Chapter 14

Economic and Trust Models for Field Governance

Responsibility, Contracts, and Institutional Trust

proFQuansistor

Abstract

This document introduces economic and trust models as first-class components of field governance within the Quansistor Field Computing architecture. Building upon execution semantics, governance mechanisms, isolation, orchestration, and auditability, it formalizes how responsibility, incentives, and trust are established and maintained in field-native computational systems.

Rather than treating trust as an external assumption or economic incentives as an afterthought, this whitepaper integrates them directly into governance structures. Trust is defined through verifiable execution, contract enforcement, and auditability. Economic models are expressed through governed admission, execution obligations, and accountable domain lifecycles rather than through opaque metering or heuristic pricing.

The chapter explores how execution domains become accountable entities, how contracts acquire economic meaning, and how institutions may rely on field-native computation without requiring intrusive oversight. By aligning economic incentives with governance guarantees, the document establishes a foundation for sustainable, multi-actor field computing ecosystems.

This whitepaper prepares the transition from system-level orchestration to institutional deployment, legal interoperability, and real-world adoption of field-native computation.

1 Trust as a Property of Governance, Not Assumption

Trust in computational systems is traditionally established through external mechanisms: trusted operators, reputational systems, legal enforcement, or opaque certification processes. In field-native computation governed by QVM, trust is redefined as an intrinsic property of governance structures rather than as an external assumption. This chapter formalizes trust as a consequence of verifiable execution, contract enforcement, and auditability.

In the QFC architecture, trust does not arise from belief in correct behavior, but from the impossibility of unlawful behavior within admitted execution domains. Governance mechanisms ensure that only admissible execution trajectories may occur and that all admissible trajectories are subject to invariant enforcement. Trust is therefore structural: it follows from what execution is allowed to do, not from who operates the system.

A trusted execution domain is defined as a domain whose governance configuration provides

sufficient guarantees for a given purpose. These guarantees may include determinism bounds, auditability requirements, isolation properties, or contractual obligations. Trust is contextual rather than absolute; a domain may be trusted for one class of computation and not for another, depending on its declared governance obligations.

Crucially, trust is separable from execution. QVM does not observe execution to determine whether it should be trusted. Instead, trust is inferred from governance configuration and verified through auditability. If an execution domain satisfies its declared contracts and remains auditable, its results are trustworthy by construction. No runtime supervision or behavioral monitoring is required.

This formulation eliminates the need for trusted intermediaries. Institutions, users, or systems interacting with field-native computation do not need to trust operators, hardware vendors, or orchestration logic. They need only trust the governance framework and its enforcement mechanisms, which are explicit, verifiable, and auditable.

Trust is also compositional. Governed field systems composed of multiple execution domains inherit trust properties from their constituents under explicit composition rules. Where trust boundaries exist, they are encoded as isolation boundaries or interface contracts rather than as informal assumptions. This makes trust relationships explicit, analyzable, and enforceable.

By redefining trust as a property of governance rather than assumption, this chapter establishes a foundation for economic interaction, responsibility attribution, and institutional reliance on field-native computation. Trust becomes a measurable and verifiable characteristic of execution environments, enabling economic and legal systems to interact with computation without sacrificing rigor or accountability. s

2 Execution Domains as Accountable Economic Actors

In field-native systems governed by QVM, economic accountability is not attached to users, processes, or hardware components, but to execution domains themselves. This chapter formalizes execution domains as accountable economic actors whose obligations, responsibilities, and liabilities are defined entirely through governance configuration.

An execution domain becomes an economic actor when it is admitted with explicit obligations that have economic meaning. Such obligations may include determinism guarantees, auditability requirements, isolation commitments, service availability constraints, or bounded nondeterminism for exploratory computation. These obligations are not behavioral promises; they are governance-enforced invariants.

Accountability arises from the alignment between declared obligations and verifiable execution. Because execution domains cannot violate their active contracts, compliance is not inferred from observed behavior but guaranteed by admissibility. An accountable domain is therefore one whose governance configuration makes non-compliance structurally impossible.

