Elasticsearch support for Java source code

CAST supports **Elasticsearch** via its NoSQL for Java extension. Details about how this support is provided for **Java source code** is discussed below.

Supported Client Libraries

TransportClient	✓
LowLevelRestClient	✓
HIghLevelRestClient	✓
JestClient	✓

Supported Operations

Operation	Methods Supported
Insert	• index • prepareIndex • indexAsync
Update	• update • prepareUpdate • updateAsync

Select	• get • prepareGet • multiGet • multigetAsync • prepareMultiGet • multiSearch • multisearchAsync • prepareMultiSearch • search • searchAsync • prepareSearch • searchScroll • searchScrollAsync • explain • prepareExplain • exists • existsAsync • fieldCaps • execute • executeAsync
Delete	• delete • prepareDelete

Objects

Icon	Description
	Java Elasticsearch Cluster
	Java Elasticsearch Index
	Java Unknown Elasticsearch Cluster
	Java Unknown Elasticsearch Index

Links

Links are created for transaction and function point needs:

Link type	Source and destination of link	Methods supported
belongsTo	From Java Elasticsearch Index object to Java Elasticsearch Cluster object	
useLink	Between the caller .NET Class / Method objects and Java Elasticsearch Index objects	• bulk • bulkAsync • performRequest • performRequestAsync
useInsertLink		• index • prepareIndex • indexAsync
useDeleteLink		• delete • prepareDelete

useSelectLink		• get • prepareGet • multiGet • multigetAsync • prepareMultiGet • multiSearch • multisearchAsync • prepareMultiSearch • search • searchAsync • prepareSearch • searchScroll • searchScrollAsync • explain • prepareExplain • exists • existsAsync • execute • fieldCaps • executeAsync
useUpdateLink		• update • prepareUpdate • updateAsync

What results can you expect?

Once the analysis/snapshot generation has completed, you can view the results in the normal manner (for example via CAST Enlighten). Some examples are shown below.

