
# API reference

This documentation is a reference for the REST API of the picturesafe-search Acceleration Server. The queries shown in the examples can be executed with tools like 'curl' or 'Postman'.

# Table of Contents

## 1. Authorization

The REST API uses JSON Web Tokens (JWT) for authorization. You have to provide a valid JWT in the authorization header to access the REST API resources. You can request a JWT via login resource.

*curl header parameter*

```
curl -H 'Authorization: Bearer <JWT-TOKEN>' ...
```

## 2. Login resource

The login resource provides a `JSON Web Token` (JWT) based on username and password.

*Table 1. Resource details*

| | |
|---|---|
| Resource path | /api/v1/login |
| HTTP method | POST |
| Content type | application/json |
| Response type | application/json |

*Request:*

```
{
    "username": "myusername",
    "password": "mypassword"
}
```

*Response:*

```
{
    "username": "myusername",
    "jwt":
"exJhbAciOiJIUzUxMiJ8.exJydWIiOiJhzG1sbiIsIuV5cBQ5MtU5MAE5NzQzNywifWF0IjoxNTkvMzYxNDM2g
Q.CgJDIj_JpsWcaW6AvrjTDPyjBV5QDWE1sd3nofZnZyDZSBK5gklGvttv0enMN17FMM1TVXGzjA4rbPOQgbQt_
R"
}
```

## 3. Version resource

The version resource provides version information about the picturesafe-search Acceleration Server REST API.

*Table 2. Resource details*

| | |
|---|---|
| Resource path | /api/v1/version |
| HTTP method | GET |
| Response type | application/json |
| Authorization | not required |

*Request:*

```
curl -X GET -H 'Authorization: Bearer <JWT-TOKEN>' 'http://<SERVER-
ADDRESS>/api/v1/version'
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 0,
    "serverVersion": "{server-version}"
}
```

# 4. Index resource

The index resource provides functions for Elasticsearch index operations, such as creating and deleting indices, updating the index configuration, or adding and deleting documents.

## 4.1. List indices

Gets a list of all Elasticsearch indices managed by the picturesafe-search Acceleration Server. Protected indices (such as system indices) are not returned by default.

*Table 3. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index |
| HTTP method | GET |
| Parameter type | URL parameter |
| Response type | application/json |

*Table 4. Request parameter*

| Name | Description | Required | Default |
|---|---|---|---|
| protected | Include protected aliases? | no | false |

*Request:*

```
GET /api/v1/index?protected=false
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 7,
    "indices": [
        {
            "indexAlias": "enterprise-app-test",
            "fieldCount": 0,
            "hasId": true,
            "hasDefaultFulltext": true,
            "hasDefaultSuggest": true,
            "shards": 1,
            "replicas": 0
        }
    ]
}
```

*Table 5. Response fields of indices*

| Name | Description |
|---|---|
| indexAlias | Index alias |
| fieldCount | Number of fields in the index |
| hasId | Does the index has a predefined ID field? |
| hasDefaultFulltext | Does the index has a predefined fulltext field? |
| hasDefaultSuggest | Does the index has a predefined suggest field? |
| shards | Number of shards |
| replicas | Number of replicas |

*Table 6. Http status codes*

| | |
|---|---|
| 200 | OK |
| 500 | Internal server error |

## 4.2. Get index settings

Gets the settings of the specified index alias.

*Table 7. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias] |
| HTTP method | GET |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Response type | application/json |

*Request:*

```
GET /api/v1/index/enterprise-app-test
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 6,
    "indexAlias": "enterprise-app-test",
    "settings": {
        "shards": 1,
        "replicas": 0,
        "maxResultWindow": 500000,
        "fieldsLimit": null,
        "hasId": false,
        "hasDefaultFulltext": false,
        "hasDefaultSuggest": false,
        "locales": [
            "de",
            "en"
        ],
        "fieldConfigurations": null
    }
}
```

A description of the settings can be found under "Create index with settings".

*Table 8. Http status codes*

| | |
|---|---|
| 200 | OK |
| 404 | Index not found |
| 500 | Internal server error |

## 4.3. Get index state

Gets the state of the specified index alias.

*Table 9. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/state |
| HTTP method | GET |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Response type | application/json |

The following example shows the state of a normal index. The response fields `progress`,

`documentsProcessed` and `totalDocuments` only have values if the index is currently being rebuilt at the time of the state request.

*Request:*

```
GET /api/v1/index/enterprise-app-test/state
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 12,
    "indexAlias": "enterprise-app-test",
    "state": "READY",
    "progress": null,
    "documentsProcessed": null,
    "totalDocuments": null
}
```

*Table 10. Index states*

| Index state | Description |
|---|---|
| READY | The index is available for queries and index operations |
| IN_PROGRESS | The index is currently being created or rebuilt |
| DELETED | The index was deleted |
| ERROR | The index is incorrect |

*Table 11. Http status codes*

| | |
|---|---|
| 200 | OK |
| 404 | Index not found |
| 500 | Internal server error |

## 4.4. Create index

Creates an Elasticsearch index with or without settings and/or data.

### 4.4.1. Create index without settings

Creates an Elasticsearch index without defined settings.

In this case the index is created with default index settings and without defined field mapping.

*Table 12. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/create |
| HTTP method | POST |

| | |
|---|---|
| Parameter type | Path parameter |
| Parameter | Index alias |
| Response type | application/json |

*Request:*

```
POST /api/v1/index/enterprise-app-test/create
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 229,
    "indexAlias": "enterprise-app-test",
    "state": "IN_PROGRESS",
    "progress": "CREATE_INDEX",
    "documentsProcessed": null,
    "totalDocuments": null
}
```

### 4.4.2. Create index with settings

Creates an Elasticsearch index with defined settings.

