

---

# **ESEngine**

***Release latest***

January 07, 2016



<b>1</b>	<b>Payload</b>	<b>3</b>
1.1	API . . . . .	3
1.2	Exceptions . . . . .	3
<b>2</b>	<b>Queries</b>	<b>5</b>
2.1	Query.span_or . . . . .	5
2.2	Query.terms . . . . .	5
2.3	Query.has_child . . . . .	5
2.4	Query.span_first . . . . .	5
2.5	Query.prefix . . . . .	5
2.6	Query.term . . . . .	5
2.7	Query.fuzzy . . . . .	6
2.8	Query.nested . . . . .	6
2.9	Query.dis_max . . . . .	6
2.10	Query.query_string . . . . .	6
2.11	Query.fuzzy_like_this . . . . .	6
2.12	Query.has_parent . . . . .	6
2.13	Query.function_score . . . . .	6
2.14	Query.geo_shape . . . . .	6
2.15	Query.fuzzy_like_this_field . . . . .	6
2.16	Query.span_multi . . . . .	7
2.17	Query.match_all . . . . .	7
2.18	Query.span_near . . . . .	7
2.19	Query.simple_query_string . . . . .	7
2.20	Query.multi_match . . . . .	7
2.21	Query.span_term . . . . .	7
2.22	Query.regexp . . . . .	7
2.23	Query.ids . . . . .	7
2.24	Query.more_like_this . . . . .	7
2.25	Query.range . . . . .	8
2.26	Query.bool . . . . .	8
2.27	Query.common . . . . .	8
2.28	Query.wildcard . . . . .	8
2.29	Query.indices . . . . .	8
2.30	Query.filtered . . . . .	8
2.31	Query.span_not . . . . .	8
2.32	Query.boost . . . . .	8
2.33	Query.constant_score . . . . .	8

2.34	Query.match . . . . .	9
2.35	Query.top_children . . . . .	9
<b>3</b>	<b>Aggregates</b>	<b>11</b>
3.1	Aggregate.geo_bounds . . . . .	11
3.2	Aggregate.date_histogram . . . . .	11
3.3	Aggregate.global . . . . .	11
3.4	Aggregate.nested . . . . .	11
3.5	Aggregate.ip_range . . . . .	11
3.6	Aggregate.filters . . . . .	11
3.7	Aggregate.avg . . . . .	12
3.8	Aggregate.children . . . . .	12
3.9	Aggregate.stats . . . . .	12
3.10	Aggregate.scripted_metric . . . . .	12
3.11	Aggregate.min . . . . .	12
3.12	Aggregate.sum . . . . .	12
3.13	Aggregate.extended_stats . . . . .	12
3.14	Aggregate.value_count . . . . .	12
3.15	Aggregate.percentiles . . . . .	12
3.16	Aggregate.terms . . . . .	13
3.17	Aggregate.missing . . . . .	13
3.18	Aggregate.max . . . . .	13
3.19	Aggregate.histogram . . . . .	13
3.20	Aggregate.date_range . . . . .	13
3.21	Aggregate.cardinality . . . . .	13
3.22	Aggregate.geohash_grid . . . . .	13
3.23	Aggregate.geo_distance . . . . .	13
3.24	Aggregate.filter . . . . .	13
3.25	Aggregate.percentile_ranks . . . . .	14
3.26	Aggregate.range . . . . .	14
3.27	Aggregate.significant_terms . . . . .	14
3.28	Aggregate.top_hits . . . . .	14
3.29	Aggregate.reverse_nested . . . . .	14
<b>4</b>	<b>Suggesters</b>	<b>15</b>
4.1	Suggester.completion . . . . .	15
4.2	Suggester.phrase . . . . .	15
4.3	Suggester.term . . . . .	15
<b>5</b>	<b>Filters</b>	<b>17</b>
5.1	Filter.geohash_shell . . . . .	17
5.2	Filter.geo_polygon . . . . .	17
5.3	Filter.exists . . . . .	17
5.4	Filter.not_ . . . . .	17
5.5	Filter.nested . . . . .	17
5.6	Filter.prefix . . . . .	17
5.7	Filter.has_parent . . . . .	18
5.8	Filter.geo_distance_range . . . . .	18
5.9	Filter.script . . . . .	18
5.10	Filter.bool . . . . .	18
5.11	Filter.type . . . . .	18
5.12	Filter.terms . . . . .	18
5.13	Filter.has_child . . . . .	18
5.14	Filter.missing . . . . .	18