Economic interaction with an execution domain does not require trust in its operator or implementer. Counterparties rely on the domain's governance declaration and the auditability of its lifecycle. If a domain satisfies its declared invariants, it fulfills its economic role by construction. If it cannot satisfy them, it is not admissible as an economic actor in that role.

Responsibility is attached to domain lifecycle events rather than to execution steps. Admission, reconfiguration, suspension, and termination are governance acts that may carry economic consequences. These acts define when obligations begin, how they may change, and when they end. Economic responsibility is thus discrete, auditable, and attributable without inspecting execution internals.

Execution domains may participate in economic relationships individually or as part of composed

systems. In composed domains, accountability is governed by composition contracts that specify how obligations are distributed or aggregated. This enables complex economic arrangements such as federated computation, cooperative workflows, or delegated execution under shared governance.

Importantly, domains do not act strategically. They do not optimize profit, negotiate terms, or adapt behavior. Economic agency is purely structural. Incentives and sanctions operate at the governance layer by admitting, configuring, or excluding domains rather than by influencing execution behavior.

By defining execution domains as accountable economic actors, this chapter establishes a rigorous foundation for economic interaction with field-native computation. Accountability becomes a property of governance and auditability rather than of behavior or intent. This formulation prepares the next chapter on contracts, obligations, and incentive alignment, where economic meaning is bound explicitly to governance structures.

3 Contracts, Obligations, and Incentive Alignment

In QVM-governed field systems, contracts are not symbolic agreements or external enforcement instruments. They are governance-level structures that encode obligations directly into the admissibility and lifecycle of execution domains. This chapter formalizes contracts as the primary mechanism through which economic obligations and incentives are aligned with field-native computation.

An execution contract specifies a set of obligations that an execution domain must satisfy throughout its lifecycle. These obligations may include determinism bounds, auditability guarantees, isolation commitments, availability constraints, or limits on admissible nondeterminism. Crucially, contracts do not prescribe behavior; they prescribe invariants. Compliance is ensured not by monitoring but by structural enforcement.

Obligations become economically meaningful when they are associated with consequences at the governance layer. Such consequences may include domain admission eligibility, priority in workflow orchestration, access to compositional interfaces, or eligibility for institutional reliance. Incentives arise from the selective admission and configuration of domains rather than from rewards tied to execution outcomes.

Incentive alignment in QFC does not rely on strategic agents or behavioral optimization. Domains do not seek to maximize utility. Instead, alignment is achieved by designing governance rules such that only domains that satisfy economically desirable obligations are admissible for economically valuable roles. Incentives are structural filters, not motivational signals.

Sanctions are expressed as governance actions rather than penalties imposed after the fact. A domain that cannot satisfy its contractual obligations is simply inadmissible for certain configurations, workflows, or institutional uses. Suspension, reconfiguration denial, or termination are governance outcomes, not punishments. This eliminates adversarial dynamics and reduces the need for dispute resolution mechanisms.

Contracts may be composed hierarchically. System-level contracts may impose obligations on collections of domains, while domain-level contracts govern individual execution contexts. Composition rules specify how obligations aggregate or constrain each other, enabling complex contractual arrangements such as federated execution, delegated responsibility, or shared audit guarantees.

Importantly, contracts are auditable objects. The existence, activation, modification, and satisfaction of contracts are governance events that can be verified independently of execution internals. This auditability allows economic counterparties and institutions to rely on contractual

guarantees without requiring access to proprietary execution details.

By defining contracts as governance-enforced obligations and incentives as admissibility conditions, this chapter establishes a non-adversarial economic model for field-native computation. Economic alignment emerges from structural compatibility between obligations and governance roles rather than from behavioral incentives. This framework prepares the next chapter on auditability, liability, and institutional verification.

4 Auditability, Liability, and Institutional Verification

For field-native computation to be adopted by institutions, auditability must support not only technical verification but also responsibility attribution and legal accountability. This chapter formalizes how auditability in QVM-governed systems enables liability assessment and institutional verification without introducing intrusive oversight or execution-level surveillance.