*Table 13. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/create |
| HTTP method | POST |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Content type | application/json |
| Response type | application/json |

*Table 14. Request parameters (index settings)*

| Name | Description | Required | Default |
|---|---|---|---|
| shards | Number of shards | yes | |
| replicas | Number of replicas | yes | |
| maxResultWindow | Maximum result window size (maximum number of hits in a search result) | | 500000 |
| fieldsLimit | maximum number of fields in the index | | Elastcisearch default, see Elasticsearch mapping |

| Name | Description | Required | Default |
|---|---|---|---|
| hasId | Add field `id` of Elasticsearch type `TEXT` to index mapping | | false |
| hasDefaultFulltext | Add field `fulltext` of Elasticsearch type `TEXT` to index mapping. Required for fulltext searches. | | false |
| hasDefaultSuggest | Add field `suggest` of Elasticsearch type `COMPLETION` to index mapping. Required for suggest API calls. | | false |
| locales | Supported locales for multilingual fields | | |
| fieldConfigurations | Definition of fields to be stored in the Elasticsearch index | | |

*Table 15. Field configuration parameters*

| Name | Description | Required | Default |
|---|---|---|---|
| name | Field name | yes | |
| elasticType | Elasticsearch type of field in Elasticsearch mapping. See allowed Elasticsearch types below. | yes | |
| sortable | Should the field be sortable? | | false |
| aggregatable | Should the field be aggregatable? | | false |
| multilingual | Should the field store values in different languages? Required for language dependent searches. | | false |
| copyTo | Copy field value to one or more group field. Add `fulltext` if field content should be found via fulltext search, add `suggest` if field content should be suggestable. | | |
| analyzer | Name of Elasticsearch analyzer for this field | | |
| nestedFields | Definition of nested fields. If nested fields are defined, the `elasticType` must be `NESTED` (see request sample below). | | |

*Table 16. Allowed Elasticsearch types*

| Name | Description |
|---|---|
| TEXT | Text content, analyzed by default |
| KEYWORD | Text content, **not analyzed** (can only be found when searching the whole text) |
| LONG | Number in long format (int64) |
| INTEGER | Number in integer format (int32) |
| SHORT | Number in short format (int16) |

| Name | Description |
|---|---|
| BYTE | Number in byte format (int8) |
| DOUBLE | Number in double format (floating-point number with double precision) |
| FLOAT | Number in float format (floating-point number) |
| DATE | Date (format: ISO-8601) |
| BOOLEAN | Boolean |
| NESTED | Nested object (stored as sub-documents, arrays of objects can be searched independently) |
| OBJECT | Object (stored flattened, arrays of objects can only be searched as a group, needs less resources than nested objects) |
| COMPLETION | Data for completion suggester (needed for suggest functionality) |

*Request:*

```
{
  "shards": 2,
  "replicas": 0,
  "maxResultWindow": 123321,
  "fieldsLimit": 1221,
  "hasId": true,
  "hasDefaultFulltext": true,
  "hasDefaultSuggest": true,
  "locales": ["de", "en", "fr", "es"],
  "fieldConfigurations": [
    {
      "name": "title",
      "elasticType": "TEXT",
      "sortable": true,
      "copyTo": ["fulltext", "suggest"]
    },
    {
      "name": "description",
      "elasticType": "TEXT",
      "multilingual": true,
      "copyTo": ["fulltext"]
    },
    {
      "name": "category",
      "elasticType": "KEYWORD"
    },
    {
      "name": "ebook",
      "elasticType": "BOOLEAN",
      "sortable": true,
      "aggregatable": true
    },
    {
      "name": "publicationDate",
      "elasticType": "DATE",
      "sortable": true,
      "aggregatable": true
    },
    {
      "name": "pageCount",
      "elasticType": "INTEGER",
      "sortable": true,
      "aggregatable": true
    },
    {
      "name": "author",
```

```
      "elasticType": "NESTED",
      "nestedFields": [
        {
          "name": "name",
          "elasticType": "TEXT",
          "sortable": true,
          "aggregatable": true,
          "copyTo": ["fulltext", "suggest"]
        },
        {
          "name": "email",
          "elasticType": "TEXT",
          "copyTo": ["fulltext"],
          "analyzer": "simple"
        }
      ]
    }
  ]
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 229,
    "indexAlias": "enterprise-app-test",
    "state": "IN_PROGRESS",
    "progress": "CREATE_INDEX",
    "documentsProcessed": null,
    "totalDocuments": null
}
```

*Table 17. Http status codes*

| | |
|---|---|
| 200 | OK |
| 400 | Bad request |
| 500 | Internal server error |

*Table 18. Response fields*

| Name | Description |
|---|---|
| state | State of index rebuild |
| progress | Progress of index rebuild |
| documentsProcessed | Number of documents processed (so far) |
| totalDocuments | Number of total documents to be processed |

*Table 19. Index states*

| Index state | Description |
|---|---|
| READY | The index is available for queries and index operations |
| IN_PROGRESS | The index is currently being created or rebuilt |
| DELETED | The index was deleted |
| ERROR | The index is incorrect |

*Table 20. Progress states*

| Index state | Description |
|---|---|
| PREPARE | Preparing index rebuild |
| CREATE_INDEX | New index is currently be created |
| ADD_DOCUMENTS | Documents are currently added to index |
| SET_ALIAS | New index alias is being set |
| PROCESS_DELTA | Processing document delta |
| DELETE_OLD_INDEX | Old index is currently deleted |
| END | Rebuild finished |
| ERROR | An error occurred |

### 4.4.3. Create index with data

Creates an Elasticsearch index and fills it with the provided data. Defined settings may also be provided optionally.