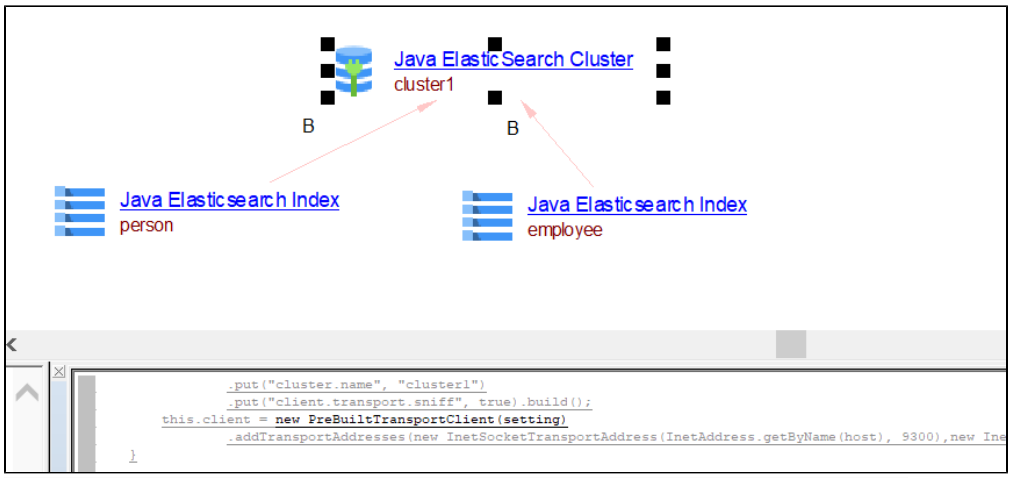
Cluster and Index creation

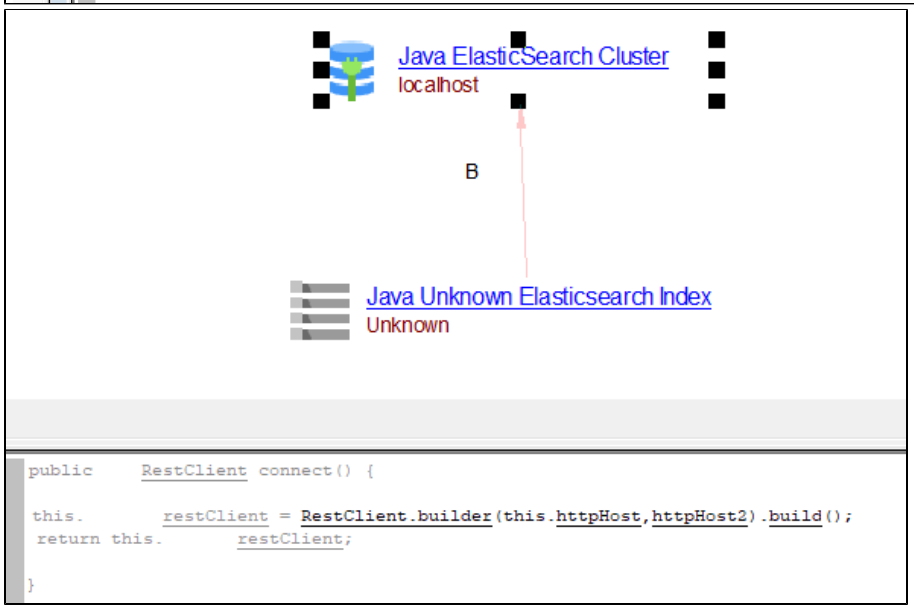
Cluster and Index creation

```
 public test() throws UnknownHostException {
            Settings setting = Settings.builder()
                .put("cluster.name", "cluster1")
                .put("client.transport.sniff", true).build();
        this.client = new PreBuiltTransportClient(setting)
                .addTransportAddresses(new InetSocketTransportAddress(InetAddress.getByName(host), 9300),new
InetSocketTransportAddress(InetAddress.getByName(host), 9301));
 }
```

cluster creation

```
public class restClientConnection {

        private HttpHost httpHost;
        private RestClient restClient;
        private static HttpHost httpHost2 = new HttpHost("localhost",9303,"http");

        public restClientConnection(String host, int port, String protocol) {

                this.httpHost = new HttpHost(host, port, protocol);
        }

        public RestClient connect() {

                this.restClient = RestClient.builder(this.httpHost,httpHost2).build();
                return this.restClient;

        }
}
```





