs, Continuity Paths, and Trust Evaluation outputs.

5. Formal Continuity Model

Chapter Principle

This chapter formalizes continuity as supported relationships among Trust States rather than as a standalone scalar object.

Definition 5.1 (Trust State Space)

Let $\mathcal{T}$ be the set of Trust States. A Trust State $t \in \mathcal{T}$ is an observable trust condition evaluated within a defined protocol context.

Definition 5.2 (Continuity Relationship)

A continuity relationship is an ordered pair of Trust States. Let

$$\mathcal{L} \subseteq \mathcal{T} \times \mathcal{T}$$

be the set of continuity relationships represented by the protocol.

Membership in $\mathcal{L}$ does not by itself establish that the relationship is valid; support is supplied by Continuity Evidence.

Definition 5.3 (Continuity Path)

A Continuity Path is a finite sequence

$$\pi = \langle t_0, t_1, \dots, t_n \rangle \in \mathcal{T}^*$$

such that $(t_j, t_{j+1}) \in \mathcal{L}$ for every $0 \leq j < n$.

Definition 5.4 (Trust Continuity)

For a defined evaluation context $k \in \mathcal{K}$, Trust Continuity is the property that the relevant trust relationship is supported by independently verifiable Continuity Evidence along at least one applicable Continuity Path under the applicable policy and Deployment Profile constraints.

This definition states the public mathematical property. It does not disclose or prescribe the internal Graph Puzzle algorithm used to derive candidate paths.

Definition 5.5 (Trust Discontinuity)

A Trust Discontinuity exists for context k when the applicable evaluation determines that available evidence does not support continuity for that context. Insufficient evidence is distinct from demonstrated discontinuity and is represented by an Indeterminate evaluation outcome.

Chapter Summary

Continuity is a path property over evidence-supported Trust State relationships.

6. Formal Continuity Evidence Model

Chapter Principle

This chapter formalizes immutable, independently verifiable Continuity Evidence.

Definition 6.1 (Continuity Evidence Space)

Let $\mathcal{E}$ be the set of valid Continuity Evidence objects. Each $e \in \mathcal{E}$ represents one continuity-relevant event, condition, or relationship and has the required observable properties defined by the Protocol Specification: identifier, type, creation context, integrity information, and semantic version.

Definition 6.2 (Evidence Context)

The evidence-context function is

$$\text{ctx} : \mathcal{E} \rightarrow \mathcal{K}. \quad \square$$

No requirement is imposed that every evidence object reference exactly one identity or exactly one continuity relationship. The creation context determines the applicable interpretation.

Definition 6.3 (Evidence Support Relation)

The evidence support relation is

$$\text{supports} \subseteq \mathcal{E} \times (\mathcal{T} \cup \mathcal{L}). \quad \square$$

An evidence object may support a Trust State, a continuity relationship, or both, according to its normative semantics.

Definition 6.4 (Evidence Identity)

The stable identity of an evidence object is represented by

$$\text{id} : \mathcal{E} \rightarrow \mathcal{D}_{\mathcal{E}}, \quad \square$$

where $\mathcal{D}_{\mathcal{E}}$ is the evidence-identifier space and $\text{id} : \mathcal{E} \rightarrow \mathcal{D}_{\mathcal{E}}$ is injective within the applicable Deployment Profile.

Definition 6.5 (Evidence Immutability)

For any execution states Σ_j and Σ_k , if evidence with identifier d occurs in both states, then the corresponding evidence objects are semantically equivalent:

$$\text{id}(e_j) = \text{id}(e_k) \rightarrow e_j \equiv e_k. \quad \square$$

Chapter Summary

Evidence is immutable, context-bound, independently verifiable information that supports Trust States or continuity relationships.

7. Formal CEP Model

Chapter Principle

This chapter formalizes the CEP as the immutable semantic container for one or more evidence objects.

Definition 7.1 (CEP Space)

Let $\mathcal{P}$ be the set of valid Continuity Evidence Packages.

Definition 7.2 (CEP Evidence Function)

The contained-evidence function is

$$\text{ev} : \mathcal{P} \rightarrow \mathcal{P}(\mathcal{E}) \setminus \{\varnothing\}.$$

Thus every valid CEP contains at least one Continuity Evidence object.

Definition 7.3 (CEP Identity)

The CEP identifier function is

$$\text{id} : \mathcal{P} \rightarrow \mathcal{D},$$

and is injective within the applicable Deployment Profile.

Definition 7.4 (CEP Context)

The CEP context function is

$$\text{ctx} : \mathcal{P} \rightarrow \mathcal{K}.$$

The context includes the protocol version and applicable Deployment Profile required for deterministic interpretation.

Definition 7.5 (CEP Immutability)

If CEPs with the same stable identity occur in two observable states, then they are semantically equivalent and contain semantically equivalent evidence:

$$\begin{aligned} \text{id}_{\{P\}}(p_j) = \text{id}_{\{P\}}(p_k) \rightarrow p_j \equiv p_k \text{ and } \\ \text{ev}(p_j) \equiv \text{ev}(p_k). \end{aligned}$$

Definition 7.6 (Evidence Encapsulation Relation)

Define

$$\text{encapsulates} \subseteq \mathcal{P} \times \mathcal{E}$$

by $(p, e) \in \text{encapsulates}$ iff $e \in \text{ev}(p)$.

A CEP may contain evidence relevant to multiple Trust States, continuity relationships, or continuity domains. The model therefore does not assign each CEP to exactly one continuity object.

Chapter Summary

A CEP is a non-empty immutable evidence container with stable identity and explicit interpretation context.

8. Formal Continuity Graph and Trust Evaluation Model

Chapter Principle

This chapter formalizes the public input and output boundary of correlation without disclosing Graph Puzzle internals.

Definition 8.1 (Continuity Graph)

A Continuity Graph is a tuple

$$G = (T_G, L_G, S_G)$$

where $T_G \subseteq \mathcal{T}$, $L_G \subseteq \mathcal{L}$, and

$$S_G \subseteq \mathcal{E} \times (T_G \cup L_G)$$

is the restricted evidence support relation.

Let $\mathcal{G}$ denote the set of valid Continuity Graphs.

Definition 8.2 (Graph Extension)

For $G_1 = (T_1, L_1, S_1)$ and $G_2 = (T_2, L_2, S_2)$, write

$G_1 \preceq G_2$

iff $T_1 \subseteq T_2$, $L_1 \subseteq L_2$, and $S_1 \subseteq S_2$.