*Table 21. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/create |
| HTTP method | PUT |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Content type | multipart/form-data |
| Response type | application/json |

*Table 22. Request parameters*

| Name | Description | Content type | Required |
|---|---|---|---|
| dataType | Upload data type | application/json | yes |
| data | Data file to ingest | application/octet-stream | yes |

| Name | Description | Content type | Required |
|---|---|---|---|
| settings | Index settings (same as in create index with settings request) | application/json | |

*Table 23. Upload data types*

| Name | Description |
|---|---|
| CSV | CSV file (separator `;` - table header containing the field names is required) |

*Request:*

```
PUT /api/v1/index/enterprise-app-test/create
Authorization: Bearer [jwt-token]
Content-Type: multipart/form-data; boundary=boundary

--boundary
Content-Disposition: form-data; name="settings"
Content-Type: application/json

{
  "shards": 2,
  "replicas": 0,
  "maxResultWindow": 500000,
  "fieldsLimit": 1200,
  "hasId": true,
  "hasDefaultFulltext": true,
  "hasDefaultSuggest": true,
  "locales": ["de", "en"],
  "fieldConfigurations": [

...

  ]
}
--boundary
Content-Disposition: form-data; name="dataType"
Content-Type: application/json

"CSV"
--boundary
Content-Disposition: form-data; name="data"; filename="test.csv"
Content-Type: application/octet-stream

< index_data.csv
--boundary--
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 223,
    "indexAlias": "enterprise-app-test",
    "state": "IN_PROGRESS",
    "progress": "CREATE_INDEX",
    "documentsProcessed": null,
    "totalDocuments": null
}
```

*Table 24. Http status codes*

| | |
|---|---|
| 200 | OK |
| 400 | Bad request |
| 500 | Internal server error |

*Table 25. Response fields*

| Name | Description |
|---|---|
| state | State of index rebuild (same as in create index with settings request) |
| progress | Progress of index rebuild (same as in create index with settings request) |
| documentsProcessed | Number of documents processed (so far) |
| totalDocuments | Number of total documents to be processed |

### 4.4.4. Rebuild index with data

Creates an existing Elasticsearch index and replaces its data with the provided data. Defined settings may also be provided optionally. This request can be useful to update the data and/or the settings of an existing index while keeping it searchable during the update. The server will create a "shadow index" besides the existing index and ingest the data in the background. When the new index is ready, the alias will be switched and the old index will be deleted afterwards.

The request and response parameters of the index rebuild with settings are the same as for the create index with data request.

*Table 26. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/rebuild |
| HTTP method | PUT |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Content type | multipart/form-data |
| Response type | application/json |

*Request:*

```
PUT /api/v1/index/enterprise-app-test/rebuild
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 252,
    "indexAlias": "enterprise-app-test",
    "state": "IN_PROGRESS",
    "progress": "CREATE_INDEX",
    "documentsProcessed": null,
    "totalDocuments": null
}
```

*Table 27. Http status codes*

| | |
|---|---|
| 200 | OK |
| 400 | Bad request |
| 500 | Internal server error |

## 4.5. Delete index

Deletes the index of the specified index alias and removes the alias.

*Table 28. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/state |
| HTTP method | DELETE |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Response type | application/json |

*Request:*

```
DELETE /api/v1/index/enterprise-app-test
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 440,
    "indexAlias": "enterprise-app-test",
    "state": "DELETED",
    "progress": null,
    "documentsProcessed": null,
    "totalDocuments": null
}
```

*Table 29. Http status codes*

| | |
|---|---|
| 200 | OK |
| 404 | Index not found |
| 500 | Internal server error |

## 4.6. Add documents to index

Adds documents to the given Elasticsearch index.

*Table 30. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/docs |
| HTTP method | PUT |
| Parameter type | Path parameter |
| Parameter | Index alias |
| Response type | application/json |

*Request:*

```
[
  {
    "id": "100001",
    "title": "Test #1",
    "description.de" : "Test #1 mit dem Dude",
    "description.en" : "Test #1 with the Dude"
  },
  {
    "id": "100002",
    "title": "Test #2",
    "description.de" : "Test #2 mit dem Dude",
    "description.en" : "Test #2 with the Dude"
  },
  {
    "id": "100003",
    "title": "Test #3",
    "description.de" : "Test #3 mit dem Dude",
    "description.en" : "Test #3 with the Dude"
  }
]
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 64,
    "indexAlias": "enterprise-app-test",
    "documentsProcessed": 3
}
```

*Table 31. Http status codes*

| | |
|---|---|
| 200 | OK |
| 404 | Index not found |
| 500 | Internal server error |

## 4.7. Delete documents from index

Deletes documents from the given Elasticsearch index.

*Table 32. Resource details*

| | |
|---|---|
| Resource path | /api/v1/index/[Index alias]/delete |
| HTTP method | POST |
| Parameter type | Path parameter |

| Parameter | Index alias |
|---|---|
| Response type | application/json |

*Request:*

```
["100001", "100002", "100003"]
```

*Table 33. Http status codes*

| | |
|---|---|
| 204 | No content (Delete successfull) |
| 404 | Index not found |
| 500 | Internal server error |

## 5. Search resource

The search resource can be used to run fulltext, field, or combined searches.

