NOTICE

The information contained in this document is believed to be accurate in all respects but is not warranted by Martello Technologies Corporation. The information is subject to change without notice and should not be construed in any way as a commitment by Martello Technologies or any of its affiliates or subsidiaries. Martello Technologies and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes.

No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Martello Technologies.

Trademarks

MarWatch™, Savision, Martello Technologies, and the Martello Technologies logo are trademarks of Martello Technologies Corporation.

Windows and Microsoft are trademarks of Microsoft Corporation.

Other product names mentioned in this document may be trademarks of their respective companies and are hereby acknowledged.

© Copyright 2021, Martello Technologies Corporation
All rights reserved
# Contents

## CHAPTER 1

- Introduction ............................................................................................................. 4
- Document Purpose and Intended Audience ......................................................... 4
- Revision History ..................................................................................................... 4

## CHAPTER 2

- About the REST API ............................................................................................... 5
- Basic Components .................................................................................................. 5
  - Elasticsearch Node .............................................................................................. 5
  - Elasticsearch Index .......................................................................................... 5
  - Elasticsearch Type ......................................................................................... 6
  - Elasticsearch Document ................................................................................. 6
  - Source ............................................................................................................. 6
  - Savision iQ Object Types ................................................................................ 6
    - Component .................................................................................................... 7
  - Component Relationships ............................................................................... 11
    - Hosting Relationship ................................................................................ 12
    - Containment Relationship .......................................................................... 12
    - Reference Relationship .............................................................................. 13

## CHAPTER 3

- Configuring an API Source ................................................................................... 15

## CHAPTER 4

- Using the REST API ............................................................................................... 17
  - Create and Update a Document ..................................................................... 17
  - Retrieve a Document ...................................................................................... 19
  - Delete a Document ......................................................................................... 20
  - Batch Processing ............................................................................................. 21
  - Search Documents ......................................................................................... 22
  - List all Indexes .............................................................................................. 23
  - Delete an Index .............................................................................................. 23
CHAPTER 1

Introduction

Document Purpose and Intended Audience
This document contains information about configuring and using the Savision iQ Rest API.

Revision History

<table>
<thead>
<tr>
<th>Document Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 25, 2021</td>
<td>Savision iQ Rest API Guide</td>
</tr>
</tbody>
</table>
About the REST API

The Savision iQ REST API allows an external application to control the information stored in the Savision iQ Elasticsearch database. Use the information in the following sections to understand the core concepts of the Savision iQ REST API.

- "Basic Components" on page 5
- "Savision iQ Object Types" on page 6
- "Component Relationships" on page 11

Basic Components

There are a few concepts that are core to Savision iQ and Elasticsearch. You must understand these concepts to successfully use the Savision iQ REST API.

Elasticsearch Node

By default, Savision iQ installs a single Elasticsearch node co-located with the Savision iQ application in the same server. This node is responsible for storing the data that originates from different Sources, and provides indexing and search capabilities.

Elasticsearch Index

An index is a collection of documents that have similar characteristics. In Savision iQ, there are specific indices for Components and States, Component Relationships, Alerts, and Incidents. An index is identified by a name (that must be all lowercase) and this name is used to refer to the index when performing indexing, search, update, and delete operations against the documents in it. The following index categories are defined in Savision iQ:

- `savisioniq_components_<SourceGuid>`—Each of these indices stores documents of type Component and Component State related to a specific Source.
- `savisioniq_component_relationships_<SourceGuid>`—Each of these indices stores documents of type Component Relationship related to a specific Source.
- `savisioniq_alerts_<SourceGuid>`—Each of these indices stores documents of type Alert related to a specific Source.
- `savisioniq_incidents_<SourceGuid>`—Each of these indices stores documents of type Incident related to a specific Source.