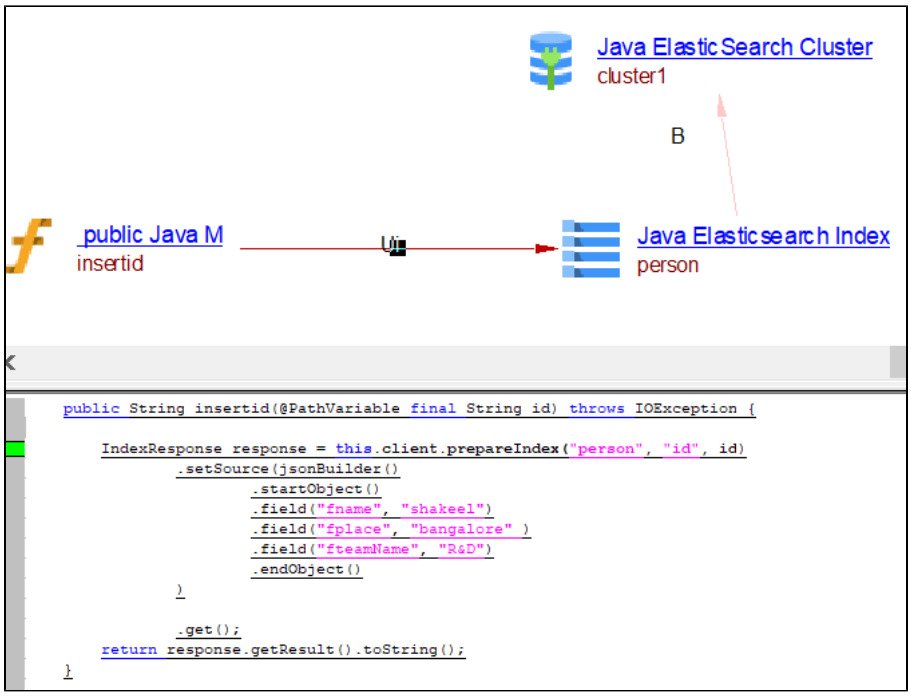
Insert Operation

IndexDocument

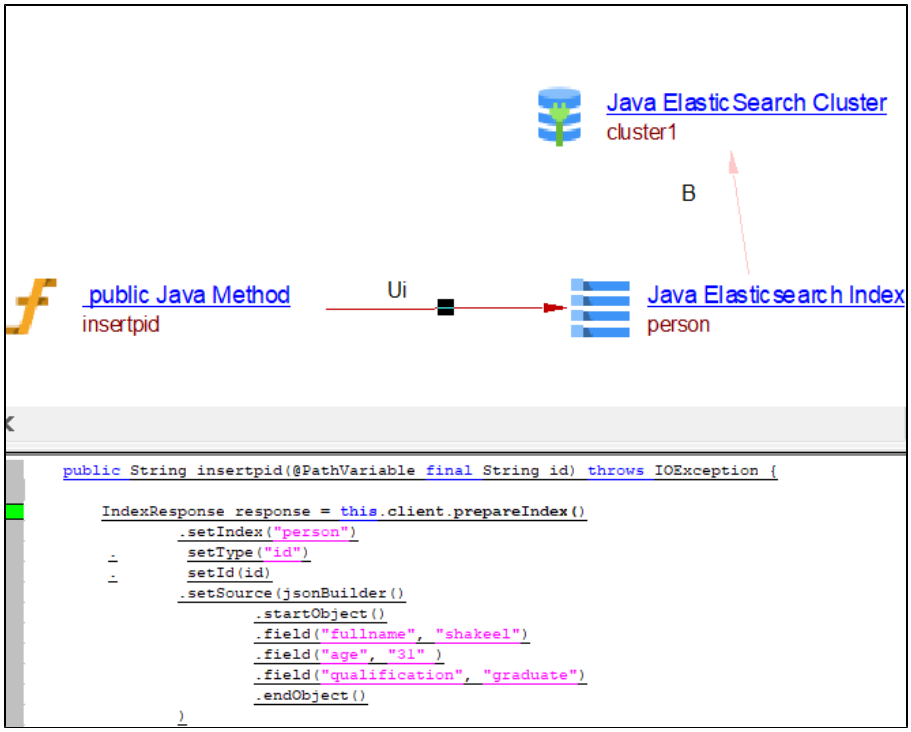
```
public String insertid(@PathVariable final String id) throws IOException {

	IndexResponse response = this.client.prepareIndex("person", "id", id)
			.setSource(jsonBuilder()
					.startObject()
					.field("fname", "shakeel")
					.field("fplace", "bangalore" )
					.field("fteamName", "R&D")
					.endObject()
			)

			.get();
	return response.getResult().toString();
   }
```

IndexDocumentAsync

```
public String insertpid(@PathVariable final String id) throws IOException {

        IndexResponse response = this.client.prepareIndex()
                        .setIndex("person")
                        .setType("id")
                        .setId(id)
                .setSource(jsonBuilder()
                        .startObject()
                        .field("fullname", "shakeel")
                        .field("age", "31" )
                        .field("qualification", "graduate")
                        .endObject()
                )

                .get();
        return response.getResult().toString();
    }
```




```
public String insertid(@PathVariable final String id) throws IOException {

    IndexResponse response = this.client.prepareIndex("person", "id", id)
            .setSource(jsonBuilder()
                    .startObject()
                    .field("fname", "shakeel")
                    .field("fplace", "bangalore" )
                    .field("fteamName", "R&D")
                    .endObject()
            )

            .get();
    return response.getResult().toString();
}
```




```
public String insertpid(@PathVariable final String id) throws IOException {

    IndexResponse response = this.client.prepareIndex()
            .setIndex("person")
            .setType("id")
            .setId(id)
            .setSource(jsonBuilder()
                    .startObject()
                    .field("fullname", "shakeel")
                    .field("age", "31" )
                    .field("qualification", "graduate")
                    .endObject()
            )
```