*Table 34. Resource details*

| | |
|---|---|
| Resource path | /api/v1/search |
| HTTP method | POST |
| Content type | application/json |
| Response type | application/json |
| Authorization | Bearer token (JWT) |

*Overview:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "language": "de",
    "sortOptions": [{
        "attribute": "id",
        "direction": "DESC"
    }],
    "aggregations": [{
        "aggregationType": "DEFAULT",
        "field": "ebook"
    }],
    "resultAttributes": [
        "id",
        "title",
        "category"
    ]
}
```

*Table 35. Request parameters*

| Name | Description | Required | Default |
|---|---|---|---|
| context | Search context | yes | |
| query | query term(s), see query examples below | yes | |

| Name | Description | Required | Default |
|---|---|---|---|
| maxResuls | Maximum number of results | | 10000 |
| pageIndex | Pagination page index (starts with 1) | | 1 |
| pageSize | Pagination page size | | 100 |
| language | Language for searching multilingual content. Format: ISO-639-1 language code, e.g. 'de' | | all languages |
| sortOptions | Array of result sort options (in descending priority) | | unsorted |
| aggregations | Array of aggregations of type `DEFAULT`, `TERMS`, `DATE_HISTOGRAM` and `DATE_RANGE` | | no aggregations |
| resultAttributes | Names of attributes to be returned with the query result | | all attributes (field names) |

*Table 36. Search context parameters*

| Name | Description | Required |
|---|---|---|
| searchType | Search type, currently only type `INDEX` supported | yes |
| indexAlias | Name of index alias | yes |

*Table 37. Aggregation parameters of type DEFAULT*

| Name | Description | Required |
|---|---|---|
| aggregationType | Type of aggregation: `DEFAULT` | yes |
| field | Field to be aggregated | yes |

*Table 38. Aggregation parameters of type TERMS*

| Name | Description | Required | Default |
|---|---|---|---|
| aggregationType | Type of aggregation: `TERMS` | yes | |
| field | Field to be aggregated | yes | |
| maxCount | Maximum count of buckets | | 10 |
| minDocCount | Minimum count of documents for buckets. Buckets containing less than the minimum count will be omitted. | | 1 |
| order | `COUNT`, `KEY_ASC`, `KEY_DESC` | | `COUNT` |

*Table 39. Aggregation parameters of type DATE_HISTOGRAM*

| Name | Description | Required | Default |
|---|---|---|---|
| aggregationType | Type of aggregation: `DATE_HISTOGRAM` | yes | |

| Name | Description | Required | Default |
|---|---|---|---|
| field | Field to be aggregated | yes | |
| name | Name of the aggregation, for example 'month-overview' | | Field name |
| intervalType | Type of interval: `CALENDAR` or `FIXED` | | `CALENDAR` |
| interval | Interval for example `month` or `last month` | | 1y |
| format | Format for the bucket key, for example `yyyy-MM`. If `format` is not specified, the first date format specified in the Elasticsearch field mapping is used. | | |
| minDocCount | Minimum count of documents for buckets. Buckets containing less than the minimum count will be omitted. | | 1 |
| order | Order of the returned buckets: `COUNT`, `KEY_ASC`, `KEY_DESC` | | `COUNT` |

*Table 40. Aggregation parameters of type DATE_RANGE*

| Name | Description | Required | Default |
|---|---|---|---|
| aggregationType | Type of aggregation: `DATE_RANGE` | yes | |
| field | Field to be aggregated | yes | |
| name | Name of the aggregation, for example 'week-overview' | | Field name |
| format | Format for the bucket key, for example `yyyy-MM-dd`. If `format` is not specified, the first date format specified in the Elasticsearch field mapping is used. | | |
| ranges | Date ranges for aggregations | | Default ranges |

*Table 41. Ranges parameters of DATE_RANGE aggregations*

| Name | Description | Required |
|---|---|---|
| from | From date (Elasticsearch date math format, for example `now/M`). Either `from` or `to` must be set. | (yes) |
| to | To date (Elasticsearch date math format, for example `now/M+1M`). Either `from` or `to` must be set. | (yes) |
| key | Key for the bucket, for example 'this month'. If `key` is not specified, the aggregation `format` is used. | |

*Table 42. Default ranges of DATE_RANGE aggregations*

| From | To | Key |
|---|---|---|
| now/d | now/d+1d | today |
| now/d-1d | now/d | yesterday |

| From | To | Key |
|---|---|---|
| now/w | now/w+1w | week |
| now/w-1w | now/w | last week |
| now/M | now/M+1M | month |
| now/M-1M | now/M | last month |

## 5.1. Fulltext search

The search refers to a fulltext field, typically containing the content of several fields.

*Table 43. Fulltext query parameters*

| Parameter | Description | Required |
|---|---|---|
| queryType | Type of query term, must be `FULLTEXT` | yes |
| value | Fulltext query string | yes |

*Request:*

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "sortOptions": [{
        "attribute": "id",
        "direction": "DESC"
    }],
    "aggregations": [{
        "aggregationType": "DEFAULT",
        "field": "ebook"
    }]
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 876,
    "resultCount": 1000,
    "totalHitCount": 1000,
    "exactHitCount": true,
    "items": [{
        "id": "ff79e2fc-b235-4a9b-a5b3-45c7f22fa5a8",
        "attributes": [{
                "name": "description.en",
                "type": "text",
                "value": "Chuck Norris programs do not accept input."
            },
            {
                "name": "pageCount",
                "type": "integer",
                "value": -531704192
            },
            {
                "name": "ebook",
```

```
                "type": "boolean",
                "value": true
            },
            {
                "name": "description.fr",
                "type": "text",
                "value": "Chuck Norris doesn't program with a keyboard. He stares the
computer down until it does what he wants."
            },
            {
                "name": "author",
                "type": "nested",
                "value": {
                    "name": "Freiherrin Loris vom Dobler",
                    "email": "chuck38ib@gmail.com"
                }
            },
            {
                "name": "description.de",
                "type": "text",
                "value": "Chuck Norris isst keinen Honig. Chuck Norris kaut Bienen."
            },
            {
                "name": "description.es",
                "type": "text",
                "value": "Chuck Norris doesn't get compiler errors, the language
changes itself to accommodate Chuck Norris."
            },
            {
                "name": "id",
                "type": "text",
                "value": "ff79e2fc-b235-4a9b-a5b3-45c7f22fa5a8"
            },
            {
                "name": "title",
                "type": "text",
                "value": "Am Anfang war das Nichts. Dann roundhousekickte Chuck Norris
dieses Nichts und sagte: \"Such' Dir einen Job!\" Das ist die Geschichte der Entstehung
des Universums."
            },
            {
                "name": "category",
                "type": "keyword",
                "value": "Fenchel"
            },
            {
                "name": "publicationDate",
                "type": "date",
                "value": "2020-04-25T18:55:25.494+02:00"
```

```
            }
        ],
...
    }],
    "facets": [{
        "name": "ebook",
        "facets": [{
                "facetType": "VALUE",
                "attribute": "ebook",
                "label": "false",
                "type": "boolean",
                "value": "false",
                "count": 520,
                "from": null,
                "to": null,
                "children": null
            },
            {
                "facetType": "VALUE",
                "attribute": "ebook",
                "label": "true",
                "type": "boolean",
                "value": "true",
                "count": 480,
                "from": null,
                "to": null,
                "children": null
            }
        ]
    }]
}
```


