KnowledgeGraphService

September 4, 2026

KnowledgeGraphService

Kind: Service

Source: `atloria-monorepo/apps/parser-orchestrator/src/knowledge-graph/kg.service.ts`

`KnowledgeGraphService` manages graph entities and relationships for the parser orchestrator. It handles lifecycle connectivity, single and batch upserts, entity lookups, deletions, and relationship creation so other backend components can interact with the knowledge graph through a consistent service API.

Methods

Method	Signature	Returns	Description
`isConnected`	`isConnected()`	`boolean`	Check if Neo4j is available
`onModuleInit`	`onModuleInit()`	`unknown`	
`onModuleDestroy`	`onModuleDestroy()`	`unknown`	
`upsertEntity`	`upsertEntity(entity: Entity)`	`Promise<void>`	Upsert an entity (create or update)
`upsertBatch`	`upsertBatch(entities: Entity[])`	`Promise<void>`	Upsert multiple entities in batch
`getEntity`	`getEntity(id: string)`	`Promise<Entity	null>`	
`getEntitiesByType`	`getEntitiesByType(type: EntityType, limit: unknown)`	`Promise<Entity[]>`	Get entities by type
`deleteEntity`	`deleteEntity(id: string)`	`Promise<void>`	Delete entity by ID
`createRelationship`	`createRelationship(relationship: Relationship)`	`Promise<void>`	Create a relationship between two entities

Method	Signature	Returns	Description
`createRelationshipsBatch`	`createRelationshipsBatch(relationships: Relationship[])`	`Promise<void>`	Create multiple relationships in batch
`getRelationships`	`getRelationships(entityId: string)`	`Promise<Relationship[]>`	Get relationships for an entity
`deleteRelationship`	`deleteRelationship(relationshipId: string)`	`Promise<void>`	Delete a relationship
`getComponentDependencies`	`getComponentDependencies(componentId: string, depth: unknown)`	`Promise<DependencyTree`	null>`
`getAPIFlow`	`getAPIFlow(endpointId: string)`	`Promise<APIFlow`	null>`
`query`	`query(cypher: string, params: Record<string, unknown>)`	`Promise<any[]>`	Execute custom Cypher query
`clearAll`	`clearAll()`	`Promise<void>`	Clear all data (for testing only)
`getStatistics`	`getStatistics()`	`Promise<{`	

```
totalNodes: number;
totalRelationships: number;
nodesByLabel: Record<string, number>;
relationshipsByType: Record<string, number>;
```

}>` | Get statistics about the Knowledge Graph |

Dependencies

- `ConfigService`

Where it refuses work

- `KnowledgeGraphService` stops the work with `Error` when `!this.connected` — “Neo4j is not connected. Knowledge graph features are disabled.”.
- `KnowledgeGraphService` stops the work with an early return when `result.records.length === 0`, in 2 places.
- `KnowledgeGraphService` stops the work with an early return when `entities.length === 0`.
- `KnowledgeGraphService` stops the work with an early return when `relationships.length === 0`.
- `KnowledgeGraphService` stops the work with an early return when `rootResult.records.length === 0`.
- `KnowledgeGraphService` stops the work with an early return when `!api`.

When something fails

- `KnowledgeGraphService` handles failure in 3 places: it logs it and continues in all 3.

Diagram

```
sequenceDiagram
  participant Consumer as Parser/Backend Consumer
  participant KGS as KnowledgeGraphService
  participant Graph as Knowledge Graph Store

  Consumer->>KGS: onModuleInit()
  KGS->>Graph: Establish connection
  Graph-->>KGS: Connected

  Consumer->>KGS: upsertBatch(entities)
  KGS->>Graph: Create or update entities

  Consumer->>KGS: createRelationshipsBatch(relationships)
  KGS->>Graph: Create graph relationships

  Consumer->>KGS: getEntity(id)
  KGS->>Graph: Query entity
  Graph-->>KGS: Entity | null
  KGS-->>Consumer: Entity | null

  Consumer->>KGS: onModuleDestroy()
  KGS->>Graph: Close connection
```

Usage

```
import { Injectable } from '@nestjs/common';
import { KnowledgeGraphService } from './knowledge-graph/kg.service';

@Injectable()
export class DocumentGraphProcessor {
  constructor(
    private readonly knowledgeGraphService: KnowledgeGraphService,
  ) {}

  async indexDocument(): Promise<void> {
    // Confirm the graph client is available before performing writes.
    if (!this.knowledgeGraphService.isConnected()) {
      throw new Error('Knowledge graph service is not connected');
    }

    await this.knowledgeGraphService.upsertBatch([
      {
        id: 'document:123',
        type: 'Document',
        name: 'Architecture Overview',
      },
      {
        id: 'service:parser',
        type: 'Service',
```

```
        name: 'Parser Orchestrator',
      },
    ] as Entity[]);

    await this.knowledgeGraphService.createRelationship(
      'document:123',
      'service:parser',
      'REFERENCES',
    );

    const document = await this.knowledgeGraphService.getEntity(
      'document:123',
    );

    if (document) {
      console.log(`Indexed entity: ${document.id}`);
    }
  }
}
```

AI Coding Instructions

- Inject `KnowledgeGraphService` through NestJS dependency injection; do not create service instances manually.
- Use `upsertBatch` and `createRelationshipsBatch` when processing multiple records to avoid repeated graph operations.
- Check `isConnected()` when work may run before initialization or during shutdown-sensitive background processing.
- Treat `getEntity()` results as nullable and handle the `null` case before accessing entity properties.
- Keep entity IDs and relationship types consistent across parser, orchestration, and graph ingestion code to prevent duplicate or disconnected nodes.

Relationships

- DEPENDS_ON → `configservice`