**Elasticsearch Type**

Within an index, only one single Elasticsearch type is defined by Savision iQ. A type is defined for documents that have a set of common fields. The index category `savisioniq_components_*` contains the Elasticsearch type “esentity,” which may represent two different Savision iQ types: the parent type Component and the child type Component State. The following Elasticsearch types are defined in Savision iQ:

- `esentity`—This type is defined in the indices `savisioniq_components_*` and may represent the parent type Component and the child type Component State.
- `componentrelationship`—This type is defined in the indices `savisioniq_component_relationships_*`.
- `alert`—This type is defined in the indices `savisioniq_alerts_*`.
- `incident`—This type is defined in the indices `savisioniq_incidents_*`.

**Elasticsearch Document**

A document is a basic unit of information that can be indexed. For example, in Savision iQ you can have a document for a single Alert, another document for a single Component, and another for a single Incident. This document is expressed in JSON (JavaScript Object Notation), which is a ubiquitous internet data interchange format. Within an index/type, you can store as many documents as you want.

**Note:** Although a document physically resides in an index, a document actually must be indexed/assigned to a type inside an index.

**Source**

In Savision iQ a Source (also called Integration in the interface) represents a specific Monitoring or ITSM system—such as SCOM or ServiceNow—that provides the data to be stored into Elasticsearch. Each Source is uniquely identified by a GUID.

**Savision iQ Object Types**

The Savision iQ REST API has five different types of objects:

- Component
- Component State
- Component Relationship
- Alert
- Incident
You must provide a valid object of a specific type to correctly create and update a
document in Elasticsearch using the Savision iQ REST API.

Component
A Component is defined as it follows:

```json
entity {
    joinKey (join)
    key (text)
    sourceId (text)
    sourceName (text)
    sourceType (text)
    source (object)
    name (text)
    typeEnum (integer)
    host (text)
    path (text)
    url (text)
    ipAddress (text)
    fqdn (text)
    geoLocation (geo_point)
}
```

For additional information about the Elasticsearch datatypes see:

The following table lists the fields in the Component document:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>joinKey</td>
<td>This is a special field that creates a parent/child relation between one Component (the parent) and the associated Component States (the children). For a Component it must be always set to “parent”.</td>
</tr>
<tr>
<td>key</td>
<td>The unique identifier of the Component document. A key is composed of two parts separated by a pipe character</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sourceld</td>
<td>The unique identifier of the Source related to the Component. A sourceld is the string representation of a globally unique identifier (GUID).</td>
</tr>
<tr>
<td>sourceName</td>
<td>The name of the Source related to the Component.</td>
</tr>
<tr>
<td>sourceType</td>
<td>The type of the Source related to the Component, for example “SCOM” or “ServiceNow”. For a Savision iQ API Source the type must always be “VirtualConnector”.</td>
</tr>
<tr>
<td>source</td>
<td>An object that contains the raw properties of the Component specific to the Source.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the Component object.</td>
</tr>
<tr>
<td>typeEnum</td>
<td>An integer that represents the type of the Component. For a list of possible Component types and their values, please see below.</td>
</tr>
<tr>
<td>host</td>
<td>The name of the Component that hosts the Component.</td>
</tr>
<tr>
<td>path</td>
<td>The path that identifies Source related to the Component.</td>
</tr>
<tr>
<td>url</td>
<td>The URL that locates the Component in the Monitoring/ITSM system. You can use this URL to navigate to the page of the Monitoring/ITSM system that contains specific information about this Component.</td>
</tr>
<tr>
<td>iPAddress</td>
<td>The IP address(es) associated with the Component.</td>
</tr>
<tr>
<td>fqdn</td>
<td>The fully qualified domain name associated with the Component.</td>
</tr>
<tr>
<td>geoLocation</td>
<td>The geolocation of the Component.</td>
</tr>
</tbody>
</table>

For information about how to specify a geoLocation field, see [https://www.elastic.co/guide/en/elasticsearch/reference/6.8/geo-point.html](https://www.elastic.co/guide/en/elasticsearch/reference/6.8/geo-point.html)