Update Operation

Update

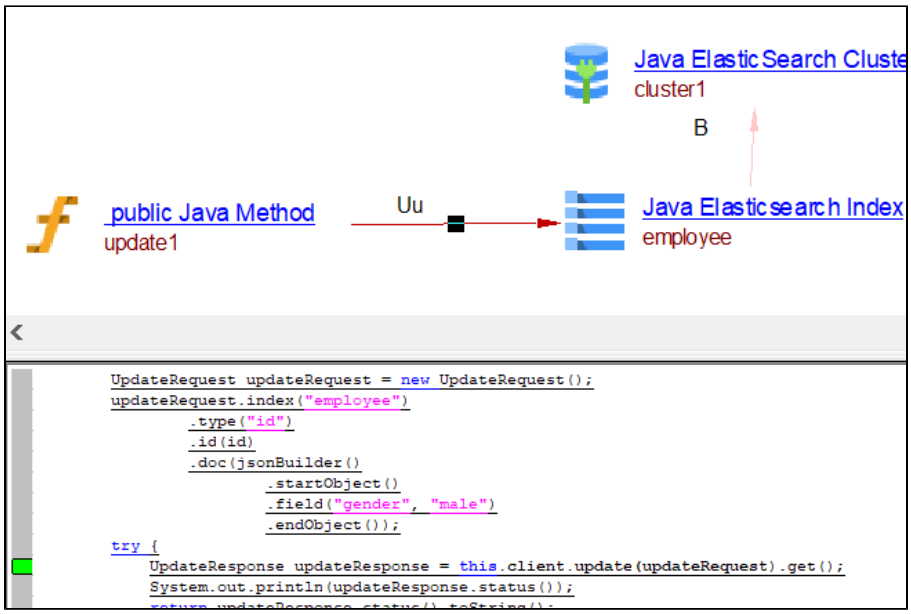
```
 public String update1(@PathVariable final String id) throws IOException {

        UpdateRequest updateRequest = new UpdateRequest();
        updateRequest.index("employee")
                .type("id")
                .id(id)
                .doc(jsonBuilder()
                        .startObject()
                        .field("gender", "male")
                        .endObject());
        try {
            UpdateResponse updateResponse = this.client.update(updateRequest).get();
            System.out.println(updateResponse.status());
            return updateResponse.status().toString();
        } catch (InterruptedException | ExecutionException e) {
            System.out.println(e);
        }
        return "Exception";
    }
```

Update

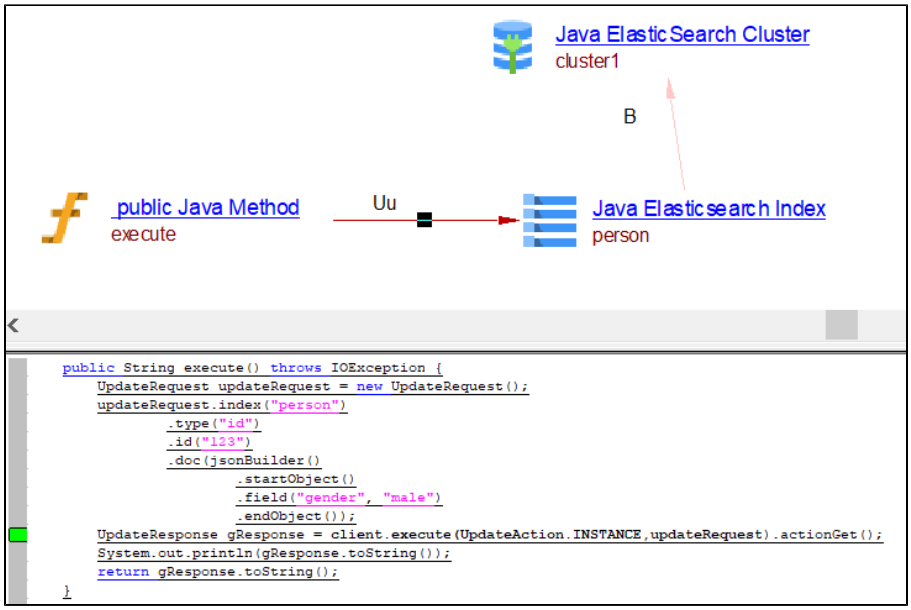
```
 public String update1(@PathVariable final String id) throws IOException {

        UpdateRequest updateRequest = new UpdateRequest();
        updateRequest.index("employee")
                .type("id")
                .id(id)
                .doc(jsonBuilder()
                        .startObject()
                        .field("gender", "male")
                        .endObject());
        try {
            UpdateResponse updateResponse = this.client.update(updateRequest).get();
            System.out.println(updateResponse.status());
            return updateResponse.status().toString();
        } catch (InterruptedException | ExecutionException e) {
            System.out.println(e);
        }
        return "Exception";
    }
```

