

# operationDetails Parameter

The `operationDetails` parameter allows you to fine-tune query operations with operational directives similar to GraphQL capabilities. It accepts a JSON array containing one or more operation directives that refine the shape and scope of returned objects.

## Parameter Format

```
json
[
  {
    "opType": "<operation_type>",
    "<operation_specific_details>": [...]
  }
]
```

## Supported Operation Types

### 1. projections - Select Specific Attributes

Specify which attributes to include for particular object types (similar to GraphQL field selection).

#### Format:

```
json
{
  "opType": "projections",
  "projectionsDetails": [
    {
      "type": "<ClassName>",
      "attrs": ["attribute1", "attribute2", "..."]
    }
  ]
}
```

**Example:** Only retrieve `name`, `id`, and `email` from Employee objects:

```
json
```

```
{
  "opType": "projections",
  "projectionsDetails": [{
    "type": "Employee",
    "attrs": ["name", "id", "email"]
  }]
}
```

## 2. ignore - Skip Object References

Exclude specific referenced objects from deep queries (useful for avoiding circular references or unnecessary data).

### Format:

```
json
{
  "opType": "ignore",
  "references": ["<ClassName>", "<attributeName>"]
}
```

**Example:** Ignore the `departments` reference in Employee objects:

```
json
[{
  "opType": "ignore",
  "references": ["Employee", "departments"]
}]
```

## 3. follow - Include Object References

Force inclusion of specific referenced objects in shallow queries.

### Format:

```
json
{
  "opType": "follow",
  "references": ["<ClassName>", "<attributeName>"]
}
```

**Example:** Include the `manager` reference in Employee objects:

```
json
[
  {
    "opType": "follow",
    "references": ["Employee", "manager"]
  }
]
```

## 4. filters - Apply Predicates

Filter objects of specific types using predicate expressions.

**Format:**

```
json
{
  "opType": "filters",
  "predicates": [
    {
      "type": "<ClassName>",
      "predicate": "<filter_expression>"
    }
  ]
}
```

**Example:** Only include active employees with salary > 50000:

```
json
[
  {
    "opType": "filters",
    "predicates": [
      {
        "type": "Employee",
        "predicate": "active = true AND salary > 50000"
      }
    ]
  }
]
```

## Combining Multiple Operations

You can combine multiple operation types in a single `operationDetails` array:

```
json
```

```
[
  {
    "opType": "projections",
    "projectionsDetails": [{
      "type": "Employee",
      "attrs": ["name", "id", "salary"]
    }]
  },
  {
    "opType": "filters",
    "predicates": [{
      "type": "Employee",
      "predicate": "department = 'Engineering'"
    }]
  },
  {
    "opType": "ignore",
    "references": ["Employee", "personalDetails"]
  }
]
```

## Usage Tips

- **Empty string or null:** No operational directives applied
- **projections:** Most useful for reducing payload size and improving performance
- **ignore/follow:** Control object graph traversal depth and scope
- **filters:** Apply server-side filtering before data transmission
- **Combine operations:** Mix and match operations for precise data control

## MCP Client Implementation

When calling the query tool, pass the operationDetails as either:

- **JSON object/array** (if your MCP client auto-parses JSON)
- **JSON string** (if your MCP client expects string parameters)

The MCP server will automatically handle both formats.