### 5.1.1. Fulltext search with defined result attributes

The request parameter `resultAttributes` can be used to specify attributes which will be included in the response. If the request parameter `resultAttributes` does not exist or is empty, all attributes are returned.

*Request:*

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10,
    "sortOptions": [{
        "attribute": "id",
        "direction": "DESC"
    }],
    "aggregations": [{
        "aggregationType": "DEFAULT",
        "field": "ebook"
    }],
    "resultAttributes": [
        "id",
        "title",
        "category"
    ]
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 46,
    "resultCount": 1000,
    "totalHitCount": 1000,
    "exactHitCount": true,
    "items": [{
        "id": "ff79e2fc-b235-4a9b-a5b3-45c7f22fa5a8",
        "attributes": [{
                "name": "id",
                "type": "text",
                "value": "ff79e2fc-b235-4a9b-a5b3-45c7f22fa5a8"
            },
            {
                "name": "title",
```

```
                "type": "text",
                "value": "Am Anfang war das Nichts. Dann roundhousekickte Chuck Norris
dieses Nichts und sagte: \"Such' Dir einen Job!\" Das ist die Geschichte der Entstehung
des Universums."
            },
            {
                "name": "category",
                "type": "keyword",
                "value": "Fenchel"
            }
        ],
...
    }],
    "facets": [{
        "name": "ebook",
        "facets": [{
                "facetType": "VALUE",
                "attribute": "ebook",
                "label": "false",
                "type": "boolean",
                "value": "false",
                "count": 520,
                "from": null,
                "to": null,
                "children": null
            },
            {
                "facetType": "VALUE",
                "attribute": "ebook",
                "label": "true",
                "type": "boolean",
                "value": "true",
                "count": 480,
                "from": null,
                "to": null,
                "children": null
            }
        ]
    }]
}
```


## 5.2. Field search

Field searches can be used to address specific fields with a variety of comparison operators.

*Table 44. Field query parameters*

| Attribute | Description | Required | Default |
|---|---|---|---|
| queryType | Type of query term, must be `FIELD` | yes | |

| Attribute | Description | Required | Default |
|---|---|---|---|
| name | Field name | yes | |
| comparator | Comparison operator | | EQ |
| value | Query value | yes | |
| keyword | Handle query as keyword query | | false |

The attribute `keyword=true` can be useful for values containing separators (e.g. blanks, hyphens, etc.): The query string "Schleswig-Holstein" for example would also match on values "Schleswig" or "Holstein" when the parameter `keyword=true` is not passed.

*Table 45. Supported comparison operators*

| Operator | Comparision | Description/Notes |
|---|---|---|
| `EQ` | equals | token based equality |
| `NOT_EQ` | not equals | negated token based equality |
| `LIKE` | like | token based wildcard query - wildcards ? for single character and * for multiple characters, e.g (`*n?ce* sample*`) |
| `NOT_LIKE` | not like | negated token based wildcard query - wildcards ? for single character and * for multiple characters, e.g (`*n?ce* sample*`) |
| `GT` | greater than | mathematical > (does not work on text fields) |
| `GE` | greater equals | mathematical >= (does not work on text fields) |
| `LT` | lower than | mathematical < (does not work on text fields) |
| `LE` | lower equals | mathematical <= (does not work on text fields) |
| `TERM_STARTS_WITH` | term starts with | term based query (only works on keyword fields, fields must be sortable or aggregatable) |
| `TERM_ENDS_WITH` | term ends with | term based query (only works on keyword fields, fields must be sortable or aggregatable) |
| `TERM_WILDCARD` | term wildcard | term based wildcard query (only works on keyword fields, fields must be sortable or aggregatable) - wildcards ? for single character and * for multiple characters, e.g (`*n?ce* sample*`) |
| `IN` | in | Value must be an array. |
| `NOT_IN` | not in | Value must be an array. |
| `IS_EMPTY` | is empty | Value will be ignored. |
| `IS_NOT_EMPTY` | is not empty | Value will be ignored. |

When searching with the operators `IS_EMPTY`/`IS_NOT_EMPTY`, the `value` parameter is ignored and therefore does not have to be set.

When searching with the operators `IN/NOT_IN` or `IS_EMPTY/IS_NOT_EMPTY`, the `keyword` parameter is ignored and therefore does not have to be set.

### 5.2.1. Field search with EQ

*Field search with EQ operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "EQ",
        "value": "Tamarinde"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

*Field search with NOT_EQ operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "NOT_EQ",
        "value": "Tamarinde"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

### 5.2.2. Field search with LIKE

*Field search with LIKE operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "LIKE",
        "value": "*mar?n*"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

*Field search with NOT_LIKE operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "NOT_LIKE",
        "value": "*mar?n*"
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

### 5.2.3. Field search with GE

*Field search with GE operator (GT, LE and LT work in the same way):*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "pageCount",
        "comparator": "GE",
        "value": 1000
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

### 5.2.4. Field search with IN

The IN comparator can be used to search over multiple values of a field within a single query term. The value must be passed as an array.

*Field search with IN operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "IN",
        "value": ["Tamarinde", "Zimt"]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

*Field search with NOT_IN operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "NOT_IN",
        "value": ["Tamarinde", "Zimt"]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

### 5.2.5. Field search with IS_EMPTY

The IS_EMPTY comparator can be used to search for documents for which the specified field is empty.

