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
```

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
```

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.

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](#)