Broker Service Microservices Analysis Document

# 1. Overview

This document provides a detailed breakdown of the Broker Service into dedicated microservices, including their responsibilities, utilized fields, data sources, caches, and views. The content is based on Swagger API definitions and internal service design patterns.

# 2. Microservices Breakdown

## 2.1 Legal Entity Identifier (LEI) Service

This microservice handles all operations related to Legal Entity Identifiers. It interacts with the EMMA system to retrieve legal entity details.

Endpoints:

* /lei/defaults – Get default LEIs for settlement broker and product
* /lei/emmaLookup – Lookup LEI details from EMMA via LEI or EMMA ID
* /lei/all – List all EMMA Legal Entity details
* /lei/account – Get LEI details for a specific account
* /lei/fix – Lookup LEI details for a FIX message

Fields Used: settlementBroker, productType, leiString, emmaId, accountId

Data Source: EMMA System

Cache: None explicitly mentioned

## 2.2 Portfolio Permissions Service

Manages broker permissions related to desk and broker code combinations.

Endpoint:

* /portfolios/permissions – Aggregates all permitted/restricted brokers

Fields Used: deskId, brokerCodes[]

Data Source: Portfolio Management System

Cache: Possibly in-memory for quick reference

## 2.3 Mandate Restrictions Service

Handles mandate restrictions and Volcker 23A compliance.

Endpoints:

* /mandates/all – Get all mandate details with EMMA info
* /mandates/volcker (GET/POST) – Retrieve and confirm Volcker 23A scope

Fields Used: mandateId, volckerScope, emmaInfo

Data Source: Mandate Repository, EMMA

Cache: Partial for mandate rules

## 2.4 Cache Management Service

Controls cache lifecycle for guideline and account data.

Endpoints:

* /cache/clearGuidelines
* /cache/loadGuidelines
* /cache/refreshAccount

Fields Used: accountId, productType

Cache: In-memory; controlled by above endpoints

Data Source: Sentinel

## 2.5 Broker Restrictions Service

Handles all broker restrictions and permissions.

Endpoints:

* /brokers/perdesk – List RADAR brokers for a desk
* /brokers/permittedaccounts – Permitted accounts for broker
* /brokers/brokerdetails – Broker metadata
* /brokers/volcker – GS Affiliated Brokers in Volcker scope
* /brokers/permissions – Main broker permission API
* /brokers/health – Broker Service health check
* /brokers/japan – Japanese broker mapping
* /brokers/permissionsperdesk – Aggregate brokers by desk/product

Fields Used: brokerId, deskId, productIds[], radarBrokerId

Cache: Used for broker metadata

Data Sources: RADAR System, Volcker Registry

# 3. Conclusion

The Broker Service has been modularized into distinct microservices to allow independent scalability, ownership, and efficient data handling. The design leverages caching strategies and external data sources for optimal performance.