The possible Component types are:

- 1: Object
- 2: Group
- 3: Service
- 4: Computer
- 5: Database
- 6: Website
- 7: Virtual Machine
An example of a SCOM Component in JSON format is the following:
{
    "joinKey": "parent",
    "name": "TestDB1",
    "typeEnum": 5,
    "path": "BRSRV2012R2-2.savisionlab.Savision.int;MSSQLSERVER",
    "url": "/ManagedEntity/ManagedEntity/9a84ffd3-5b42-85e7-c242-9264cd6a62e9",
    "fqdn": "BRSRV2012R2-2.savisionlab.Savision.int",
    "key": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab|9a84ffd3-5b42-85e7-c242-9264cd6a62e9",
    "sourceId": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab",
    "sourceName": "Unity iQ SCOM 01",
    "sourceType": "SCOM",
    "source": {
        "scom": {
            "Object DisplayName": "TestDB1",
            "FullName": "Microsoft.SQLServer.2014.Database:BRSRV2012R2-2.savisionlab.Savision.int;MSSQLSERVER;TestDB1",
            "Id": "9a84ffd3-5b42-85e7-c242-9264cd6a62e9",
            "ManagementGroupName": "savisonUnityiQ",
            "Path": "BRSRV2012R2-2.savisionlab.Savision.int;MSSQLSERVER",
            "TimeAdded": "2017-02-07T18:04:24.4991017",
            "Database Name": "TestDB1",
            "Recovery Model": "",
            "Database Autogrow Set": "",
            "Log Autogrow Set": "",
            "Updateability": "",
            "User Access": "",
            "Collation": "",
            "Owner": "",
            "Resource Pool": "",
            "Asset Status": "",
            "Notes": "",
            "Display Name": "TestDB1",
            "Instance Name": "MSSQLSERVER",
            "Principal Name": "BRSRV2012R2-2.savisionlab.Savision.int"
        }
    }
}
In order to create, update, delete, and retrieve a Component or a batch of Components, you need to specify the routing parameter in the request, and the routing parameter must equal the key property of the component. See the section "Create and Update a Document" on page 17 for an example of a request.

## Component Relationships

A Component Relationship is defined as it follows:

```json
componentrelationship {
    key (text)
    sourceId (text)
    sourceName (text)
    sourceType (text)
    source (object)
    name (text)
    sourceComponent (text)
    destinationComponent (text)
    typeEnum (integer)
}
```

The following table lists the fields in the Component Relationships document:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The unique identifier of the Component Relationship document. A key is composed of two parts separated by a pipe character</td>
</tr>
<tr>
<td></td>
<td>. The first part is the ID of the Source of the Component Relationship; the second part is a string that uniquely identifies the Relationship among all the Relationships of the Source.</td>
</tr>
<tr>
<td>sourceId</td>
<td>The unique identifier of the Source related to the Component Relationship. A sourceId is the string representation of a globally unique identifier (GUID).</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sourceName</td>
<td>The name of the Source related to the Component Relationship.</td>
</tr>
<tr>
<td>sourceType</td>
<td>The type of the Source related to the Component Relationship, for example “SCOM” or “ServiceNow”. For a Savision iQ API Source the type must always be “VirtualConnector”.</td>
</tr>
<tr>
<td>source</td>
<td>An object that contains the raw properties of the Component Relationship specific to the Source.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the Component Relationship object.</td>
</tr>
<tr>
<td>sourceComponent</td>
<td>The key (unique identifier) of the Component that represents the “source” in the relationship.</td>
</tr>
<tr>
<td>destinationComponent</td>
<td>The key (unique identifier) of the Component that represents the “destination” in the relationship.</td>
</tr>
<tr>
<td>typeEnum</td>
<td>An integer that represents the type of the Component Relationship. For a list of possible Component Relationship types and their values, please see below.</td>
</tr>
</tbody>
</table>

The possible Component Relationship types are:
- 1: Hosting
- 2: Containment
- 3: Reference

Hosting Relationship
The most restrictive relationship between Components is the hosting relationship. When one Component is hosted by another, that Component relies on its hosting parent for its very existence. If the hosting parent is removed, the hosted child will also be removed. For example, a logical disk cannot exist without the computer that it is installed on. A hosted Component can have only one hosting parent, but one parent can host multiple children. For example, a particular disk can be installed on only a single computer, but one computer can have several disks installed.