Update Execute

```
public String execute() throws IOException {
            UpdateRequest updateRequest = new UpdateRequest();
        updateRequest.index("person")
                .type("id")
                .id("123")
                .doc(jsonBuilder()
                        .startObject()
                        .field("gender", "male")
                        .endObject());
            UpdateResponse gResponse = client.execute(UpdateAction.INSTANCE,updateRequest).actionGet();
            System.out.println(gResponse.toString());
        return gResponse.toString();
    }
```




```
UpdateRequest updateRequest = new UpdateRequest();
updateRequest.index("employee")
        .type("id")
        .id(id)
        .doc(jsonBuilder()
                .startObject()
                .field("gender", "male")
                .endObject());
try {
    UpdateResponse updateResponse = this.client.update(updateRequest).get();
    System.out.println(updateResponse.status());
```

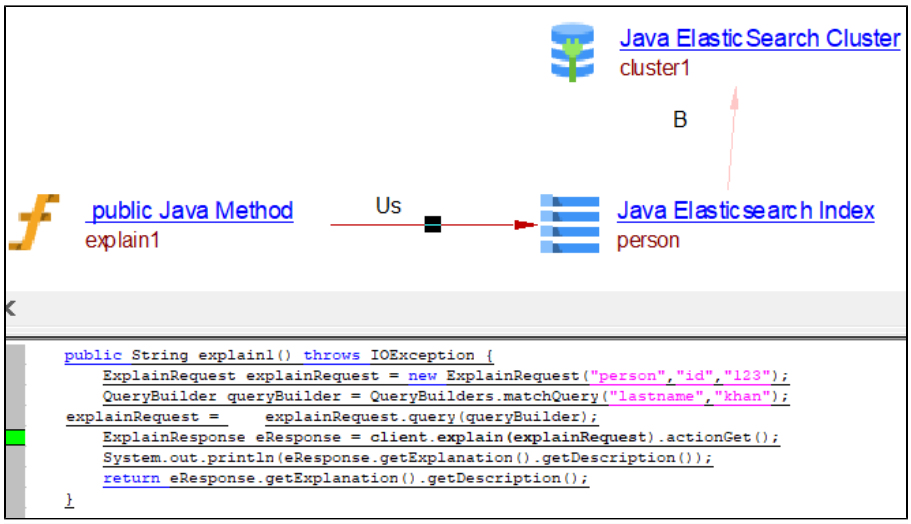



```
public String execute() throws IOException {
    UpdateRequest updateRequest = new UpdateRequest();
    updateRequest.index("person")
            .type("id")
            .id("123")
            .doc(jsonBuilder()
                    .startObject()
                    .field("gender", "male")
                    .endObject());
    UpdateResponse gResponse = client.execute(UpdateAction.INSTANCE,updateRequest).actionGet();
    System.out.println(gResponse.toString());
    return gResponse.toString();
}
```