5.15	Filter.term . . . . .	18
5.16	Filter.geo_shape . . . . .	19
5.17	Filter.regexp . . . . .	19
5.18	Filter.or_ . . . . .	19
5.19	Filter.match_all . . . . .	19
5.20	Filter.geo_distance . . . . .	19
5.21	Filter.geo_bounding_box . . . . .	19
5.22	Filter.and_ . . . . .	19
5.23	Filter.ids . . . . .	19
5.24	Filter.range . . . . .	19
5.25	Filter.limit . . . . .	20
5.26	Filter.indices . . . . .	20
<b>6</b>	<b>Meta</b>	<b>21</b>
<b>7</b>	<b>Credits</b>	<b>23</b>



ESEngine is an ODM (Object Document Mapper) it maps Python classes in to Elasticsearch index/doc\_type and object instances() in to Elasticsearch documents and also provides a Pythonic query builder.

Install with `pip install esengine`





---

# Payload

---

The main `Payload` class allows you to build Elasticsearch queries

```
from eengine import Payload, Query, Aggregate

# Create a new payload
p = payload()

# Match something
p.query(Query.match('some_field', 'some_text'))

# Aggregate something
p.aggregate(Aggregate.terms('my_terms', 'some_field'))
```

## 1.1 API

## 1.2 Exceptions



---

## Queries

---

Note that all Query calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

### 2.1 Query.span\_or

```
Query.span_or([Query])
```

### 2.2 Query.terms

```
Query.terms(field, [value])
```

### 2.3 Query.has\_child

```
Query.has_child(type, filter=Filter, query=Query)
```

### 2.4 Query.span\_first

```
Query.span_first(Query)
```

### 2.5 Query.prefix

```
Query.prefix(field, value, boost=None)
```

### 2.6 Query.term

```
Query.term(field, value, boost=None)
```

## 2.7 Query.fuzzy

```
Query.fuzzy(field, value, boost=None, fuzziness=None, prefix_length=None, max_expansions=None)
```

## 2.8 Query.nested

```
Query.nested(path, Query)
```

## 2.9 Query.dis\_max

```
Query.dis_max([Query])
```

## 2.10 Query.query\_string

```
Query.query_string(query, fields=[])
```

## 2.11 Query.fuzzy\_like\_this

```
Query.fuzzy_like_this([fields], like_text)
```

## 2.12 Query.has\_parent

```
Query.has_parent(parent_type, filter=Filter, query=Query)
```

## 2.13 Query.function\_score

```
Query.function_score([functions], filter=Filter, query=Query)
```

## 2.14 Query.geo\_shape

```
Query.geo_shape(field, type=None, coordinates=[])
```

## 2.15 Query.fuzzy\_like\_this\_field

```
Query.fuzzy_like_this_field(field, like_text, max_query_terms=None, ignore_tf=None, fuzziness=None, p
```

