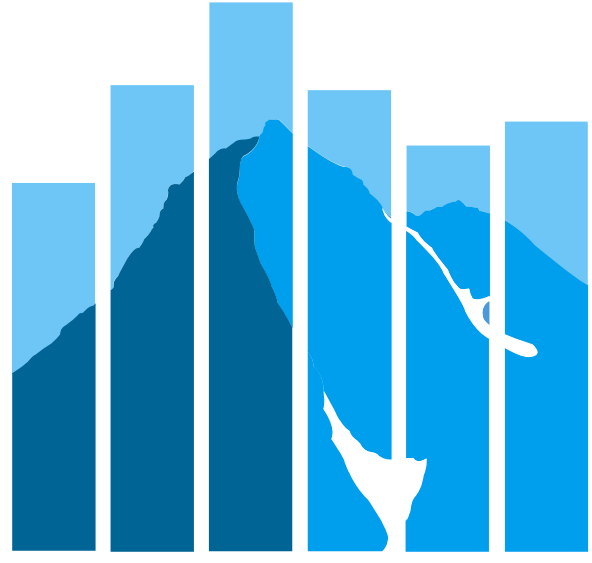

THE LOANPRO API

WORKING WITH ELASTICSEARCH (PART 1)

loanprosoftware.simnang.com

API SAMPLES

Fork Customer Search — https://plnkr.co/edit/xmemSm

Fork Loan Search — https://plnkr.co/edit/1MrwiP

⚠ CAUTIONARY NOTE ⚠

The provided API samples store API credentials in the browser

Use the API samples to explore the API and not as your integration

WHAT IS ELASTICSEARCH?

From their website:

> *Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases*

Means Elasticsearch does fast searching and data analysis

WHY ELASTICSEARCH?

LoanPro uses Elasticsearch in the backend for the following reasons:

- Fast, efficient, distributed (scalable)
- Well used, well tested, and very reliable
- Plenty of documentation

HOW DO I USE ELASTICSEARCH?

LoanPro doesn't provide direct access to the Elasticsearch databases

All access is through the LoanPro API custom query and search endpoints

- The API performs security checks before submitting to Elasticsearch
- Simplifies integration (only one API authentication token)

HOW DO I RUN SEARCHES AND QUERIES?

API endpoints that communicate with Elasticsearch have a query field

- Usually at the root level, but sometimes is nested

The query field is an Elasticsearch query object

- Most Elasticsearch query parameters are supported

SEARCHING: DEFINING THE SEARCH

First step is to define what to search

- What values do you want in specific data fields?
- What values do you NOT want in specific data fields?
- What values are absolutely required?

SEARCHING: GENERATE THE QUERY OBJECT

LoanPro uses the Elasticsearch query language

- Use **bool** to encapsulate the query
- Use **must** to encapsulate what's required
- Use **should** to encapsulate "at least one"
- Use **not** to encapsulate "I don't want this"
- Use **match** to specify an exact value match
- Use **query_string** to specify a RegEx match

BOOL OBJECT

bool is used to represent that the child object will return *true* or *false*

```
"bool":{
  "must":[
    {
      ...
    }
  ]
}
```

Either *true* or *false* will be returned

MUST OBJECT

must states that all children objects must be true

- (think of it as an AND gate)

```
"must":[
    {
      "match":{
        ...
      }
    },
    {
      "nested":{
        "match":{
          ...
        }
      }
    }
  ]
]
```

Both the entity *and* the nested entity need to match

SHOULD OBJECT

should states that at least one child object must be true

- (think of it as an OR gate)

```
"should":[
    {
      "match":{
        ...
      }
    },
    {
      "nested":{
        "match":{
          ...
        }
      }
    }
  ]
]
```

Either the entity *or* the nested entity need to match

NOT OBJECT

not states that the child must be false in order to return true

- (think of it as an NOT gate)

```
"not":{
  "must":[
    {
      "match":{
        ...
      }
    }
  ]
}
```

The entity should *not* match

MATCH

match checks that a field exactly matches the specified field

- (think of it as "==")

```
"must":[
  {
    "match":{
      "loanId":"534"
    }
  }
]
```

The entity must have a loanId of "534"

QUERY STRING

query_string does a Regex match

- Has a "query" field which is the RegEx to use
- Has a "fields" field which is an array of fields to check
- Has a "default_operator" field which determines if an "OR" or "AND" operation is used on fields
  - defaults to "OR"

```
"must":[
  {
    "query_string":{
      "query":"*John*",
      "default_operator": "OR",
      "fields":[
        "firstName",
        "email"
      ]
    }
  }
]
```

Returns entities with "John" appearing anywhere in firstName or email

MORE RESOURCES

The query language is rather complex.

For a full list of options in LoanPro, see the article API Query objects

SEARCHING: GETTING SUMMARY INFO

Summary information is returned through Aggregates

Aggregates can aggregate the data returned

- Ususally sums or averages a field

GENERATE AGGREGATES

Aggregates collect summary information about the results so you don't have to

- Can collect average
  - Prepend *avg_* to the field to average
  - Have a sub object "avg" with key "field"

List of fields ↓

AGGREGATE FIELDS

Aggregate fields differ based on the search being performed

Refer to the documentation for your specific search for a list of aggregate fields

SAMPLE AGGREGATE FIELDS

```
"avg_loansCount": {
  "avg": {
    "field": "loansCount"
  }
}
```

Gets the average loan count

SEARCHING: COMBINING EVERYTHING

- The Elasticsearch query object will go into a **query** object
- The aggregate fields will go into an **aggs** object

COMBINED SAMPLE

Below is a sample that grabs customers with "John" in the first name and averages their loan count

```
POST https://loanpro.simnang.com/api/public/api/1/Customers/Autopal.Search()
```

```
{
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "*John*",
            "fields": [
              "firstName"
            ]
          }
        }
      ]
    }
  },
  "aggs": {
    "avg_loansCount": {
      "avg": {
        "field": "loansCount"
      }
    }
  }
```

TOO MANY RESULTS!

If you have lot of data, you can easily get hundreds or thousands of results

If there are too many results, you could time out and not get anything back

Solution: Paginate!

PAGINATION

Pagination is done through a URL query

- **$top** - the number of results to return
- **$start** - the number of results to skip

```
$top=10
```

Returns the top 10 results

```
$start=25
```

Skips the first 25 results

GETTING RESULTS 1-25

```
POST loanpro.simnang.com/api/public/api/1/Customers/Autopal.Search()?$top=25&$skip=0
```

```
{
  "query": {
    "bool": {
      "must": [
        {
          ...
        }
      ]
    }
  },
  "aggs": {
    ...
  }
}
```

GETTING RESULTS 26-50

```
POST loanpro.simnang.com/api/public/api/1/Customers/Autopal.Search()?$top=25&$skip=25
```

```
{
  "query": {
    "bool": {
      "must": [
        {
          ...
        }
      ]
    }
  },
  "aggs": {
    ...
  }
}
```

QUESTIONS?

Feel free to ask in our Forums

Or, email us at support@simnang.com