This relation represents append-only graph extension; it does not assert that every newly observed relationship is valid for every evaluation context.

Definition 8.3 (Evaluation Outcome Space)

The evaluation outcome space is

$\mathcal{Y} = \{\text{Continuous}, \text{Conditional}, \text{Indeterminate}, \text{Discontinuous}\}$.

Conditional is the formal label for the frozen outcome “Continuous with Conditions.”

Definition 8.4 (Evaluation Context)

Let $\mathcal{Q}$ be the set of evaluation requests, each identifying the evaluated context, purpose, policy constraints, Deployment Profile constraints, and referenced continuity inputs required by the public contract.

Definition 8.5 (Trust Evaluation Result)

Let $\mathcal{V}$ be the set of observable Trust Evaluation results. Each $v \in \mathcal{V}$ has an outcome

$\text{outcome}: \mathcal{V} \rightarrow \mathcal{Y}$

and references its evaluation context and continuity inputs.

Definition 8.6 (Evaluation Boundary)

The observable evaluation function is

$\text{evaluate}: \mathcal{G} \times \mathcal{Q} \rightarrow \mathcal{V}$.

The function models only the deterministic observable result required by the Protocol Specification. Its internal correlation, scoring, path-selection, and heuristic mechanisms are outside the proof scope.

Definition 8.7 (Application Separation)

Let $\mathcal{B}$ be the set of application decisions. There is no identity function from $\mathcal{V}$ to $\mathcal{B}$. An application policy may consume $v \in \mathcal{V}$ to produce a decision, but the evaluation result and application decision are distinct Observable Objects.

8.1 Formal Dependency Workflow

Workflow Figure 1 summarizes the dependency structure established by Definitions 4.1 through 8.7. It is a semantic dependency view, not an execution algorithm.

Workflow Figure 1 — Formal dependency workflow

flowchart LR

I[Identity and protocol context]

TS[Trust States]

CE[Continuity Evidence]

CEP[CEP]

CG[Continuity Graph]

CP[Continuity Paths]

TE[Trust Evaluation]

AD[Application Decision]

I -->|scopes| TS

CE -->|supports| TS

CE -->|supports| CG

CE -->|encapsulated by| CEP

CEP -->|supplies validated evidence| CG

CG -->|contains| CP

CP -->|evaluated under context and policy| TE

TE -. informs but does not determine .-> AD

Caption. Identity and protocol context scope Trust States; Continuity Evidence supports states and relationships, CEPs preserve evidence, the Continuity Graph contains candidate paths, and Trust Evaluation informs—but does not determine—an application decision. The figure does not disclose Graph Puzzle internals.

Chapter Summary

The Continuity Graph correlates evidence-supported Trust States and relationships; Trust Evaluation exposes one of four frozen outcomes while remaining separate from application authorization.

9. Formal Observable State Model

Chapter Principle

This chapter composes the previously defined structures into one observable protocol state.

Definition 9.1 (Protocol State Space)

The protocol state space is

$$\mathcal{Q}_P = \{\text{Initialized}, \text{Enrolled}, \text{Operational}, \text{Synchronizing}, \text{Recovering}, \text{Suspended}, \text{Archived}\}$$

These are protocol states, not application states or Trust States.

Definition 9.2 (Observable Protocol State)

An observable protocol state is a tuple

$$\Sigma = (q, E, P, G, V)$$

where:

- $q \in \mathcal{Q}_P$ is the current Protocol State;
- $E \in \mathcal{P}_{\text{fin}}(\mathcal{E})$ is the preserved evidence set;
- $P \in \mathcal{P}_{\text{fin}}(\mathcal{P})$ is the preserved CEP set;
- $G \in \mathcal{G}$ is the observable Continuity Graph derived from available validated evidence;
- $V \in \mathcal{P}_{\text{fin}}(\mathcal{V})$ is the set of preserved observable Trust Evaluation results.

Let $\mathcal{S}$ be the set of all well-formed tuples satisfying the consistency conditions below.

Definition 9.3 (State Well-Formedness)

Define $\text{wellFormed} : \mathcal{S} \rightarrow \{\text{true}, \text{false}\}$. A state $\Sigma = (q, E, P, G, V)$ is well formed iff:

- every $p \in P$ satisfies $\text{ev}(p) \subseteq E$;
- every evidence object referenced by G belongs to E ;
- every evaluation result in V references a defined evaluation context and only continuity inputs available to the evaluation operation;
- the state components satisfy the Protocol State constraints applicable to q .

Definition 9.4 (State Projections)

The projections are π_q , π_E , π_P , π_G , and π_V , with their evident tuple meanings.

Definition 9.5 (Initial State Set)

$\mathcal{S}_0$ is the set of well-formed states with protocol state **Initialized** and no completed operational continuity establishment. The Protocol Specification determines the observable initialization context; this model does not prescribe an implementation representation.

9.1 State-Space Distinction

$\mathcal{S}$ contains all well-formed mathematical states. The reachable state set, defined in Chapter 11, contains only states obtainable from $\mathcal{S}_0$ through admissible Protocol Operations. The two sets are not interchangeable.

Chapter Summary

The observable state contains the normative Protocol State and finite preserved collections of evidence, CEPs, graph information, and evaluation results.

Part III — State Evolution

Part III formalizes Protocol Operations, valid state transitions, synchronization, and recovery.

10. Formal Protocol Operation Model

Chapter Principle

Protocol evolution occurs through bounded observable Protocol Operations.

Definition 10.1 (Protocol Operation Space)

The core Protocol Operation space contains

$\mathcal{O}_P = \{\text{Enrollment}, \text{Authentication}, \text{Synchronization}, \text{Recovery}, \text{TrustEvaluation}, \text{VersionNegotiation}\}$.

A future conformant version or Deployment Profile may extend the operation space only under the extension rules of the Protocol Specification.

Definition 10.2 (Observable Action)

An observable action $a \in \mathcal{A}$ is a tuple containing a Protocol Operation, its observable inputs, applicable protocol/profile context, and the success or defined failure disposition needed to determine observable state effects.

Definition 10.3 (Admissibility Predicate)

The admissibility predicate is

$\text{adm} : \mathcal{S} \times \mathcal{A} \rightarrow \{\text{true}, \text{false}\}$.

It is true precisely when the current Protocol State permits the operation and all normative preconditions are satisfied.

