QCOS

Operating System for Governed Quantum Infrastructure

Policy, Auditability, Cost Attribution, and Vendor-Neutral Execution at Scale

SoftQuantus innovative OÜ

Company Registry: 17048927
https://softquantus.com

Version 1.0 January 2026

Contents

Executive Summary

The Governance Gap in Production Quantum

As quantum computing transitions from proof-of-concept to production deployment, organizations face a critical challenge: **governance failures block scaling more than algorithmic limitations**. Enterprise and sovereign users require that every quantum execution be treated as an **auditable, cost-attributable transaction**analogous to the transformation that occurred in cloud computing with the emergence of governed infrastructure.

Core Thesis

To achieve scale and institutional trust, quantum infrastructure must support:

- **Policy-as-code**: Automated guardrails for access, usage, and resource allocation
- **Verifiable audit trails**: Complete traceability from submission to results
- **FinOps integration**: Cost attribution, showback, and chargeback mechanisms
- **Vendor neutrality**: Consistent governance across multi-vendor environments

The QCOS Proposition

QCOS (Quantum Computing Operating System) is the control plane that transforms quantum job execution into governed transactions. QCOS abstracts vendor specifics while enforcing policies, generating evidence, and enabling operational accountability.

Key Messages

1. **Governance > Algorithms**: Production quantum fails on operational controls, not quantum circuits
2. **Transaction Model**: Quantum execution must become a governed transaction (policy + audit + cost)
3. **Vendor Neutrality + Compliance**: QCOS enables multi-vendor strategies without sacrificing traceability
4. **Operational Value**: Measurable improvements in time, evidence quality, cost attribution, and vendor management

Expected Outcomes

Organizations implementing QCOS-governed quantum infrastructure can expect:

- **Operational efficiency**: Reduced time per job through automation and standardized policies
- **Complete traceability**: Full audit trail from job submission to artifact generation
- **FinOps alignment**: Per-execution cost models compatible with enterprise financial practices
- **Vendor flexibility**: Execute workloads across multiple providers with unified observability

1 The Production Problem: Governance Over Algorithms

1.1 Why Governance Matters Now

Quantum computing has reached an inflection point. The bottleneck to production deployment is no longer purely technicalit is **operational and institutional**.

1.1.1 Observable Symptoms in Enterprise/HPC/Defense Environments

Organizations attempting to operationalize quantum computing consistently encounter the same challenges:

Critical Questions Without Answers

- **Authorization**: "Who can run what, where, and with which budget?"
- **Reproducibility**: "How do we reproduce and explain results 618 months later?"
- **Vendor Management**: "How do we compare performance and detect drift across providers?"
- **Cost Attribution**: "How do we allocate costs per project/team/cost center?"

These are not quantum-specific problemsthey mirror challenges solved in cloud computing through **governed infrastructure** patterns. However, quantum computing lacks the equivalent control plane.

1.2 First Principles: What Production Execution Requires

A quantum execution in a production environment is a **critical event** that must contain, at minimum:

Element	Requirement
What	Circuit/algorithm specification, input parameters
When	Reliable timestamp (not system clock)
Where	Execution backend, region, compliance zone
Who	Authenticated identity, project/team attribution
Why	Business justification, classification, purpose
Result	Output data, execution metadata, resource consumption
Evidence	Cryptographic proof of integrity and provenance

Table 1: Minimal audit record content (aligned with NIST 800-53 AU-3)

This is consistent with classical audit controls such as NIST 800-53 AU (Audit and Accountability) family.

1.3 The Scaling Barrier

Without auditable trails and governance mechanisms, quantum computing cannot scale to support:

- **Procurement processes**: Vendor selection requires objective performance metrics
- **Compliance regimes**: Regulatory frameworks demand evidence and accountability

- **Service-level agreements**: SLAs require measurable, verifiable performance
- **Financial accountability**: Cost allocation requires precise metering and attribution

Conclusion: The quantum industry needs a governance layer as urgently as it needs better qubits.