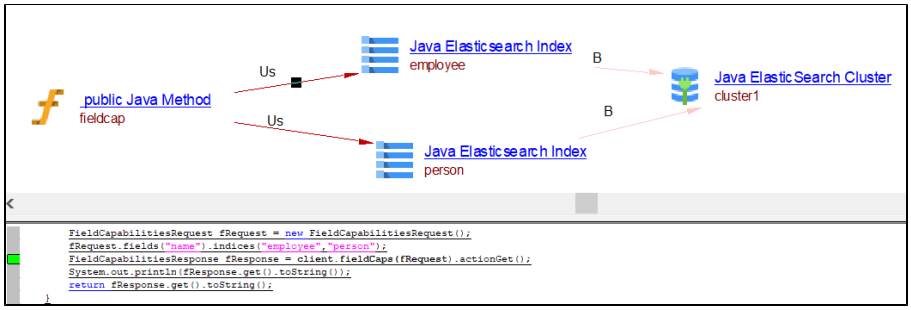
Select Operation

Select

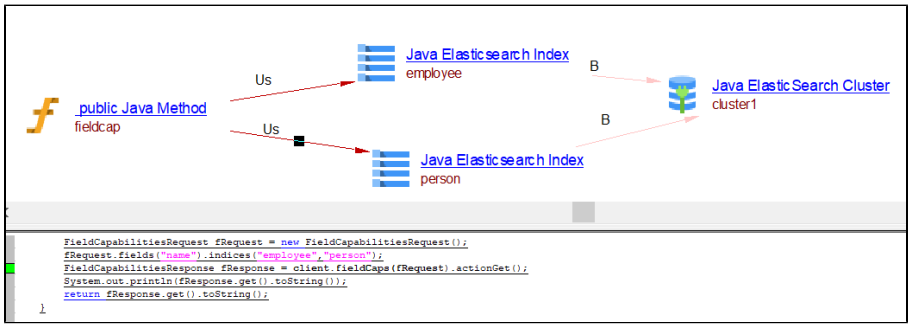
```
public String explain1() throws IOException {
            ExplainRequest explainRequest = new ExplainRequest("person","id","123");
            QueryBuilder queryBuilder = QueryBuilders.matchQuery("lastname","khan");
            explainRequest = explainRequest.query(queryBuilder);
            ExplainResponse eResponse = client.explain(explainRequest).actionGet();
            System.out.println(eResponse.getExplanation().getDescription());
        return eResponse.getExplanation().getDescription();
    }

public String fieldcap() throws IOException {
            FieldCapabilitiesRequest fRequest = new FieldCapabilitiesRequest();
            fRequest.fields("name").indices("employee","person");
            FieldCapabilitiesResponse fResponse = client.fieldCaps(fRequest).actionGet();
            System.out.println(fResponse.get().toString());
        return fResponse.get().toString();
    }
```




```
public String explain1() throws IOException {
    ExplainRequest explainRequest = new ExplainRequest("person","id","123");
    QueryBuilder queryBuilder = QueryBuilders.matchQuery("lastname","khan");
explainRequest =     explainRequest.query(queryBuilder);
    ExplainResponse eResponse = client.explain(explainRequest).actionGet();
    System.out.println(eResponse.getExplanation().getDescription());
    return eResponse.getExplanation().getDescription();
}
```




```
FieldCapabilitiesRequest fRequest = new FieldCapabilitiesRequest();
fRequest.fields("name").indices("employee","person");
FieldCapabilitiesResponse fResponse = client.fieldCaps(fRequest).actionGet();
System.out.println(fResponse.get().toString());
return fResponse.get().toString();
}
```

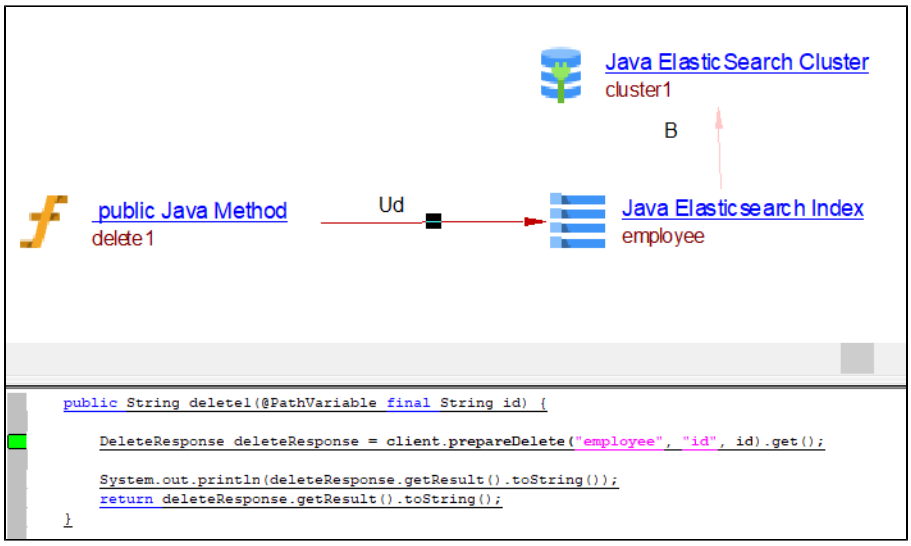



```
FieldCapabilitiesRequest fRequest = new FieldCapabilitiesRequest();
fRequest.fields("name").indices("employee","person");
FieldCapabilitiesResponse fResponse = client.fieldCaps(fRequest).actionGet();
System.out.println(fResponse.get().toString());
return fResponse.get().toString();
}
```