Definition 10.4 (Transition Function)

The transition function is partial:

$$\delta: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S},$$

with

$$\text{dom}(\delta) = \{(\Sigma, a) \mid \text{adm}(\Sigma, a) = \text{true}\}.$$

Definition 10.5 (Rejected Action Semantics)

For an inadmissible action, no successful state transition exists. A conformant implementation returns a defined error result and preserves previously valid Protocol State, evidence, and CEPs. When the applicable policy requires continuity-relevant rejection evidence, that evidence is introduced through a separate admissible failure-recording action rather than by mutating prior evidence.

Definition 10.6 (Observable Determinism)

For semantically equivalent states and actions,

$$\Sigma_1 \equiv \Sigma_2 \text{ and } a_1 \equiv a_2 \rightarrow \delta(\Sigma_1, a_1) \equiv \delta(\Sigma_2, a_2),$$

whenever both transitions are defined.

Chapter Summary

Protocol transitions are partial, preconditioned, failure-contained, and observably deterministic.

11. Formal Transition System

Chapter Principle

This chapter defines the valid Protocol State graph, executions, and reachability.

Definition 11.1 (Protocol State Transition Relation)

Let $\rightarrow \subseteq \mathcal{Q}_P \times \mathcal{Q}_P$ be exactly the frozen transition relation:

$$\begin{aligned}
& \leadsto = \{ (\& (\text{Initialized}, \text{Enrolled}), \\
& (\text{Enrolled}, \text{Operational}), \& (\text{Operational}, \text{Synchronizing}), \\
& (\text{Operational}, \text{Recovering}), (\text{Operational}, \text{Suspended}), \& \\
& (\text{Synchronizing}, \text{Operational}), (\text{Recovering}, \text{Operational}), \\
& (\text{Suspended}, \text{Operational}), \& (\text{Operational}, \text{Archived}), \\
& (\text{Recovering}, \text{Archived}) \}.
\end{aligned}$$

Definition 11.2 (State-Transition Compatibility)

For every defined transition $\Sigma' = \delta(\Sigma, a)$,

$$\pi_q(\Sigma), \pi_q(\Sigma') \in \leadsto$$

or $\pi_q(\Sigma) = \pi_q(\Sigma')$ when the Protocol Operation does not require a Protocol State change.

Definition 11.3 (Execution)

An execution is a finite or infinite alternating sequence

$$\xi = \langle \Sigma_0, a_0, \Sigma_1, a_1, \dots \rangle$$

such that $\Sigma_0 \in \mathcal{S}_0$ and $\Sigma_{j+1} = \delta(\Sigma_j, a_j)$ for every defined step.

Let $\mathcal{X}$ be the set of all executions.

Definition 11.4 (Reachable State)

A state Σ is reachable iff it occurs in some execution $\xi \in \mathcal{X}$. Let $\mathcal{S}_R \subseteq \mathcal{S}$ be the reachable state set.

Definition 11.5 (Archived Terminality)

No transition from **Archived** to **Operational**, **Synchronizing**, or **Recovering** belongs to $\leadsto$.

Historical evidence may remain verifiable after archival, but operational activity does not resume in the same protocol instance.

Chapter Summary

Reachability is defined from valid initial states and the exact frozen Protocol State transition relation.

12. Formal Synchronization Model

Chapter Principle

Synchronization exchanges and reconciles evidence and CEPs; it does not replicate private implementation state.

Definition 12.1 (Synchronization Input)

For local state Σ_L and authorized remote material represented by evidence set E_R and CEP set P_R , a synchronization input is

$$s = (E_R, P_R)$$

such that every transferred object is eligible under the applicable protocol context and synchronization authorization.

Definition 12.2 (Synchronization Result)

The synchronization operation produces an observable disposition partition over transferred objects, including accepted, duplicate, conflicting, rejected, or deferred treatment as applicable, together with a resulting correlation status.

Definition 12.3 (Synchronization Extension)

For a successful synchronization transition

$$\Sigma' = \delta(\Sigma, a_{\mathrm{sync}}),$$

the preserved collections satisfy

$$\pi_E(\Sigma) \subseteq \pi_E(\Sigma') \quad \text{and} \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma').$$

Accepted new material may extend those sets. Duplicate material does not create a second Observable Object with the same stable identity. Rejected material does not alter previously preserved material.

Definition 12.4 (Synchronization Graph Effect)

A successful synchronization may extend the observable Continuity Graph:

$$\pi_G(\Sigma) \preceq \pi_G(\Sigma').$$

A conflict remains observable and is not silently resolved by rewriting evidence.

12.1 Non-Simultaneity

Synchronization does not imply simultaneous execution, a global clock, a single total order, or global state replication. It reconciles eligible evidence and CEPs according to observable protocol semantics.

Proposition 12.1 (Synchronization Order Independence)

For a successful synchronization transition $\Sigma' = \delta(\Sigma, a_{\mathrm{sync}})$, the extension conditions

$$\pi_E(\Sigma) \subseteq \pi_E(\Sigma'), \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma'), \quad \pi_G(\Sigma) \preceq \pi_G(\Sigma')$$

do not imply a total temporal order over independently produced evidence beyond the ordering represented by the synchronized evidence and relationship context.

Proof

Definitions 12.3 and 12.4 constrain preservation and graph extension only. Section 12.1 explicitly excludes a global clock and total-order assumption. Therefore synchronization preserves represented ordering constraints without creating an additional global temporal order.

Chapter Summary

Synchronization is monotone evidence reconciliation with explicit dispositions and no destructive state replication.

13. Formal Recovery Model

Chapter Principle

Recovery is a first-class Protocol Operation that extends continuity history into a restored operational context.

Definition 13.1 (Recovery Request Space)

Let $\mathcal{W}$ be the set of recovery requests carrying the observable recovery context and evidence required by the applicable Deployment Profile.

Definition 13.2 (Recovery Validation Predicate)

The predicate

$\text{recoverable} : \mathcal{S} \times \mathcal{W} \rightarrow \{\text{true}, \text{false}\}$

is true when recoverable continuity history exists, sufficient Continuity Evidence is available, and the applicable recovery conditions are successfully validated.

