

Table of Contents

1. Planner and QueryGateway.....	1
1.1. Planner decisions	1
1.2. Public facade: QueryGateway	1
1.3. Examples end-to-end	2
1.4. REST endpoints (v1)	2

Chapter 1. Planner and QueryGateway

The planner decides whether a query runs as a simple single-collection filter (FILTER mode) or as a MongoDB aggregation (AGGREGATION mode). This is transparent to callers.

In v1 the planner recognizes `expand(path)` and selects AGGREGATION mode when present. Otherwise, FILTER mode is used and the existing Morphia path is taken.

1.1. Planner decisions

- FILTER mode: No `expand(...)` in the query. The system generates a Morphia Filter and queries a single collection.
- AGGREGATION mode: One or more `expand(...)` directives are present. The system will build an aggregation pipeline (with `$lookup`, etc.). In v1, a placeholder pipeline is returned to establish the contract.

1.2. Public facade: QueryGateway

A new collection-agnostic facade exposes plan/inspect capabilities and will later execute aggregation pipelines when enabled.

Java

```
import com.e2eq.framework.model.persistent.morphia.query.QueryGateway;
import com.e2eq.framework.model.persistent.morphia.query.QueryGatewayImpl;
import com.e2eq.framework.model.persistent.morphia.planner.PlannerResult;

QueryGateway gateway = new QueryGatewayImpl();

// Decide which mode would be used
PlannerResult pr = gateway.plan("expand(customer) && status:active", Order.class);
// pr.getMode() == PlannerResult.Mode.AGGREGATION
// pr.getExpandPaths() == ["customer"]
```

If you prefer a lower-level API:

Java

```
import com.e2eq.framework.model.persistent.morphia.MorphiaUtils;
import com.e2eq.framework.model.persistent.morphia.planner.PlannedQuery;

PlannedQuery planned = MorphiaUtils.convertToPlannedQuery(
    "expand(items[*].product) && status:active",
    Order.class
);

switch (planned.getMode()) {
case FILTER:
```

```
// Use planned.getFilter() with Morphia datastore
break;
case AGGREGATION:
    // Use planned.getAggregation() to inspect the pipeline (placeholder in v1)
    break;
}
```

1.3. Examples end-to-end

- No expansion (FILTER mode):

```
q = "realm:acme && status:active && fields:[+_id,+total]"
```

- With expansion (AGGREGATION mode):

```
q = "expand(customer, fields:[+name,+tier]) && status:active &&
fields:[+_id,+customer.name]"
```

See also: [Relationship hydration with expand\(path\)](#).

1.4. REST endpoints (v1)

Two helper endpoints expose the planner and execution in FILTER mode:

- POST /api/query/plan — returns planner mode and expand paths
- POST /api/query/find — executes FILTER-mode queries and returns the standard Collection<T> envelope; returns 501 when AGGREGATION is requested (until aggregation compiler is enabled)

Request: plan

```
{
  "rootType": "com.e2eq.framework.model.security.UserProfile",
  "query": "expand(customer) && status:active"
}
```

Response: plan

```
{
  "mode": "AGGREGATION",
  "expandPaths": ["customer"]
}
```

Request: find (FILTER mode)

```
{
```

```
"rootType": "com.e2eq.framework.model.security.UserProfile",  
"query": "displayName:*Alice*",  
"page": { "limit": 10, "skip": 0 }  
}
```

Response: find (Collection envelope)

```
{  
  "offset": 0,  
  "limit": 10,  
  "rows": [ { "displayName": "Alice T", "refName": "alice" } ],  
  "rowCount": 1,  
  "filter": "displayName:*Alice*"  
}
```