Containment Relationship
The containment relationship type is less restrictive than the hosting relationship. It declares that one Component is related to another Component, although one is not
required for the other. Unlike a hosting relationship, a containment relationship is many-to-many. This means that one Component can contain multiple Components, and a single Component can be contained by multiple other Components. For example, one group can contain multiple components, and a single component can be a member of multiple groups.

Reference Relationship

The reference relationship is the most general relationship type. A reference relationship is used when the source and destination components are not dependent on one another; for example, a database can reference another database that it is replicating. One database is not dependent on the other, and the Components exist separately.

An example of a SCOM Component Relationship in JSON format is the following:
```json
{
  "name": "BRSRV2012R2-2.savisionlab.Savision.int\MSSQLSERVER\Hosts BRSRV2012R2-2.savisionlab.Savision.int\MSSQLSERVER\TestDB1",
  "sourceComponent": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab|da967a6a-0ac8-57fc-287e-2b21f818ca13",
  "destinationComponent": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab|9a84fffd3-5b42-85e7-c242-9264cd6a62e9",
  "typeEnum": 1,
  "type": "Hosting",
  "key": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab|d6a9160b-785c-388f-d09d-32c5a4861ea4",
  "sourceId": "12b3b4b4-ec8b-4d03-8b64-19f90c47c1ab",
  "sourceName": "Unity iQ SCOM 01",
  "sourceType": "SCOM",
  "source": {
    "scom": {
      "Object Display Name": "MSSQLSERVER - TestDB1",
      "TypeName": "MSSQL 2014: SQL Server 2014 Database Engine Hosts SQL Database",
      "TypeDescription": null,
      "SourceDisplayName": "MSSQLSERVER",
      "SourceFullName": "Microsoft.SQLServer.2014.DBEngine:BRSRV2012R2-2.savisionlab.Savision.int\MSSQLSERVER",
      "SourceId": "da967a6a-0ac8-57fc-287e-2b21f818ca13",
      "SourcePath": "BRSRV2012R2-2.savisionlab.Savision.int",
      "TargetDisplayName": "TestDB1",
      "TargetFullName": "Microsoft.SQLServer.2014.Database:BRSRV2012R2-2.savisionlab.Savision.int;MSSQLSERVER;TestDB1",
      "TargetId": "9a84fffd3-5b42-85e7-c242-9264cd6a62e9",
      "TargetPath": "BRSRV2012R2-2.savisionlab.Savision.int;MSSQLSERVER",
      "ManagementGroupName": "savisonUnityiQ",
      "LastModified": "2017-02-07T18:04:24.673"
    }
  }
}
```
Configuring an API Source

Before starting to use the Savision iQ REST API, you need to configure an Open API source in Savision iQ. Use the following procedure to configure the source.

1. Connect to Savision iQ using a browser.
2. From the main menu, select Settings.
   The Integrations page displays.
3. Click the Add button at the bottom of the page.
4. Select Savision API.
5. Enter the following information and click Save:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Select a server to communicate with the source system. This can be the Savision iQ web server or a machine that has a Savision iQ Remote agent installed on it.</td>
</tr>
<tr>
<td>Name</td>
<td>Provide a name for the integration; this name displays on the Savision iQ interface.</td>
</tr>
<tr>
<td>Discovery Interval</td>
<td>How often the objects are loaded from the integrated system. The default is 3600 seconds.</td>
</tr>
<tr>
<td>Operation Interval</td>
<td>How often health states, alerts, and/or incidents are collected. The default is 120 seconds.</td>
</tr>
</tbody>
</table>

6. To view and copy the source GUID, click the button.

Note: It is important to know the GUID because it is used to define the indices of Elasticsearch. The source GUID is reported in the property sourceGuid and it is used to populate the property sourceId of any Document created with the Savision iQ REST
API. The Connector Type is always VirtualConnector and it is used to populate the property `sourceType` of the created Document.