Definition 13.3 (Successful Recovery Transition)

A successful recovery action is admissible only from a Protocol State that permits Recovery and produces a state Σ' such that:

1. $\pi_q(\Sigma') \in \{\text{Operational}, \text{Archived}\}$ as allowed by the frozen transition relation;
2. prior evidence and CEPs are preserved;
3. at least one new recovery Continuity Evidence object is present;
4. the Continuity Graph is extended rather than restarted.

Definition 13.4 (Failed Recovery)

A failed Recovery operation preserves prior Protocol State, evidence, and CEPs except for separately issued continuity-relevant failure evidence when required. Failure does not itself prove Trust Discontinuity; that conclusion belongs to Trust Evaluation.

Proposition 13.1 (Recovery by Extension)

For every successful recovery transition $\Sigma' = \delta(\Sigma, a_{\text{recovery}})$,

$\pi_E(\Sigma) \subsetneq \pi_E(\Sigma'), \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma'), \quad \pi_G(\Sigma) \preceq \pi_G(\Sigma').$

The strict evidence inclusion follows from the required creation of at least one new recovery Continuity Evidence object.

Proof

Definition 13.3 requires preservation of prior evidence and CEPs, addition of recovery evidence, and graph extension. These conditions establish the three relations directly.

Chapter Summary

Recovery is validated, append-only continuity extension, not state reconstruction from a CEP alone and not a restart of protocol history.

Part IV — Formal Properties

Part IV states and proves the principal internal properties of the public mathematical model.

14. Property Language

Chapter Principle

This chapter defines predicates over states, transitions, and executions.

14.1 Formal Statement Conventions

Formal statements are classified by their role in the proof structure. The classification denotes purpose rather than typography or prestige.

Statement class	Role in this specification
Definition	Introduces a formal object, relation, function, predicate, or boundary.
Axiom	States an assumption accepted within the declared proof boundary.
Invariant	States a property required to hold in every reachable state.
Proposition	Establishes an intermediate result of independent value.
Lemma	Establishes a supporting result used by a later theorem.
Theorem	Establishes a principal property of the model.
Meta-Theorem	Combines previously established results into a model-level conclusion.
Corollary	Records a result that follows immediately from an earlier formal statement.
Boundary	Delimits a claim that the model intentionally does not establish.

A **Proof** provides the complete argument required at this specification's abstraction level. A **Proof Sketch** identifies the decisive dependencies and reasoning when completion requires lower-level semantics, mechanization, or environmental assumptions outside the proof scope.

Definition 14.1 (State Predicate)

A state predicate is $S \rightarrow \{\text{true}, \text{false}\}$.

Definition 14.2 (Transition Predicate)

A transition predicate is

$$Q: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \{\text{true}, \text{false}\}.$$

Definition 14.3 (Execution Property)

An execution property is a predicate over $\mathcal{X}$.

Definition 14.4 (Invariant)

A state predicate P is an invariant iff it holds for every $\Sigma_0 \in \mathcal{S}_0$ and is preserved by every defined transition.

Definition 14.5 (Safety Property)

A safety property is an execution property whose violation has a finite bad prefix.

Definition 14.6 (Progress Property)

A progress property states that a defined admissible operation can produce its specified successor or defined failure disposition under the assumptions explicitly stated for that operation.

The document uses `progress` for the operational results proven here. Strong temporal liveness claims requiring fairness, scheduling, delivery, or eventuality assumptions are outside the current proof scope.

Chapter Summary

The property language separates invariants and safety from conditional operational progress.

15. Invariants

Chapter Principle

This chapter derives the protocol invariants frozen by the Protocol Specification.

Axiom 15.1 (Protocol Conformance Assumption)

Every modeled action and state transition satisfies the normative preconditions and observable semantics of the frozen QVTM Protocol Specification.

This is the sole conformance axiom. The following invariants are formal consequences or explicit mathematical formulations of frozen protocol invariants.

Invariant 15.1 (Evidence Immutability)

For every transition $\Sigma' = \Delta(\Sigma, a)$ and every evidence identifier present in both states, the corresponding evidence objects are semantically equivalent.

Invariant 15.2 (Append-Only Evidence)

For every defined transition,

$$\pi_E(\Sigma) \subseteq \pi_E(\Sigma').$$

Invariant 15.3 (CEP Preservation)

For every defined transition,

$$\pi_P(\Sigma) \subseteq \pi_P(\Sigma'),$$

and equal CEP identifiers imply semantic equivalence.

Invariant 15.4 (Observable Determinism)

Semantically equivalent admissible inputs produce semantically equivalent observable successor states and results.

Invariant 15.5 (Application Separation)

No Protocol Operation rewrites a Trust Evaluation result into an application authorization decision, and no application decision mutates preserved Continuity Evidence.

Invariant 15.6 (Failure Containment)

A rejected or failed operation does not corrupt previously valid Protocol State, Continuity Evidence, CEPs, or Continuity Graph information.

Invariant 15.7 (Cryptographic Independence)

No core state, path, evidence, CEP, or evaluation definition depends on one cryptographic algorithm generation. Algorithm identifiers and cryptographic contexts may occur as profile-bound observable data.

Invariant 15.8 (State Well-Formedness)

Every reachable state is well formed under Definition 9.3.

Lemma 15.1 (Encapsulation Preservation)

If $p \in \pi_P(\Sigma)$ and $e \in \text{ev}(p)$, then $e \in \pi_E(\Sigma)$. This property is preserved by every transition.

Proof Sketch

It holds by state well-formedness. CEP and evidence preservation prevent a later state from retaining p while deleting e .

Lemma 15.2 (Graph Evidence Availability)

Every evidence object referenced by the Continuity Graph of a reachable state belongs to that state's preserved evidence set.

Proof

Immediate from state well-formedness and append-only preservation.

Theorem 15.1 (Invariant Preservation)

Every invariant stated in Sections 15.1–15.8 holds for every reachable state.

Proof Sketch

The invariants hold in every valid initial state by construction. Axiom 15.1 restricts transitions to conformant observable semantics. The Protocol Specification requires immutability, append-only continuity, CEP preservation, determinism, application separation, failure containment, and cryptographic independence. Definitions 9.3, 10.4, and 10.5 preserve well-formedness across successful, rejected, and failed operations. Induction over execution length yields the result.

Chapter Summary

The invariant set now matches the frozen protocol invariants rather than merely asserting closure of a Cartesian product.

16. Safety Properties

Chapter Principle

Safety states what no conformant execution can do.