*Field search with IS_EMPTY operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "IS_EMPTY",
        "value": ""
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

The IS_NOT_EMPTY comparator can be used to search for documents for which the specified field is NOT empty.

*Field search with IS_NOT_EMPTY operator:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FIELD",
        "name": "category",
        "comparator": "IS_NOT_EMPTY",
        "value": ""
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

## 5.3. Combined search

Combined searches (`"queryType": "COMBINED"`) can be used to combine different queries to perform more complex searches, e.g. searches over several fields or a combination of fulltext and field terms.

*Overview*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "COMBINED",
        "operator": "AND",
        "queries": [{
                "queryType": "FULLTEXT",
                "value": "chuck"
            },
            {
                "queryType": "FIELD",
                "name": "ebook",
                "comparator": "EQ",
                "value": true
            }
        ]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

Search terms can be combined with AND and OR and can also be nested. Field query terms have different parameters for a variety of search operations:

*Table 46. Combined search query parameters*

| Parameter | Description | Required |
|---|---|---|
| queryType | Type of query term, must be `COMBINED` | yes |
| operator | operator for the search terms, must be AND or OR | yes |
| queries | Array of query terms, see supported query types below | yes |

*Table 47. Supported query term types:*

| Query type | Description |
|---|---|
| `FULLTEXT` | The fulltext query refers to a fulltext field, typically containing the content of several fields. |
| `FIELD` | Field queries can be used to match values on regular fields. |
| `COMBINED` | Can be used to create nested queries. |

### 5.3.1. Combined field search

*Combined search sample with three fields terms:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "COMBINED",
        "operator": "AND",
        "queries": [{
                "queryType": "FIELD",
                "name": "title",
                "comparator": "LIKE",
                "value": "Chuck"
            },
            {
                "queryType": "FIELD",
                "name": "ebook",
                "comparator": "EQ",
                "value": true
            },
            {
                "queryType": "FIELD",
                "name": "category",
                "comparator": "IS_NOT_EMPTY",
                "value": ""
            }
        ]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

### 5.3.2. Nested combined queries

*Combined search with nested query terms:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "COMBINED",
        "operator": "AND",
        "queries": [{
                "queryType": "FIELD",
                "name": "ebook",
                "comparator": "EQ",
                "value": true
            },
            {
                "queryType": "COMBINED",
                "operator": "OR",
                "queries": [{
                        "queryType": "FIELD",
                        "name": "title",
                        "comparator": "LIKE",
                        "value": "Chuck"
                    },
                    {
                        "queryType": "FIELD",
                        "name": "description",
                        "comparator": "LIKE",
                        "value": "Chuck"
                    }
                ]
            }

        ]
    },
    "maxResults": 10000,
    "pageIndex": 1,
    "pageSize": 10
}
```

## 5.4. Pagination

For each search request the page size (`pageSize`) and the page index (`pageIndex`) must be passed. The search result contains the metadata items (`items`) of the requested page, the number of retrievable hits (`resultCount`) and the total number of hits (`totalHitCount`). The number of pages can be calculated using the page size and the number of retrievable hits.

The additional return value `exactHitCount` indicates whether the value of `totalHitCount` is

exact or not: With `true` `totalHitCount` is the exact number of hits, with `false` the number of hits was only determined up to the value of `totalHitCount`, but there are more hits.

To query and retrieve another page of a search result the previous search request must be executed again, changing the page index.

**Note:** Since the pagination always performs a new search the search result may vary when changing pages.

*Paginated search for second result page:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 2,
    "pageSize": 10,
    "sortOptions": [{
        "attribute": "id",
        "direction": "DESC"
    }]
}
```

## 5.5. Sorting

A search can be performed unsorted (empty or missing parameter `sortOptions`) or with one or more sort criteria. The order of the sort criteria in the parameter `sortOptions` defines their priority.

*Search without sort criteria:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 2,
    "pageSize": 10
}
```

*Search with two sort criteria:*

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 10000,
    "pageIndex": 2,
    "pageSize": 10,
    "sortOptions": [{
            "attribute": "category",
            "direction": "ASC"
        },
        {
            "attribute": "title",
            "direction": "DESC"
        }
    ]
}
```

*Table 48. Supported sort directions*

| Modifier | Direction |
|---|---|
| ASC | ascending |
| DESC | descending |

## 5.6. Content language

The content language of multilingual fields can be specified with the `language` parameter.

*Table 49. Request parameter*

| Name | Description | Required |
|---|---|---|
| language | Content language (Format: ISO-639-1 language code, e.g. 'de') | no |

If the `language` parameter is provided in a search request, the content of multilingual fields will only be returned in the language variant of the specified content language.

This parameter is also used to specify the language variant to be searched when searching in a multilingual field.

*Example 1. Search providing the content language:*

*Request:*

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 1,
    "pageIndex": 1,
    "pageSize": 10,
    "language": "de",
    "resultAttributes": [
        "id",
        "description"
    ]
}
```