Auditability in QFC is a structural property of governance, not a logging mechanism. Execution domains are auditable because their admissible behavior is constrained by invariant contracts and their lifecycle is governed by explicit, verifiable acts. Institutional verification therefore does not require access to execution internals, source code, or runtime traces. It relies on governance records and admissibility proofs.

Liability is attached to governance obligations rather than to execution events. When an execution domain is admitted under a declared contract set, it assumes responsibility for satisfying those obligations. If an obligation cannot be satisfied, the domain is inadmissible for that role by construction. Liability thus concerns incorrect admission, misconfiguration, or inappropriate reliance on a domain, not misbehavior during execution.

This separation is critical for institutional use. Courts, regulators, and auditors do not need to reconstruct execution step-by-step. Instead, they verify whether the governance framework was correctly applied: whether the domain was admitted under appropriate contracts, whether lifecycle transitions were lawful, and whether any reliance on execution results respected declared guarantees.

Institutional verification operates at the governance boundary. Verification procedures may include validation of contract definitions, inspection of domain configurations, confirmation of execution mode assignments, and review of lifecycle events. These procedures are deterministic, finite, and independent of execution complexity or scale.

Importantly, auditability enables ex ante assurance rather than ex post investigation. Institutions can rely on execution domains because governance guarantees make unlawful execution impossible, not because wrongdoing can be detected afterward. This shifts the role of audit from policing to assurance.

Liability allocation follows naturally from this model. Responsibility lies with the entities that define governance policies, admit domains, and rely on declared guarantees. Execution domains themselves are accountable entities, but they do not bear intent or fault. Human or institutional actors bear responsibility for governance decisions, not for execution dynamics.

By grounding auditability and liability in governance rather than behavior, this chapter establishes a bridge between field-native computation and institutional trust. Verification becomes a matter of checking governance correctness, enabling legal, regulatory, and organizational systems to interact with computation rigorously without requiring invasive control or subjective judgment.

5 Auditability, Liability, and Institutional Verification

For field-native computation to be adopted by institutions, auditability must support not only technical verification but also responsibility attribution and legal accountability. This chapter formalizes how auditability in QVM-governed systems enables liability assessment and institutional verification without introducing intrusive oversight or execution-level surveillance.

Auditability in QFC is a structural property of governance, not a logging mechanism. Execution domains are auditable because their admissible behavior is constrained by invariant contracts and their lifecycle is governed by explicit, verifiable acts. Institutional verification therefore does not require access to execution internals, source code, or runtime traces. It relies on governance records and admissibility proofs.

Liability is attached to governance obligations rather than to execution events. When an execution domain is admitted under a declared contract set, it assumes responsibility for satisfying those obligations. If an obligation cannot be satisfied, the domain is inadmissible for that role by construction. Liability thus concerns incorrect admission, misconfiguration, or inappropriate reliance on a domain, not misbehavior during execution.

This separation is critical for institutional use. Courts, regulators, and auditors do not need to reconstruct execution step-by-step. Instead, they verify whether the governance framework was correctly applied: whether the domain was admitted under appropriate contracts, whether lifecycle transitions were lawful, and whether any reliance on execution results respected declared guarantees.

Institutional verification operates at the governance boundary. Verification procedures may include validation of contract definitions, inspection of domain configurations, confirmation of execution mode assignments, and review of lifecycle events. These procedures are deterministic, finite, and independent of execution complexity or scale.

Importantly, auditability enables ex ante assurance rather than ex post investigation. Institutions can rely on execution domains because governance guarantees make unlawful execution impossible, not because wrongdoing can be detected afterward. This shifts the role of audit from policing to assurance.

Liability allocation follows naturally from this model. Responsibility lies with the entities that define governance policies, admit domains, and rely on declared guarantees. Execution domains themselves are accountable entities, but they do not bear intent or fault. Human or institutional actors bear responsibility for governance decisions, not for execution dynamics.

By grounding auditability and liability in governance rather than behavior, this chapter establishes a bridge between field-native computation and institutional trust. Verification becomes a matter of checking governance correctness, enabling legal, regulatory, and organizational systems to interact with computation rigorously without requiring invasive control or subjective judgment.