Table of Contents

1. Configuration Reference

This appendix explains key configuration properties used in Quantum applications, organized by functional area.

1.1. Configuration Basics

File Locations

- `src/main/resources/application.properties` - main configuration
- `src/test/resources/application.properties` - test overrides
- `.env` file - local development secrets (not committed)

Profiles and Environment Variables

```
# Profile-specific properties
%dev.auth.jwt.duration=7200
%test.quarkus.mongodb.database=test-db
%prod.quarkus.log.level=WARN

# Environment variable substitution
quarkus.mongodb.connection-
string=${MONGODB_CONNECTION_STRING:mongodb://localhost:27017}
auth.jwt.secret=${JWT_SECRET:change-me-in-production}
```

Precedence

System properties > Environment variables > application.properties

1.2. Database Configuration

MongoDB Connection

```
# Basic connection
quarkus.mongodb.connection-string=mongodb://localhost:27017
quarkus.mongodb.database=my-app

# With authentication
quarkus.mongodb.connection-
string=mongodb://user:pass@host:27017/database?authSource=admin

# Disable dev services (use real MongoDB)
quarkus.mongodb.devservices.enabled=false
```

Morphia Configuration

```
# Database and package scanning
quarkus.morphia.database=my-app-db
quarkus.morphia.packages=com.example.model,com.example.entities

# Schema management
quarkus.morphia.create-caps=true
quarkus.morphia.create-indexes=true
quarkus.morphia.create-validators=true
```

1.3. Authentication Configuration

JWT Provider

```
# Enable JWT authentication
quarkus.smallrye-jwt.enabled=true
auth.provider=custom

# JWT validation
mp.jwt.verify.publickey.location=publicKey.pem
mp.jwt.verify.issuer=my-app
mp.jwt.verify.audiences=my-app-users

# Custom JWT settings
auth.jwt.secret=${JWT_SECRET:your-secret-here}
auth.jwt.expiration=60
auth.jwt.refresh-expiration=1440
```

OIDC Provider

```
# Enable OIDC
quarkus.oidc.enabled=true
auth.provider=oidc

# OIDC configuration
quarkus.oidc.auth-server-url=https://your-provider.com/auth/realms/your-realm
quarkus.oidc.client-id=your-client-id
quarkus.oidc.credentials.secret=your-client-secret
quarkus.oidc.roles.role-claim-path=groups
```

1.4. Quantum Framework Configuration

Database Migration

```
# Migration settings
```

```
quantum.database.version=1.0.0
quantum.database.scope=DEV
quantum.database.migration.enabled=true
quantum.database.migration.changeset.package=com.example.migrations
```

Seed packs (filesystem + classpath)

```
# Enable seed pack discovery and application on startup
quantum.seed-pack.enabled=true
quantum.seed-pack.apply.on-startup=true

# Filesystem root for seed packs when present in the app repo
# If the path does not exist, the framework still loads classpath seed packs
# contributed by dependencies (e.g., quantum-framework) from resources under seed-
packs/
quantum.seed-pack.root=src/main/resources/seed-packs

# Optional CSV filter to apply only specific seed pack names
# Example: quantum.seed.apply.filter=system-initialize,my-tenant
quantum.seed.apply.filter=
```

Classpath seed loading: The framework automatically discovers seed packs packaged under `seed-packs/` on the application classpath (including JAR dependencies). Apps depending on the framework no longer need to copy framework seeds locally or reference absolute file paths. If both filesystem and classpath contain seed packs with the same name, the latest semantic version is applied.

Security rules diagnostics

```
# Elevate RuleContext auto‑reload diagnostics from DEBUG to INFO when troubleshooting
quantum.securityrules.ruleContext.debugAutoReload=false
```

Tips:

- In unit tests that run inside Quarkus, prefer using a CDI-managed `RuleContext`. A helper is provided in tests: `RuleContextTestUtils.reload(<realm>)` to force a refresh of policies from the repository for a given realm.
- If you construct `RuleContext` manually (outside CDI), repository injection is not available, and auto-reload is skipped to avoid wiping programmatically added rules.

Realm and Tenant Configuration

```
# System realm configuration
quantum.realmConfig.systemTenantId=system
quantum.realmConfig.systemAccountId=system-account
```

```
quantum.realmConfig.systemOrgRefName=SYSTEM
quantum.realmConfig.systemUserId=system-user

# Default tenant settings
quantum.realmConfig.defaultTenantId=default
quantum.realmConfig.defaultAccountId=default-account
quantum.realmConfig.defaultOrgRefName=DEFAULT

# Anonymous user (use carefully)
quantum.anonymousUserId=anonymous
```

1.5. See Also

- Quarkus Configuration Guide
- Authentication Configuration

1.6. Ontology Configuration (optional)

Enablement

```
e2eq.ontology.enabled=false   # default; set true to enable ontology wiring in your
app
```

TBox Persistence

```
# Enable TBox persistence to MongoDB (default: true)
# When enabled, the ontology TBox (schema) is persisted to MongoDB for faster startup
# and version history. The TBox is automatically rebuilt if the YAML source changes.
quantum.ontology.tbox.persist=true

# Force rebuild of TBox even if persisted version exists (default: false)
# Useful for development or when you want to ensure fresh rebuild from sources
quantum.ontology.tbox.force-rebuild=false

# Package scanning for @OntologyClass and @OntologyProperty annotations
quantum.ontology.scan.packages=com.example.model,com.example.entities
```

Registry modes

- In-memory (recommended to start): build a TBox in code/resources and instantiate OntologyRegistry.inMemory(tbox).
- Mongo-backed (advanced): load a TBox from a Mongo collection and cache it in memory for hot reload/versioning.
- Persisted TBox (recommended for production): TBox is persisted to MongoDB and loaded on startup for faster initialization.

Edges collection and indexes

- Create an edges collection (name of your choice) when enabling ontology. Ensure the following indexes:
- Compound: { tenantId:1, p:1, dst:1 }
- Compound: { tenantId:1, src:1, p:1 }
- Optionally, add a TTL on ts for inferred edges if you plan periodic full recomputes.

Materialization hooks

- Use OntologyMaterializer to recompute inferred edges when entities change (sources and intermediates for your property chains).

See also

- Ontologies in Quantum
- Integrating Ontology