*Response:*


```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 9,
    "resultCount": 1,
    "totalHitCount": 1000,
    "exactHitCount": true,
    "items": [{
        "id": "bc594284-4f5f-4c4d-84ed-2abf0ec91934",
        "attributes": [{
                "name": "description",
                "type": "text",
                "value": "Chuck Norris isst keinen Honig. Chuck Norris kaut Bienen."
            },
            {
                "name": "id",
                "type": "text",
                "value": "bc594284-4f5f-4c4d-84ed-2abf0ec91934"
            }
        ]
    }],
    "facets": []
}
```


If the `language` parameter is not provided in the search request, the content of multilingual fields is returned in every existing language variant (see field name suffix).

*Example 2. Search without specific language:*

*Request:*

```
POST /api/v1/search/
```

```
{
    "context": {
        "searchType": "INDEX",
        "indexAlias": "enterprise-app-test"
    },
    "query": {
        "queryType": "FULLTEXT",
        "value": "chuck"
    },
    "maxResults": 1,
    "pageIndex": 1,
    "pageSize": 10,
    "resultAttributes": [
        "id",
        "description"
    ]
}
```

*Response:*


```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 8,
    "resultCount": 1,
    "totalHitCount": 1000,
    "exactHitCount": true,
    "items": [{
        "id": "bc594284-4f5f-4c4d-84ed-2abf0ec91934",
        "attributes": [{
                "name": "description.en",
                "type": "text",
                "value": "Chuck Norris does not eat honey. Chuck Norris chews bees."
            },
            {
                "name": "description.fr",
                "type": "text",
                "value": "Chuck Norris ne mange pas de miel. Chuck Norris mâche des
abeilles."
            },
            {
                "name": "description.de",
                "type": "text",
                "value": "Chuck Norris isst keinen Honig. Chuck Norris kaut Bienen."
            },
            {
                "name": "description.es",
                "type": "text",
                "value": "Chuck Norris no come miel. Chuck Norris mastica abejas."
            },
            {
                "name": "id",
                "type": "text",
                "value": "bc594284-4f5f-4c4d-84ed-2abf0ec91934"
            }
        ]
    }],
    "facets": []
}
```

## 6. Suggest resource

The suggest resource provides completion suggestions for user input which is useful for a search-as-you-type functionality.

*Table 50. Resource details*

| | |
|---|---|
| Resource path | /api/v1/suggest/ |
| HTTP method | POST |
| Content type | application/json |
| Response type | application/json |
| Authorization | Bearer token (JWT) |

*Table 51. Request parameters*

| Name | Description | Required | Default |
|---|---|---|---|
| indexAlias | Index alias | yes | |
| text | User input text (min length = 3) | yes | |
| count | Number of suggestions (max = 20) | | 10 |

*Request:*

```
POST /api/v1/suggest/
```

```
{
    "indexAlias": "enterprise-app-test",
    "text": "Chuck",
    "count": 20
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 10,
    "suggestions": [
        "Chuck Norris erfuhr einmal, dass nichts ihn besieg",
        "Chuck Norris hat bis Unendlich gezählt - zwei Mal.",
        "Chuck Norris hat einmal 37 Terroristen mit zwei Ku",
        "Chuck Norris hat seine praktische Führerscheinprüf",
        "Chuck Norris isst keinen Honig. Chuck Norris kaut ",
        "Chuck Norris ist so schnell - wenn er das Licht au",
        "Chuck Norris kann eine Bombe zerlegen - und zwar i",
        "Chuck Norris kann eine Party schmeißen. 100 Meter ",
        "Chuck Norris kann im Kinderkarussell überholen.",
        "Chuck Norris kann wirbellosen Tieren trotzdem das ",
        "Chuck Norris ließ sich an einen Lügendetektor ansc",
        "Chuck Norris lügt nicht. Die Wahrheit ist einfach ",
        "Chuck Norris wurde einmal von einer Königskobra ge",
        "Chuck Norris wurde neulich geblitzt - beim Einpark",
        "Chuck Norris zündet ein Feuer an, indem er zwei Ei",
        "Chuck Norris übernimmt die Projektleitung des Flug",
        "Chuck Norris' Passwort? Die Zahl Pi."
    ]
}
```

# 7. Stored search resource

The stored search resource provides functions for storing and loading search objects.

Search objects are stored and retrieved in a way they can be directly used for a new search request.

## 7.1. Store search

Stores a search object with the given ID.

*Table 52. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch/[*ID of stored search*] |
| HTTP method | PUT |
| Parameter | ID of stored search |
| Parameter type | Path parameter |
| Content type | application/json |
| Response type | application/json |

*Table 53. Request parameters*

| Name | Description | Required |
|---|---|---|
| userId | ID of the user storing the search | |
| name | Name of stored search. If the parameter is missing or empty, the name is created automatically. | |
| searchObject | Search object to be stored | yes |

*Request:*

```
POST /api/v1/storedsearch/enterprise-app-test
```

```
{
    "userId": "testUser",
    "name": "A stored search",
    "searchObject": {
        "context": {
            "searchType": "INDEX",
            "indexAlias": "enterprise-app-test"
        },
        "query": {
            "queryType": "FULLTEXT",
            "value": "cuck"
        },
        "maxResults": 10000,
        "pageIndex": 1,
        "pageSize": 10,
        "sortOptions": [{
            "attribute": "id",
            "direction": "DESC"
        }]
    }
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 159,
    "id": "1",
    "userId": "testUser",
    "name": "A stored search"
}
```

*Table 54. Http status codes*

| | |
|---|---|
| 201 | Created |
| 404 | Not found |
| 500 | Internal server error |

## 7.2. Update stored search

Updates a stored search with the given ID.