## 2.16 Query.span\_multi

```
Query.span_multi(Query)
```

## 2.17 Query.match\_all

```
Query.match_all(boost=None)
```

## 2.18 Query.span\_near

```
Query.span_near([Query])
```

## 2.19 Query.simple\_query\_string

```
Query.simple_query_string(query, fields=[])
```

## 2.20 Query.multi\_match

```
Query.multi_match([fields], query)
```

## 2.21 Query.span\_term

```
Query.span_term(field, value, boost=None)
```

## 2.22 Query.regexp

```
Query.regexp(field, value, boost=None, flags=None)
```

## 2.23 Query.ids

```
Query.ids([values], type=None)
```

## 2.24 Query.more\_like\_this

```
Query.more_like_this([fields], like_text)
```

## 2.25 Query.range

```
Query.range(field, gte=None, gt=None, lte=None, lt=None)
```

## 2.26 Query.bool

```
Query.bool(must=[Query], must_not=[Query], should=[Query])
```

## 2.27 Query.common

```
Query.common(query)
```

## 2.28 Query.wildcard

```
Query.wildcard(field, value, boost=None)
```

## 2.29 Query.indices

```
Query.indices([indices], query=Query, no_match_query=Query)
```

## 2.30 Query.filtered

```
Query.filtered(filter=Filter, query=Query)
```

## 2.31 Query.span\_not

```
Query.span_not(include=Query, exclude=Query)
```

## 2.32 Query.boost

```
Query.boost(positive=None, negative=None)
```

## 2.33 Query.constant\_score

```
Query.constant_score(filter=Filter, query=Query)
```

## 2.34 Query.match

```
Query.match(field, query, operator=None, zero_terms_query=None, cutoff_frequency=None, boost=None)
```

## 2.35 Query.top\_children

```
Query.top_children(type, query=Query)
```





---

## Aggregates

---

Note that all Aggregate calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

### 3.1 Aggregate.geo\_bounds

```
Aggregate.geo_bounds(name, field)
```

### 3.2 Aggregate.date\_histogram

```
Aggregate.date_histogram(name, field, interval)
```

### 3.3 Aggregate.global

```
Aggregate.global(name)
```

### 3.4 Aggregate.nested

```
Aggregate.nested(name, path)
```

### 3.5 Aggregate.ip\_range

```
Aggregate.ip_range(name, field, [ranges])
```

### 3.6 Aggregate.filters

```
Aggregate.filters(name, [Filter])
```

### 3.7 Aggregate.avg

```
Aggregate.avg(name, field)
```

### 3.8 Aggregate.children

```
Aggregate.children(name, type)
```

### 3.9 Aggregate.stats

```
Aggregate.stats(name, field)
```

### 3.10 Aggregate.scripted\_metric

```
Aggregate.scripted_metric(name)
```

### 3.11 Aggregate.min

```
Aggregate.min(name, field)
```

### 3.12 Aggregate.sum

```
Aggregate.sum(name, field)
```

### 3.13 Aggregate.extended\_stats

```
Aggregate.extended_stats(name, field)
```

### 3.14 Aggregate.value\_count

```
Aggregate.value_count(name, field)
```

### 3.15 Aggregate.percentiles

```
Aggregate.percentiles(name, field)
```

### 3.16 Aggregate.terms

```
Aggregate.terms(name, field)
```

### 3.17 Aggregate.missing

```
Aggregate.missing(name, field)
```

### 3.18 Aggregate.max

```
Aggregate.max(name, field)
```

### 3.19 Aggregate.histogram

```
Aggregate.histogram(name, field, interval)
```

### 3.20 Aggregate.date\_range

```
Aggregate.date_range(name, field, [ranges])
```

### 3.21 Aggregate.cardinality

```
Aggregate.cardinality(name, field)
```

### 3.22 Aggregate.geohash\_grid

```
Aggregate.geohash_grid(name, field)
```

### 3.23 Aggregate.geo\_distance

```
Aggregate.geo_distance(name, field, origin, [ranges])
```

### 3.24 Aggregate.filter

```
Aggregate.filter(name, Filter)
```

### 3.25 Aggregate.percentile\_ranks

```
Aggregate.percentile_ranks(name, field)
```

### 3.26 Aggregate.range

```
Aggregate.range(name, field, [ranges])
```

### 3.27 Aggregate.significant\_terms

```
Aggregate.significant_terms(name, field)
```

### 3.28 Aggregate.top\_hits

```
Aggregate.top_hits(name)
```

### 3.29 Aggregate.reverse\_nested

```
Aggregate.reverse_nested(name)
```

---

## Suggesters

---

Note that all Suggester calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

### 4.1 Suggester.completion

```
Suggester.completion(field, size=None)
```

### 4.2 Suggester.phrase

```
Suggester.phrase(field, gram_size=None, real_word_error_likelihood=None, confidence=None, max_errors=
```

