Manilogix Eureka Server Documentation

# 1. Module Overview

The Eureka Server is a critical component in the Manilogix microservices architecture. It acts as a service registry, enabling dynamic discovery of all registered services such as auth-service, order-service, inventory-service, and others. All microservices register themselves with this Eureka Server, which helps the API Gateway and other services locate each other at runtime.

# 2. Project Structure

eureka-server/  
├── src/main/java/com/manilogix/eureka/  
│ └── EurekaServerApplication.java  
└── resources/  
 └── application.properties

# 3. Main Class (EurekaServerApplication.java)

This class bootstraps the Eureka Server using @EnableEurekaServer.

package com.manilogix.eureka;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(EurekaServerApplication.class, args);  
 }  
}

# 4. Configuration (application.properties)

# === Eureka Server Settings ===  
server.port=8761  
spring.application.name=eureka-server  
  
# Prevent this server from trying to register itself  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false  
  
# Enable dashboard  
eureka.server.enable-self-preservation=false

# 5. Responsibilities

• Acts as a centralized service registry for all microservices.  
• Allows dynamic discovery of services by API Gateway or inter-service communication.  
• Ensures loose coupling between services by resolving their locations at runtime.  
• Shows all registered services via its dashboard (http://localhost:8761).

# 6. How to Use

1. Add the dependency in all services:  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
  
2. In each microservice, configure the Eureka client:  
 eureka.client.service-url.defaultZone=http://localhost:8761/eureka/  
 eureka.instance.prefer-ip-address=true  
  
3. Run Eureka Server first, then run other microservices.  
4. Open http://localhost:8761 to verify service registrations.

# 7. Notes

• Must be up and running before other services start.  
• Works seamlessly with Spring Cloud Gateway for dynamic routing.  
• Should be monitored to ensure registry consistency.  
• Can be extended with Spring Cloud Config or HA setup in production.