Theorem 16.1 (No Evidence Rewrite)

For every execution $\xi \in \text{mathcal{X}}$, all states Σ_i, Σ_j occurring in ξ , and evidence objects $e_i \in \pi_E(\Sigma_i)$ and $e_j \in \pi_E(\Sigma_j)$,

$$\text{id}_E(e_i) = \text{id}_E(e_j) \rightarrow e_i \equiv e_j.$$

Proof Sketch

Assume the earliest violating state. The preceding state contains either the original object or neither object. Evidence immutability and append-only preservation prohibit insertion of a non-equivalent replacement under the same identifier. Contradiction.

Theorem 16.2 (No CEP Rewrite or Silent Deletion)

For every execution $\xi \in \text{mathcal{X}}$ and states Σ_i, Σ_j occurring in ξ with $i \leq j$,

$$p \in \pi_P(\Sigma_i) \rightarrow \exists p' \in \pi_P(\Sigma_j): \text{id}_P(p) = \text{id}_P(p') \text{ and } p' \equiv p.$$

Proof

Directly from CEP preservation and stable CEP identity.

Theorem 16.3 (No Invalid Protocol State Transition)

For every defined reachable transition $\Sigma' = \delta(\Sigma, a)$,

$$(\pi_q(\Sigma), \pi_q(\Sigma')) \rightarrow \text{quad} \text{ or } \text{quad} \pi_q(\Sigma) = \pi_q(\Sigma').$$

The equality case is permitted only when the Protocol Operation does not require a Protocol State change.

Proof

Definition 11.2 is a transition well-formedness condition enforced by the conformance axiom.

Theorem 16.4 (No Partial Commit on Rejection)

For all $\Sigma \in \text{mathcal{S}}$ and $a \in \text{mathcal{A}}$,

$$\text{adm}(\Sigma, a) = \text{false} \rightarrow (\Sigma, a) \notin \text{dom}(\delta).$$

Consequently, an inadmissible action cannot produce a partially committed successful successor state.

Proof

The transition function is undefined for inadmissible action pairs. Defined error handling preserves prior valid state. Any separately recorded failure evidence requires its own admissible action.

Theorem 16.5 (Synchronization Non-Destruction)

For every successful synchronization transition $\Sigma' = \delta(\Sigma, a_{\mathrm{sync}})$,

$$\pi_E(\Sigma) \subseteq \pi_E(\Sigma') \quad \text{and} \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma').$$

Synchronization therefore cannot delete or rewrite previously preserved evidence or CEPs.

Proof Sketch

Definition 12.3 and the append-only invariants require set extension. Conflicting or rejected material remains a disposition, not a destructive update.

Theorem 16.6 (Recovery History Preservation)

For every successful recovery transition $\Sigma' = \delta(\Sigma, a_{\mathrm{recovery}})$,

$$\pi_E(\Sigma) \subsetneq \pi_E(\Sigma') \quad \text{and} \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma') \\ \pi_G(\Sigma) \preceq \pi_G(\Sigma').$$

A successful recovery therefore cannot restart or replace prior continuity history.

Proof Sketch

Definition 13.3 requires prior evidence and CEP preservation and graph extension. Therefore the recovered state contains the prior history plus recovery evidence.

Theorem 16.7 (Authorization Boundary Safety)

For every Trust Evaluation result $v \in \mathcal{V}$ and application decision $b \in \mathcal{B}$,

$$v \neq b.$$

No identity mapping from $\mathcal{V}$ to $\mathcal{B}$ is defined by the QVTM Formal Model.

Proof

Definition 8.7 separates the result and decision spaces, and Invariant 15.5 preserves that separation.

Chapter Summary

The model rules out evidence rewrite, CEP rewrite, invalid state movement, partial commits, destructive synchronization, history-replacing recovery, and evaluation/authorization collapse.

17. Progress Properties

Chapter Principle

This chapter establishes conditional progress without claiming fairness or guaranteed eventual delivery.

Theorem 17.1 (Enrollment Progress)

If a_{enroll} is admissible in Σ with $\pi_q(\Sigma) = \text{Initialized}$ and completes successfully, then for $\Sigma' = \delta(\Sigma, a_{\mathrm{enroll}})$,

$$\pi_q(\Sigma') = \text{Enrolled}, \quad \pi_E(\Sigma') \neq \text{varnothing}, \quad \pi_P(\Sigma') \neq \text{varnothing}.$$

Proof Sketch

These are the required observable outputs and state transition of the frozen Enrollment contract.

Theorem 17.2 (Synchronization Progress)

For every reachable $\Sigma \in \mathcal{S}_R$ and admissible authorized Synchronization action a_{sync} , if $\Sigma' = \delta(\Sigma, a_{\mathrm{sync}})$ is defined, then every transferred eligible object receives an observable disposition and

$$\pi_E(\Sigma) \subseteq \pi_E(\Sigma') \quad \text{and} \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma').$$

The resulting Protocol State is the state required by the frozen Synchronization contract, including return to Operational when that transition completes successfully.

Proof Sketch

The Synchronization contract requires accepted/rejected evidence sets and correlation status. Definition 12.3 preserves prior information.

Theorem 17.3 (Recovery Progress)

For every $\Sigma \in \mathcal{S}_R$ and $w \in \mathcal{W}$,

$$\text{recoverable}(\Sigma, w) = \text{true}$$

and successful completion of the corresponding Recovery action imply the existence of $\Sigma' = \delta(\Sigma, a_{\text{recovery}})$ such that

$$\pi_E(\Sigma) \subsetneq \pi_E(\Sigma'), \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma'),$$

with $(\pi_q(\Sigma), \pi_q(\Sigma')) \in \text{leadsto}$.

Proof

Immediate from Definitions 13.2 and 13.3 and the frozen transition relation.

Theorem 17.4 (Evaluation Progress)

If a Trust Evaluation request is admissible and sufficient processing completes, then there exists exactly one $v \in \mathcal{V}$ such that

$$\text{outcome}(v) \in \mathcal{Y},$$

and v references the evaluated context and continuity inputs required by the public result contract.

Proof Sketch

The Trust Evaluation Protocol Operation requires an evaluation result, evaluated context, and referenced continuity inputs. The outcome function is single-valued.

Boundary 17.1 (No Unconditional Temporal Liveness Claim)

The model does not prove that an operation will eventually be scheduled, that a network message will eventually arrive, that a Logical Validator will eventually respond, or that missing evidence will eventually become available. Such claims require fairness, availability, timing, and environmental assumptions not frozen by the protocol.

