Table of Contents

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

public class PlannerSnippet1 {
  static class Order {}
  public static void main(String[] args) {
    QueryGateway gateway = new QueryGatewayImpl();
    PlannerResult pr = gateway.plan("expand(customer) && status:active", Order.class);
    System.out.println(pr.getMode());        // AGGREGATION
    System.out.println(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). For concrete pipelines, see Aggregation examples: what expand(...) compiles to.

1.4. REST endpoints (v1)

Two helper endpoints expose the planner and execution path. FILTER mode executes immediately. AGGREGATION mode is planned, and execution is guarded by a feature flag.

- POST /api/query/plan — returns planner mode and expand paths detected
- POST /api/query/find — executes FILTER-mode queries and returns the standard Collection<T> envelope; when AGGREGATION is requested, behavior depends on a feature flag (see below)

1.4.1. Request/Response: /api/query/plan

Request body

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

Response body

```
{
```

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

Notes

- mode is FILTER when no expand(...) directives are present; otherwise AGGREGATION.
- expandPaths lists the paths detected by the planner for hydration. See Query Expansion.

1.4.2. Request/Response: /api/query/find

Parameters in the request body

- rootType: fully-qualified class name extending UnversionedBaseModel
- query: BIAPI query string (see Query Language Reference)
- page: optional object with limit and skip
- sort: optional array of { field, dir } objects, where dir is ASC or DESC
- realm: optional, overrides the default realm/database

Request body (FILTER mode)

```
{
  "rootType": "com.e2eq.framework.model.security.UserProfile",
  "query": "displayName:*Alice*",
  "page": { "limit": 10, "skip": 0 },
  "sort": [ { "field": "createdDate", "dir": "DESC" } ],
  "realm": "acme"
}
```

Response body (Collection envelope)

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

Behavior when AGGREGATION is requested

- If the planner selects AGGREGATION (due to expand(...)) and the configuration property feature.queryGateway.execution.enabled=false (default), the endpoint responds with HTTP 501 Not Implemented and a message indicating execution is disabled.
- If feature.queryGateway.execution.enabled=true, the gateway attempts to execute the aggregation pipeline. If the pipeline contains an unresolved $lookup.from ("*unknown*"), the

endpoint responds with HTTP 422 Unprocessable Entity and an explanatory error. Otherwise, results are returned in the same Collection envelope shape, with rows being Documents.

cURL examples

Planning with expansion

```
curl -sS -X POST \
  -H 'Content-Type: application/json' \
  localhost:8080/api/query/plan \
  -d '{
    "rootType":"com.e2eq.framework.model.security.UserProfile",
    "query":"expand(customer) && status:active"
  }'
```

Finding with paging, sort, realm (FILTER path)

```
curl -sS -X POST \
  -H 'Content-Type: application/json' \
  localhost:8080/api/query/find \
  -d '{
    "rootType":"com.e2eq.framework.model.security.UserProfile",
    "query":"displayName:*Alice*",
    "page": { "limit": 10, "skip": 0 },
    "sort": [ { "field": "createdDate", "dir": "DESC" } ],
    "realm":"acme"
  }'
```

See also: Relationship hydration with expand(path).