### 4.3 Suggester.term

```
Suggester.term(field, analyzer=None, size=None, sort=None, suggest_mode=None)
```



---

## Filters

---

Note that all Filter calls can also be passed additional keyword arguments not specified here, but no validation of inputs is done on them.

### 5.1 Filter.geohash\_shell

```
Filter.geohash_shell(field, lat=None, lon=None)
```

### 5.2 Filter.geo\_polygon

```
Filter.geo_polygon(field, [points])
```

### 5.3 Filter.exists

```
Filter.exists(field)
```

### 5.4 Filter.not\_

```
Filter.not_(filter=Filter, query=Query)
```

### 5.5 Filter.nested

```
Filter.nested(path, Filter)
```

### 5.6 Filter.prefix

```
Filter.prefix(field, value)
```

## 5.7 Filter.has\_parent

```
Filter.has_parent(parent_type, filter=Filter, query=Query)
```

## 5.8 Filter.geo\_distance\_range

```
Filter.geo_distance_range(field, lat=None, lon=None)
```

## 5.9 Filter.script

```
Filter.script(script)
```

## 5.10 Filter.bool

```
Filter.bool(must=[Filter], must_not=[Filter], should=[Filter])
```

## 5.11 Filter.type

```
Filter.type(value)
```

## 5.12 Filter.terms

```
Filter.terms(field, [value])
```

## 5.13 Filter.has\_child

```
Filter.has_child(type, filter=Filter, query=Query)
```

## 5.14 Filter.missing

```
Filter.missing(field)
```

## 5.15 Filter.term

```
Filter.term(field, value)
```



## 5.16 Filter.geo\_shape

```
Filter.geo_shape(field, type=None, coordinates=[])
```

## 5.17 Filter.regexp

```
Filter.regexp(field, value, flags=None, max_determinized_states=None)
```

## 5.18 Filter.or\_

```
Filter.or_([Filter])
```

## 5.19 Filter.match\_all

```
Filter.match_all(None)
```

## 5.20 Filter.geo\_distance

```
Filter.geo_distance(field, lat=None, lon=None)
```

## 5.21 Filter.geo\_bounding\_box

```
Filter.geo_bounding_box(field, top_left=None, bottom_right=None)
```

## 5.22 Filter.and\_

```
Filter.and_([Filter])
```

## 5.23 Filter.ids

```
Filter.ids([values], type=None)
```

## 5.24 Filter.range

```
Filter.range(field, gte=None, gt=None, lte=None, lt=None)
```

## 5.25 Filter.limit

```
Filter.limit(value)
```

## 5.26 Filter.indices

```
Filter.indices([indices], filter=Filter, no_match_filter=Filter)
```

---

**Meta**

---

- `genindex`
- `modindex`



---

### Credits

---

This library is created and maintained by <http://catholabs.com/authors/> team

- Contributors: <https://github.com/catholabs/esengine/graphs/contributors>

A lot of Open Source software were used to develop and inspire this project:

- PyCharm (IDE)
- Py.Test (test suit)
- TravisCI (Continuous Integration)
- Landscape.io (Code Quality Assurance)
- Elasticsearch-py (Base Client)
- Elasticsearch-dsl (inspiration for models)
- ElasticQuery by “Nick [Barrett/@Fizzadar](#)” (inspiration for query builder)