Chapter Summary

Progress is conditional on admissibility and completion; the document makes no unsupported eventuality claim.

18. Consistency and Correctness Properties

Chapter Principle

This chapter states the internal correctness achieved by the public mathematical model.

Definition 18.1 (Internal Model Consistency)

The model is internally consistent when its definitions are jointly satisfiable, every reachable state is well formed, every transition respects the frozen Protocol State relation and invariants, and no proven property contradicts another.

Theorem 18.1 (Reachable-State Consistency)

For every $\Sigma \in \mathcal{S}_R$,

$$\text{wellFormed}(\Sigma) = \text{true} \quad \text{and} \quad \bigwedge_{k=1}^8 I_k(\Sigma) = \text{true},$$

where I_k denotes Invariant 15.k.

Proof

By induction from $\mathcal{S}_0$ using Theorem 15.1.

Theorem 18.2 (Synchronization Consistency)

For every completed synchronization transition $\Sigma' = \delta(\Sigma, a_{\text{sync}})$ with $\Sigma \in \mathcal{S}_R$,

$$\text{wellFormed}(\Sigma') = \text{true} \quad \text{and} \quad \pi_E(\Sigma) \subseteq \pi_E(\Sigma') \quad \text{and} \quad \pi_P(\Sigma) \subseteq \pi_P(\Sigma').$$

The result exposes the disposition of transferred material without introducing a global total order.

Proof Sketch

Follows from Definitions 12.2–12.4, Proposition 12.1, Lemmas 15.1–15.2, and Theorem 16.5.

Theorem 18.3 (Recovery Consistency)

For every successful recovery transition $\Sigma' = \delta(\Sigma, a_{\text{recovery}})$ with $\Sigma \in \mathcal{S}_R$,

$$\text{wellFormed}(\Sigma') = \text{true} \quad \text{and} \quad \pi_E(\Sigma) \subseteq \pi_E(\Sigma') \quad \text{and} \quad \pi_G(\Sigma) \preceq \pi_G(\Sigma').$$

Proof

Follows from Definition 13.3, Proposition 13.1, and Theorem 16.6.

Theorem 18.4 (Evaluation Consistency)

For semantically equivalent admissible Trust Evaluation inputs $x_1 \equiv x_2$ producing results $v_1, v_2 \in \mathcal{V}$,

$$x_1 \equiv x_2 \rightarrow v_1 \equiv v_2,$$

while $v_k \neq b$ for every $b \in \mathcal{B}$ and $k \in \{1, 2\}$.

Proof

Observable determinism supplies equivalent results; application separation preserves the output boundary.

Meta-Theorem 18.1 (Canonical Correctness Theorem)

Under Axiom 15.1 and the proof boundaries of Chapter 19, $\mathcal{M}_{\text{QVTM}}$ is internally consistent with the frozen observable protocol semantics represented by this specification.

Proof Sketch

The static model is well formed by Parts I and II. The transition model restricts execution to admissible Protocol Operations and the exact Protocol State relation. Theorem 15.1 preserves all frozen protocol invariants. Theorems 16.1–16.7 exclude the prohibited finite behaviors represented by the model. Theorems 17.1–17.4 establish the specified conditional operation progress. Theorems 18.1–18.4 establish consistency of reachable states, synchronization, recovery, and evaluation. No result claims implementation, cryptographic, or adversarial security.

Corollary 18.1 (Observable Execution Correctness)

For every execution $\xi \in \mathcal{X}$ and every state Σ occurring in ξ ,

$$\text{wellFormed}(\Sigma) = \text{true} \quad \text{and} \quad \bigwedge_{k=1}^8 I_k(\Sigma) = \text{true}.$$

Chapter Summary

Correctness is explicitly internal and conditional; it is not a security or implementation proof.

19. Proof Boundaries

Chapter Principle

This chapter defines exactly what the preceding results prove and what they do not prove.

19.1 Included Proof Scope

The proof scope contains:

- the mathematical spaces and relations defined in this document;
- the frozen Protocol State relation;
- observable Protocol Operations and their public effects;
- immutable evidence and CEP preservation;
- Continuity Graph extension at the public boundary;
- Trust Evaluation outcome and application-separation semantics;
- synchronization and recovery observable effects.

19.2 Excluded Proof Scope

The results do not prove:

- correctness or security of Graph Puzzle internals;
- correctness of proprietary scoring, heuristics, or path selection;
- cryptographic confidentiality, integrity, authenticity, unforgeability, or post-quantum resistance;
- endpoint, operating-system, hardware, or physical security;
- application authorization correctness;
- implementation conformance or absence of software defects;
- availability, fairness, scheduling, delivery, or performance;
- adversarial security under any threat model;
- compliance with a certification or regulation.

19.3 Assumption Boundary

The proofs assume that modeled objects presented as valid satisfy the Protocol Specification's validation and integrity requirements. Cryptographic verification is treated as an external predicate supplied by the selected Deployment Profile and cryptographic services. Future Cryptographic Analysis may refine that predicate without changing the core continuity semantics.

19.4 Evolution Boundary

A future QVTM major version may change the model. A QVTM v1.x extension may add objects or constrained profile semantics only when existing definitions, invariant meanings, and frozen core behavior remain unchanged.

Chapter Summary

The formal results establish internal semantic consistency and no broader security claim.

Appendix A — Mathematical Symbols Reference

Symbol	Meaning	Declared in
$\mathcal{M}_{\{\mathrm{QVTM}\}}$	QVTM Formal Model	Definition 3.1
$\mathcal{I}$	Identity space	Definition 4.1
$\mathcal{K}$	Protocol/evaluation context space	Definition 4.2
$\mathcal{T}$	Trust State space	Definition 5.1
$\mathcal{L}$	Continuity relationship set	Definition 5.2
$\mathcal{E}$	Continuity Evidence space	Definition 6.1
$\mathcal{P}$	CEP space	Definition 7.1
$\mathcal{G}$	Continuity Graph space	Definition 8.1
$\mathcal{Y}$	Evaluation outcome space	Definition 8.3
$\mathcal{Q}$	Evaluation request space	Definition 8.4
$\mathcal{V}$	Trust Evaluation result space	Definition 8.5
$\mathcal{B}$	Application decision space	Definition 8.7
$\mathcal{Q}_{\{P\}}$	Protocol State space	Definition 9.1
$\mathcal{S}$	Observable protocol state space	Definition 9.2
$\mathcal{S}_0$	Valid initial state set	Definition 9.5
$\mathcal{O}_{\{P\}}$	Protocol Operation space	Definition 10.1
$\mathcal{A}$	Observable action space	Definition 10.2
δ	Partial transition function	Definition 10.4
$\leadsto$	Protocol State transition relation	Definition 11.1