*Table 55. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch/[*ID of stored search*] |
| HTTP method | PATCH |
| Parameter | ID of stored search |
| Parameter type | Path parameter |
| Content type | application/json |
| Response type | application/json |

*Table 56. Request parameters*

| Name | Description | Required |
|---|---|---|
| name | Name of stored search. If the parameter is missing or empty, the name is created automatically. | |
| searchObject | Search object to be updated | yes |

*Request:*

```
PATCH /api/v1/storedsearch/enterprise-app-test
```

```
{
    "name": "A stored search with new name",
    "searchObject": {
        "context": {
            "searchType": "INDEX",
            "indexAlias": "enterprise-app-test"
        },
        "query": {
            "queryType": "FULLTEXT",
            "value": "Norris"
        },
        "maxResults": 10000,
        "pageIndex": 1,
        "pageSize": 10,
        "sortOptions": [{
            "attribute": "id",
            "direction": "DESC"
        }]
    }
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 199,
    "id": "1",
    "userId": "testUser",
    "name": "A stored search with new name"
}
```

*Table 57. Http status codes*

| | |
|---|---|
| 200 | OK (Update successfull) |
| 404 | Not found |
| 500 | Internal server error |

## 7.3. Rename stored search

Renames a stored search with the given ID.

"Rename" is the same operation as "Update", but without passing a search object.

*Table 58. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch/[*ID of stored search*] |
| HTTP method | PATCH |
| Parameter | ID of stored search |
| Parameter type | Path parameter |
| Content type | application/json |
| Response type | application/json |

*Table 59. Request parameters*

| Name | Description | Required |
|---|---|---|
| name | Name of stored search. If the parameter is missing or empty, the name is created automatically. | |

*Request:*

```
PATCH /api/v1/storedsearch/enterprise-app-test
```

```
{
    "name": "A stored search with a very new name"
}
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 199,
    "id": "1",
    "userId": "testUser",
    "name": "A stored search with new name"
}
```

*Table 60. Http status codes*

| | |
|---|---|
| 200 | OK (Rename successfull) |
| 404 | Not found |
| 500 | Internal server error |

## 7.4. Delete stored search

Deletes a stored search with the given ID.

*Table 61. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch/[*ID of stored search*] |

| | |
|---|---|
| HTTP method | DELETE |
| Parameter | ID of stored search |
| Parameter type | Path parameter |
| Content type | application/json |
| Response type | Empty response |

*Request:*

```
DELETE /api/v1/storedsearch/1
```

*Table 62. Http status codes*

| | |
|---|---|
| 204 | No content (Delete successfull) |
| 404 | Not found |
| 500 | Internal server error |

## 7.5. Load a stored search by id

Loads a stored search by id.

The returned search object can directly be used for a new search request.

*Table 63. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch/[*ID of stored search*] |
| HTTP method | GET |
| Parameter | ID of stored search |
| Parameter type | Path parameter |
| Content type | application/json |
| Response type | application/json |

*Request:*

```
GET /api/v1/storedsearch/1
```

*Response:*


```
{
    "id": "1",
    "userId": "testUser",
    "name": "A stored search",
    "searchObject": {
        "context": {
            "searchType": "INDEX",
            "indexAlias": "enterprise-app-test"
        },
        "query": {
            "queryType": "FULLTEXT",
            "value": "Chuck"
        },
        "maxResults": 10000,
        "pageIndex": 1,
        "pageSize": 10,
        "sortOptions": [
            {
                "sortType": "ATTRIBUTE",
                "attribute": "id",
                "direction": "DESC"
            }
        ],
        "resultAttributes": null,
        "language": null
    }
}
```


*Table 64. Http status codes*

| | |
|---|---|
| 200 | OK |
| 404 | Not found |
| 500 | Internal server error |

## 7.6. Load list of stored searches

Loads a list of stored searches. The list entries contain the id, user id, index alias, and name of a stored search.

*Table 65. Resource details*

| | |
|---|---|
| Resource path | /api/v1/storedsearch |

| | |
|---|---|
| HTTP method | GET |
| Parameter type | URL parameter |
| Response type | application/json |

*Table 66. Request parameters*

| Name | Description | Required |
|---|---|---|
| userId | ID of the user whose stored searches are to be loaded | |
| indexAlias | Alias name under which the searches were stored | |

*Request (load list of all stored searches):*

```
GET /api/v1/storedsearch
```

*Request (load list of all stored searches of given user):*

```
GET /api/v1/storedsearch?userId=testUser
```

*Request (load list of all stored searches of given alias name):*

```
GET /api/v1/storedsearch?indexAlias=enterprise-app-test
```

*Request (load list of all stored searches of given user and given alias name):*

```
GET /api/v1/storedsearch?userId=testUser&indexAlias=enterprise-app-test
```

*Response:*

```
{
    "apiVersion": "{api-version}",
    "processingTimeMillis": 27,
    "storedSearchListEntries": [
        {
            "id": "1",
            "userId": "testUser",
            "indexAlias": "enterprise-app-test",
            "name": "A stored search"
        },
        {
            "id": "2",
            "userId": "testUser",
            "indexAlias": "enterprise-app-test",
            "name": "Another stored search"
        }
    ]
}
```

*Table 67. Http status codes*

| | |
|---|---|
| 200 | OK |
| 500 | Internal server error |