2 Quantum as Infrastructure: Transactions, Policies, Evidence, Costs

2.1 Defining "Governed Quantum Infrastructure"

We propose a new operational category for the industry:

Governed Quantum Infrastructure (Definition)

Governed Quantum Infrastructure is quantum computing capacity operated as critical infrastructure with:

- **Access and usage policies**: Who can do what, enforced programmatically
- **Verifiable evidence**: Cryptographic proof of execution integrity
- **Cost attribution and accountability**: Metering, showback, and chargeback
- **Vendor neutrality**: Consistent control plane across providers

This mirrors the transformation that occurred in cloud computing, where "infrastructure as code" and governance tooling (IAM, cost management, audit logs) became as critical as the compute resources themselves.

2.2 The Transaction Model for Quantum Execution

In governed infrastructure, a quantum job is not merely "code that runs"it is a **governed transaction** with:

1. **Pre-execution validation**: Policy checks, budget verification, authorization
2. **Execution tracking**: Real-time telemetry, resource consumption monitoring
3. **Post-execution evidence**: Immutable audit record with cryptographic signatures
4. **Cost settlement**: Metered resource usage attributed to responsible entity

2.3 What QCOS Is NOT (Avoiding Confusion)

To clarify scope and positioning:

QCOS is NOT	Explanation
Algorithm framework	Does not replace Qiskit, Cirq, or quantum SDKs
Vendor SDK	Does not compete with AWS Braket or Azure Quantum APIs
Dashboard/UI only	Includes UI but primarily a control plane
QPU runtime	Does not execute circuits; governs execution

Table 2: QCOS scope clarification

Key distinction: QCOS does not replace vendor runtimesit governs and orchestrates them.

3 QCOS Architecture

3.1 Layered Architecture Overview

QCOS implements a layered architecture separating concerns between application, control plane, hardware abstraction, and data plane.

3.1.1 Architecture Diagram 1: QCOS Layers

QCOS Layered Architecture

Layer 1: Application Layer

- SDK / CLI / API
- Job submission with policies, budgets, evidence requests

Layer 2: QCOS Control Plane (Operating System)

- **Identity & Access Management**: Federated auth (OIDC/SAML), RBAC/ABAC
- **Policy Engine**: Policy-as-code enforcement
- **Job Orchestrator**: Queue management, routing, retry logic
- **Evidence & Audit Ledger**: Append-only audit trail with signatures
- **Metering & Cost Attribution**: Showback/chargeback calculation
- **Observability**: Traces, logs, metrics (OpenTelemetry-based)

Layer 3: Hardware Abstraction Layer (HAL) / Drivers

- Cloud providers: AWS Braket, Azure Quantum, IBM Quantum, IonQ, Rigetti
- On-premise QPU gateways
- Simulators and HPC backends

Layer 4: Data Plane

- Actual quantum execution
- Result retrieval and artifact storage

3.2 Detailed Component Specifications

3.2.1 1. QCOS Job Specification (Canonical Format)

The QCOS Job Spec is a vendor-neutral, declarative format for quantum workloads:

Listing 1: QCOS Job Spec Example

```
{
  "job_id": "qcos-job-2026-01-09-a7f3",
  "circuit": {
    "format": "QASM3",
    "source": "base64-encoded-circuit",
    "compiler_version": "qiskit-1.0.0"
  },
```

```
  "execution": {
    "shots": 1000,
    "backend_constraints": {
      "min_qubits": 20,
      "max_depth": 100,
      "allowed_providers": ["aws-braket", "ibm-quantum"]
    }
  },
  "governance": {
    "project": "material-simulation-q1-2026",
    "cost_center": "CC-RD-42",
    "classification": "internal-restricted",
    "purpose": "production-benchmark"
  },
  "evidence_requirements": {
    "level": "full",
    "retention_days": 2555
  }
}
```

3.2.2 2. Policy Engine (Policy-as-Code)

QCOS implements policy enforcement using declarative policy languages (e.g., OPA/Rego):
Example policies:

- Budget enforcement: "Project X cannot exceed $5,000/month on quantum resources"
- Access control: "Only 'quantum-engineer' role can submit production jobs"
- Compliance: "Classified data must only execute in sovereign on-premise backends"
- Resource quotas: "Team Y limited to 50,000 shots/day"

3.2.3 3. Evidence Bundle

Every executed job generates an immutable evidence bundle:

Listing 2: Evidence Bundle Structure

```
{
  "bundle_id": "evidence-2026-01-09-xyz",
  "job_id": "qcos-job-2026-01-09-a7f3",
  "input_hash": "sha256:abcd1234...",
  "output_hash": "sha256:5678efgh...",
  "execution_metadata": {
    "backend": "aws-braket-sv1",
    "start_time": "2026-01-09T10:30:00Z",
    "end_time": "2026-01-09T10:30:45Z",
    "shots_executed": 1000,
    "queue_time_ms": 2300
  },
  "governance_record": {
    "actor": "user:alice@company.com",
    "project": "material-simulation-q1-2026",
    "policies_applied": ["budget-check", "classification-enforcement"],
    "policy_decisions": ["allow"]
  },
  "signatures": {
    "qcos_signature": "ML-DSA-sig...",
```

```
    "timestamp_authority": "RFC3161-timestamp..."
  }
}
```

3.2.4 4. Audit Ledger

The Audit Ledger is an append-only, tamper-evident log of all governance actions:

- Job submissions and policy decisions
- Administrative actions (policy changes, user management)
- Evidence bundle creation
- Cost allocations and budget modifications

Storage options:

- Cryptographically signed append-only database
- WORM (Write-Once-Read-Many) storage for regulatory compliance
- Optional blockchain anchoring for highest assurance

3.2.5 5. Metering and Cost Attribution

QCOS implements granular metering based on:

Metric	Description
Shots executed	Primary consumption unit
QPU time	Wall-clock time on quantum processor
Queue time	Waiting time in provider queues
Circuit depth	Complexity metric
Data egress	Result data transfer
Simulator resources	CPU/memory for classical simulation

Table 3: QCOS metering dimensions

FinOps integration:

- Real-time budget tracking and alerts
- Showback: Cost visibility per team/project
- Chargeback: Automated cost allocation to billing entities
- Forecasting: Predictive cost modeling based on usage patterns

3.2.6 6. Observability (OpenTelemetry-Based)

QCOS exports telemetry using OpenTelemetry standards:

- **Traces**: End-to-end request flow (submission execution results)
- **Metrics**: Queue depths, success rates, latency distributions
- **Logs**: Structured event logs for debugging and audit

Integration with existing observability stacks (Splunk, Elastic, Grafana, Datadog) via standard OTLP exporters.

4 Execution Workflows

4.1 Workflow A: Standard Governed Execution

4.1.1 Step-by-Step Flow

1. **Authentication**: User/service authenticates via federated identity (OIDC/SAML)
2. **Job Submission**: Submit job with QCOS Job Spec including governance tags (project, cost center, classification)
3. **Policy Validation**: Policy Engine evaluates:
   - Does user have permission?
   - Is budget available?
   - Are backend constraints satisfiable?
   - Are quotas within limits?
4. **Routing Decision**: Job Orchestrator selects optimal backend based on:
   - Backend constraints
   - Current queue depths
   - Cost optimization
   - Performance history
5. **Execution**: Job submitted to selected backend via HAL driver
6. **Result Collection**: Orchestrator retrieves results and execution metadata
7. **Evidence Generation**: Evidence Bundle created with:
   - Input/output hashes
   - Execution metadata
   - Governance decisions
   - Cryptographic signatures
8. **Audit Recording**: Evidence Bundle registered in Audit Ledger
9. **Metering**: Cost calculation and attribution to cost center
10. **Observability Export**: Telemetry exported to observability platforms

4.1.2 Workflow Diagram 2: Governed Execution Flow

Governed Execution Sequence

```
User Auth Submit Job Policy Check
Orchestrator Backend Selection QPU Execution
Result Retrieval Evidence Bundle Audit Ledger
Metering Cost Attribution Observability Export
```

4.2 Workflow B: Reproducibility Replay

A key differentiator of QCOS is the ability to **reproduce past executions** using historical evidence bundles.