Symbol	Meaning	Declared in
$\mathcal{X}$	Execution set	Definition 11.3
$\mathcal{S}_R$	Reachable state set	Definition 11.4
$\mathcal{W}$	Recovery request space	Definition 13.1
$\operatorname{supports}$	Evidence support relation	Definition 6.3
ev	CEP evidence function	Definition 7.2
$\operatorname{wellFormed}$	State well-formedness predicate	Definition 9.3
adm	Action admissibility predicate	Definition 10.3
$\operatorname{outcome}$	Trust Evaluation outcome function	Definition 8.5
id_E	Stable evidence identifier function	Definition 6.4
$\mathcal{D}_E$	Evidence-identifier space	Definition 6.4
id_P	Stable CEP identifier function	Definition 7.3
$\preceq$	Continuity Graph extension relation	Definition 8.2
$\equiv$	Semantic equivalence	Definition 2.1

Appendix B — Formal Definitions Reference

Identifier	Title
Definition 2.1	Semantic Equivalence
Definition 3.1	QVTM Formal Model
Definition 3.2	Observable Object
Definition 3.3	Model Interpretation
Definition 4.1	Identity Space
Definition 4.2	Identity Context
Definition 5.1	Trust State Space

Identifier	Title
Definition 5.2	Continuity Relationship
Definition 5.3	Continuity Path
Definition 5.4	Trust Continuity
Definition 5.5	Trust Discontinuity
Definition 6.1	Continuity Evidence Space
Definition 6.2	Evidence Context
Definition 6.3	Evidence Support Relation
Definition 6.4	Evidence Identity
Definition 6.5	Evidence Immutability
Definition 7.1	CEP Space
Definition 7.2	CEP Evidence Function
Definition 7.3	CEP Identity
Definition 7.4	CEP Context
Definition 7.5	CEP Immutability
Definition 7.6	Evidence Encapsulation Relation
Definition 8.1	Continuity Graph
Definition 8.2	Graph Extension
Definition 8.3	Evaluation Outcome Space
Definition 8.4	Evaluation Context
Definition 8.5	Trust Evaluation Result
Definition 8.6	Evaluation Boundary
Definition 8.7	Application Separation
Definition 9.1	Protocol State Space
Definition 9.2	Observable Protocol State

Identifier	Title
Definition 9.3	State Well-Formedness
Definition 9.4	State Projections
Definition 9.5	Initial State Set
Definition 10.1	Protocol Operation Space
Definition 10.2	Observable Action
Definition 10.3	Admissibility Predicate
Definition 10.4	Transition Function
Definition 10.5	Rejected Action Semantics
Definition 10.6	Observable Determinism
Definition 11.1	Protocol State Transition Relation
Definition 11.2	State-Transition Compatibility
Definition 11.3	Execution
Definition 11.4	Reachable State
Definition 11.5	Archived Terminality
Definition 12.1	Synchronization Input
Definition 12.2	Synchronization Result
Definition 12.3	Synchronization Extension
Definition 12.4	Synchronization Graph Effect
Definition 13.1	Recovery Request Space
Definition 13.2	Recovery Validation Predicate
Definition 13.3	Successful Recovery Transition
Definition 13.4	Failed Recovery
Definition 14.1	State Predicate
Definition 14.2	Transition Predicate

Identifier	Title
Definition 14.3	Execution Property
Definition 14.4	Invariant
Definition 14.5	Safety Property
Definition 14.6	Progress Property
Definition 18.1	Internal Model Consistency

Appendix C — Formal Statement Reference

C.1 Axiom

Identifier	Title
Axiom 15.1	Protocol Conformance Assumption

C.2 Invariants

Identifier	Title
Invariant 15.1	Evidence Immutability
Invariant 15.2	Append-Only Evidence
Invariant 15.3	CEP Preservation
Invariant 15.4	Observable Determinism
Invariant 15.5	Application Separation
Invariant 15.6	Failure Containment
Invariant 15.7	Cryptographic Independence
Invariant 15.8	State Well-Formedness

C.3 Lemmas

Identifier	Title
Lemma 15.1	Encapsulation Preservation
Lemma 15.2	Graph Evidence Availability

C.4 Propositions

Identifier	Title
Proposition 12.1	Synchronization Order Independence
Proposition 13.1	Recovery by Extension

C.5 Theorems and Meta-Theorem

Identifier	Title
Theorem 15.1	Invariant Preservation
Theorem 16.1	No Evidence Rewrite
Theorem 16.2	No CEP Rewrite or Silent Deletion
Theorem 16.3	No Invalid Protocol State Transition
Theorem 16.4	No Partial Commit on Rejection
Theorem 16.5	Synchronization Non-Destruction
Theorem 16.6	Recovery History Preservation
Theorem 16.7	Authorization Boundary Safety
Theorem 17.1	Enrollment Progress
Theorem 17.2	Synchronization Progress
Theorem 17.3	Recovery Progress
Theorem 17.4	Evaluation Progress

Identifier	Title
Theorem 18.1	Reachable-State Consistency
Theorem 18.2	Synchronization Consistency
Theorem 18.3	Recovery Consistency
Theorem 18.4	Evaluation Consistency
Meta-Theorem 18.1	Canonical Correctness Theorem
Corollary 18.1	Observable Execution Correctness

Appendix D — Documentation Dependency and Authority Reference

This appendix records how the Formal Foundations derives from the frozen QVTM documentation. It is a traceability boundary, not a redefinition of document authority.

D.1 Authority Levels

Authority level	Document	Formal Foundations use
1	QVTM Protocol Specification v1.0	Normative observable behavior, Protocol States, operations, invariants, synchronization, recovery, data semantics
2	QVTM White Paper v1.0	Architectural semantics, Continuity Graph, Trust State relationships, evaluation interpretation
3	QVTM SDK Reference v1.0	Public integration and result-shape consistency; no protocol redefinition
4	QVTM Position Paper v1.0	Scientific motivation, Trust Continuity concept, continuity domains

D.2 Documentation Dependency Workflow

Workflow Figure 2 records the documentary derivation and authority relationships used by this specification. Arrow direction denotes derivation or formal support, not authority precedence; authority precedence remains defined by Section D.1.