Delete Operation

Delete

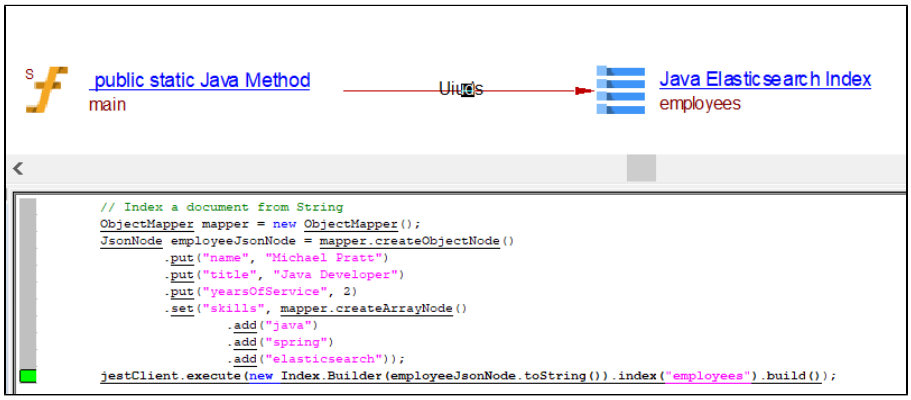
```
public String delete1(@PathVariable final String id) {

        DeleteResponse deleteResponse = client.prepareDelete("employee", "id", id).get();

        System.out.println(deleteResponse.getResult().toString());
        return deleteResponse.getResult().toString();
    }
```



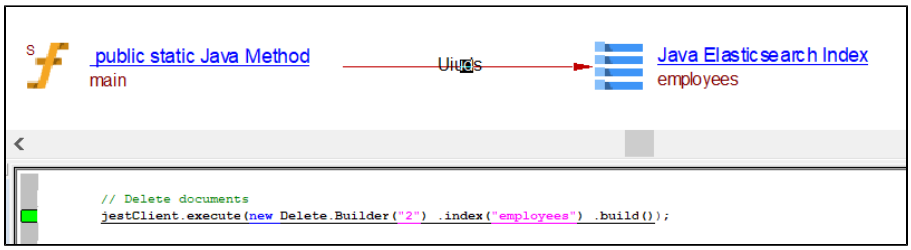
Jest Elasticsearch Java Client

APIs such as execute and executeAsync are used to perform all the CRUD operations. To identify which operation to be performed, type of Builder Request is analyzed.

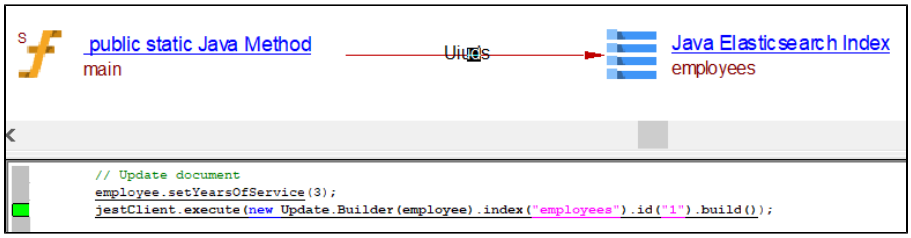
```
public static void main(String[] args) throws IOException {
                // Demo the JestClient
        JestClient jestClient = jestClient();
                // Index a document from String
        ObjectMapper mapper = new ObjectMapper();
        JsonNode employeeJsonNode = mapper.createObjectNode()
                .put("name", "Michael Pratt")
                .put("title", "Java Developer")
                .put("yearsOfService", 2)
                .set("skills", mapper.createArrayNode()
                        .add("java")
                        .add("spring")
                        .add("elasticsearch"));
        jestClient.execute(new Index.Builder(employeeJsonNode.toString()).index("employees").build());
}
```