4.2.1 Replay Process

1. **Retrieve Evidence Bundle**: Access historical bundle from Audit Ledger
2. **Verify Integrity**: Validate cryptographic signatures and timestamps
3. **Extract Configuration**:
    - Original circuit specification
    - Compiler/transpiler version
    - Random seeds and configuration parameters
    - Backend constraints
4. **Re-execute**: Submit identical job with same parameters
5. **Compare Outputs**: Analyze differences:
    - Statistical comparison of measurement distributions
    - Drift detection (hardware evolution over time)
    - Performance regression analysis
6. **Generate Reproducibility Report**: Document results and deviations

Use cases for reproducibility:

- **Scientific validation**: Verify published results
- **Regulatory compliance**: Demonstrate consistency for audits
- **Vendor comparison**: Benchmark across providers with identical workloads
- **Performance regression**: Detect hardware degradation over time

5 Deployment Models

5.1 Deployment Profiles

QCOS supports four primary deployment models, each optimized for different organizational requirements:

Model	Tenant Isolation	Connectivity	Primary Use Case
SaaS Multi-Tenant	Logical	Public internet	Enterprise R&D, non-sensitive
Dedicated Tenant	Physical	Public/private	High-compliance, data sensitivity
On-Premise	Full control	Private network	HPC centers, sovereign cloud
Air-Gapped	Full isolation	No external	Defense, critical infrastructure

Table 4: QCOS deployment models

5.2 Model 1: SaaS Multi-Tenant

Architecture:

- Shared QCOS control plane
- Logical tenant isolation (namespace-based)
- Public cloud infrastructure

Suitable for:

- Early-stage quantum exploration
- Academic and educational use
- Non-sensitive enterprise R&D

5.3 Model 2: Dedicated Tenant

Architecture:

- Dedicated QCOS instance per organization
- Network isolation (VPC/VNET)
- Dedicated encryption keys (customer-managed KMS)

Suitable for:

- Regulated industries (finance, healthcare)
- Organizations with data residency requirements
- High-value IP protection

5.4 Model 3: On-Premise

Architecture:

- QCOS deployed in customer datacenter
- Integration with on-premise QPUs or hybrid cloud
- Customer-controlled infrastructure

Suitable for:

- HPC centers with existing quantum hardware
- Sovereign cloud requirements
- Organizations with strict data governance

5.5 Model 4: Air-Gapped / Sovereign

Architecture:

- Completely isolated network
- Local timestamping and cryptographic services
- Offline update mechanisms
- No external dependencies

Suitable for:

- Defense and national security
- Critical infrastructure
- Maximum sovereignty requirements

5.6 Sovereignty Requirements (Detailed)

For sovereign deployments, QCOS implements:

1. **Cryptographic Independence**:
   - Local KMS/HSM for key management
   - Customer-controlled certificate authority
   - No reliance on external PKI
2. **Supply Chain Control**:
   - Cryptographically signed software updates
   - Offline update packages
   - Reproducible builds
3. **Data Residency**:
   - All data (logs, evidence, results) stored locally
   - Configurable retention policies
   - No data exfiltration channels

4. **Integration with Legacy Systems**:
    - LDAP/Active Directory authentication
    - Integration with existing SIEM
    - Compatibility with institutional certificate infrastructure

6 Enterprise Integrations

6.1 Identity and Access Management (IAM)

6.1.1 Federated Authentication

QCOS supports enterprise identity providers via:

- **OIDC (OpenID Connect)**: Modern OAuth2-based authentication
- **SAML 2.0**: Enterprise SSO integration
- **LDAP/Active Directory**: Direct directory integration

6.1.2 Authorization Models

Role-Based Access Control (RBAC):

- Predefined roles: quantum-admin, quantum-engineer, quantum-user, auditor
- Role-to-permission mappings
- Project-scoped role assignments

Attribute-Based Access Control (ABAC):

- Dynamic policies based on user attributes, resource tags, environmental context
- Example: "Users from research department can only submit benchmark jobs"

6.1.3 Delegation and Service Accounts

- Temporary credential delegation for automated workflows
- Service accounts for CI/CD integration
- Time-limited access tokens

6.2 Observability Integration

6.2.1 OpenTelemetry Export

QCOS implements OpenTelemetry standards for vendor-neutral telemetry:

Listing 3: OpenTelemetry Configuration Example

```
exporters:
  otlp:
    endpoint: "https://observability.company.com:4317"
    headers:
      authorization: "Bearer ${OTEL_TOKEN}"

processors:
  batch:
    timeout: 10s
    send_batch_size: 1024

service:
  pipelines:
    traces:
      receivers: [otlp]
```