Workflow Figure 2 — Documentation dependency and authority workflow

flowchart TD

PP[Position Paper v1.0]

WP[White Paper v1.0]

PS[Protocol Specification v1.0]

SDK[SDK Reference v1.0]

FF[Formal Foundations v1.0]

SA[Future Security Analysis]

CA[Future Cryptographic Analysis]

CERT[Future Certification and Compliance]

PP --> |scientific concept and motivation| WP

PP --> |conceptual source| FF

WP --> |architectural semantics| PS

WP --> |architectural derivation| FF

PS --> |highest behavioral authority| SDK

PS --> |normative formal source| FF

SDK --> |public-surface consistency| FF

FF --> |formal basis| SA

FF --> |formal basis| CA

FF --> |formal basis| CERT

Caption. The frozen first-generation documents supply scientific, architectural, normative, and integration semantics to the Formal Foundations; the Formal Foundations then provide the mathematical basis for later analytical and assurance documents.

D.3 Formal Traceability Matrix

Formal topic	Position Paper	White Paper	Protocol Specification	SDK Reference
Trust Continuity				referenced
Identity continuity			operation/context semantics	public abstraction
Trust State			normative term	result context
Continuity Evidence			normative object and invariants	immutable public object
CEP			normative container	public package abstraction

Formal topic	Position Paper	White Paper	Protocol Specification	SDK Reference
Continuity Graph and Path			normative terms/semantics	intentionally hidden internally
Trust Evaluation outcomes	conceptual		Protocol Operation/result	public result abstraction
Protocol States	—	conceptual	normative	status abstraction only
Protocol Operations	—	architectural	normative	public operation mapping
Synchronization			normative	public workflow
Recovery			normative	public workflow
Application separation			normative invariant	integration responsibility

D.4 Dependency Policy

A formal definition must identify a derivation path to at least one frozen document. When sources differ in abstraction, the Protocol Specification governs observable behavior. An inconsistency discovered in a frozen document is recorded for QVTM v2.0 and is not silently repaired by this specification.

Appendix E — Proof Dependency Reference

Statement	Direct dependencies
Proposition 12.1	Definitions 12.3–12.4; Section 12.1
Proposition 13.1	Definition 13.3
Lemma 15.1	Definitions 7.2, 9.3; Invariants 15.2–15.3
Lemma 15.2	Definitions 8.1, 9.3; Invariant 15.2

Statement	Direct dependencies
Theorem 15.1	Axiom 15.1; Invariant 15.1; Invariant 15.2; Invariant 15.3; Invariant 15.4; Invariant 15.5; Invariant 15.6; Invariant 15.7; Invariant 15.8; Definitions 9.3, 10.4–10.5
Theorem 16.1	Definition 6.4; Invariants 15.1–15.2
Theorem 16.2	Definition 7.3; Invariant 15.3
Theorem 16.3	Definitions 11.1–11.2
Theorem 16.4	Definitions 10.4–10.5; Invariant 15.6
Theorem 16.5	Definitions 12.3–12.4; Invariants 15.1–15.3
Theorem 16.6	Definition 13.3; Invariants 15.2–15.3
Theorem 16.7	Definition 8.7; Invariant 15.5
Theorem 17.1	Definitions 10.1–10.4; frozen Enrollment contract
Theorem 17.2	Definitions 12.1–12.4
Theorem 17.3	Definitions 13.2–13.3
Theorem 17.4	Definitions 8.3–8.6
Theorem 18.1	Theorem 15.1
Theorem 18.2	Proposition 12.1; Lemma 15.1; Lemma 15.2; Theorem 16.5
Theorem 18.3	Definition 13.3; Proposition 13.1; Theorem 16.6
Theorem 18.4	Definitions 8.6–8.7, 10.6; Invariant 15.5
Meta-Theorem 18.1	Theorem 15.1; Theorem 16.1; Theorem 16.2; Theorem 16.3; Theorem 16.4; Theorem 16.5; Theorem 16.6; Theorem 16.7; Theorem 17.1; Theorem 17.2; Theorem 17.3; Theorem 17.4; Theorem 18.1; Theorem 18.2; Theorem 18.3; Theorem 18.4
Corollary 18.1	Meta-Theorem 18.1

Appendix F — Future Formal Extensions Reference

This appendix records the intended derivation path from the frozen Formal Foundations to later second-generation QVTM documents. It introduces no new formal property and grants no authority

to future documents to redefine this specification.

Companion document	Formal structures extended	Permitted analytical contribution
QVTM Security Analysis	Invariants, safety properties, Protocol State transitions, synchronization, recovery, and proof boundaries	Threat model, adversarial assumptions, attack analysis, and security claims bounded by the formal model
QVTM Cryptographic Analysis	Evidence integrity, CEP preservation, cryptographic independence, synchronization, recovery, and historical verification	Primitive assumptions, cryptographic games, reductions, migration analysis, and post-quantum assurance
QVTM Certification Guide	Definitions, traceability, invariants, safety, progress, and internal correctness	Auditable control objectives, evidence requirements, assessment procedures, and certification mappings
QVTM Compliance Profiles	Model properties, Deployment Profile constraints, application separation, and proof boundaries	Jurisdictional or sector-specific requirements that constrain deployments without redefining core semantics
Language-specific SDK Guides	Public operation, result, and application-separation semantics	Idiomatic integration guidance consistent with the frozen SDK Reference and formal boundaries
Interoperability Guide	Semantic equivalence, Protocol Operations, synchronization dispositions, and observable determinism	Cross-implementation test profiles and compatibility procedures

Future analytical documents may strengthen assumptions or prove additional properties. They shall not weaken, reinterpret, or silently replace any frozen definition, invariant, transition, or proof boundary established here.

Freeze Declaration

QVTM Formal Foundations v1.0 is frozen. Its definitions, normative terminology, formal statement identifiers, invariant set, Protocol State semantics, and proof boundaries constitute the canonical mathematical foundation of QVTM v1.x.

Editorial corrections may repair typographical or cross-reference defects without changing mathematical meaning. Any semantic modification, new protocol behavior, weakened invariant, altered proof boundary, or redefinition of a frozen concept requires a future major-version process and shall not be introduced silently.