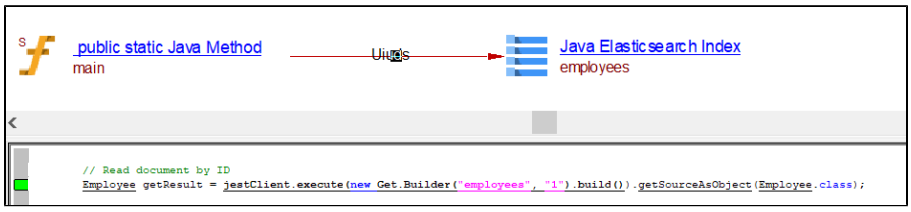

```
public static void main(String[] args) throws IOException {
                // Demo the JestClient
        JestClient jestClient = jestClient();
                // Delete documents
        jestClient.execute(new Delete.Builder("2") .index("employees") .build());
}
```



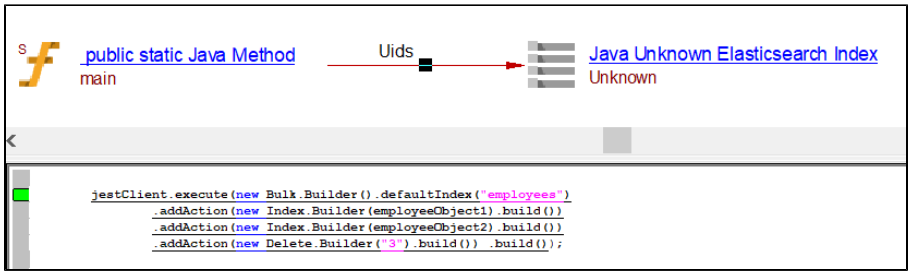

```
public static void main(String[] args) throws IOException {
                // Demo the JestClient
        JestClient jestClient = jestClient();
                // Update document
        employee.setYearsOfService(3);
        jestClient.execute(new Update.Builder(employee).index("employees").id("1").build());
}
```




```
public static void main(String[] args) throws IOException {
                // Demo the JestClient
        JestClient jestClient = jestClient();
                // Read document by ID
        Employee getResult = jestClient.execute(new Get.Builder("employees", "1").build()).getSourceAsObject
(Employee.class);
}
```



```
public static void main(String[] args) throws IOException {
                // Demo the JestClient
        JestClient jestClient = jestClient();

                // Bulk operations
        Employee employeeObject1 = new Employee();
        employee.setName("John Smith");
        employee.setTitle("Python Developer");
        employee.setYearsOfService(10);
        employee.setSkills(Arrays.asList("python"));

        Employee employeeObject2 = new Employee();
        employee.setName("Kate Smith");
        employee.setTitle("Senior JavaScript Developer");
        employee.setYearsOfService(10);
        employee.setSkills(Arrays.asList("javascript", "angular"));

        jestClient.execute(new Bulk.Builder().defaultIndex("employees")
                .addAction(new Index.Builder(employeeObject1).build())
                .addAction(new Index.Builder(employeeObject2).build())
                .addAction(new Delete.Builder("3").build()) .build());
}
```



Evolution

- Increased resolution for cluster and index names
- Support for APIs such as execute(), fieldcaps
- Support for Jest Elasticsearch Client

Limitations

- APIs such as bulk (only for Transport Client, High Level Client) and performRequst result in useLinks
- Bulk API used in Jest Client always results in appropriate CRUD links with unknown Index.
- Multiple clusters used for CRUD operations are not supported. Only first found cluster is used.

Future development

- Resolving clusters used for CRUD operations when multiple clusters are present in the project.