7. Click the Roles tab and select a role. A new page displays.
8. Click a role and select Integrations.
9. Click the Add button.
10. Select an integration from the list and click Add.
11. Optional. If you want users in this role to have read-only access to the integration, select the Read-only box.
CHAPTER 4

Using the REST API

In order to communicate to the Elasticsearch node by using the Savision iQ REST API you need to have HTTP access to the Savision iQ Server address and port 9200.

By default, the Elasticsearch node binds to loopback addresses only, i.e. 127.0.0.1 and [::1]. If you want to have access from other servers, the Elasticsearch node will need to bind to a non-loopback address: in the Elasticsearch configuration file “elasticsearch.yml” modify the setting “network.host” to make the node bind to a different hostname or IP address. For more details, please consult the Elasticsearch documentation.

The pattern of how to access data in Elasticsearch can be generally summarized as follows:

```
<REST Verb> <Savision iQ Server>:9200/<Index>/<Type>/<ID>
```

The REST verbs used to interact with Elasticsearch are GET (Retrieve/List), PUT (Create/Update), DELETE (Delete) and POST (Batch Processing).

Command-line tools like `curl`, PowerShell’s `Invoke-RestMethod` or visual tools like `Postman` can be used to perform REST requests.

To illustrate the Savision iQ REST API, we will make use of the curl syntax in the following examples. For simplicity the Unix syntax is adopted; if you use curl in a Microsoft Windows command prompt, you should replace all the single quotes (‘) with double quotes (" ) and any double quote (" ) inside single quotes with two double quotes (""").

Create and Update a Document

To create or update an existing document, you use the Index API, specifying the index, the type and the ID of document.

The basic syntax for indexing a document is:

```
PUT <Savision iQ Server>:9200/<Index>/<Type>/<ID>
```

You must provide the object to persist in JSON format in the body of the request. The ID of the document must be equals to the key property of the object to persist.
For example to index a simple Component document into the `savisioniq_components_075261f3-e421-41d7-b71b-8320dccb194` index, using the `curl` tool:

```
curl -XPUT 'localhost:9200/savisioniq_components_075261f3-e421-41d7-b71b-8320dccb194/esentity/075261f3-e421-41d7-b71b-8320dccb194|SRV0000001?routing=075261f3-e421-41d7-b71b-8320dccb194|SRV0000001' -H 'Content-Type: application/json' -d'
{
    "joinKey": "parent",
    "name": "Testing Server 1",
    "typeEnum": 4,
    "key": "075261f3-e421-41d7-b71b-8320dccb194|SRV0000001",
    "sourceId": "075261f3-e421-41d7-b71b-8320dccb194",
    "sourceName": "Samanage 01",
    "sourceType": "VirtualConnector",
    "source": {
        "api": {
            "DisplayName": "Testing Server 1",
            "Id": "SRV0000001",
            "TimeAdded": "2017-02-07T18:04:24.4991017",
        }
    }
}
```

Note how the Document ID matches the Component key. In addition, you need to specify the routing parameter in the request, and the routing parameter must equal the `key` property of the component.

And the response:
Chapter 4 Using the REST API

From the above, we can see that a new document of type esentity was successfully created inside the specified index. The document also has an internal id of “075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001” which we specified at index time.

It is important to note that Elasticsearch does not require you to explicitly create an index first before you can index documents into it. In the previous example, Elasticsearch will automatically create the index “savisioniq_components_075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001” if it didn’t already exist beforehand.

To update an existing document you can execute the above command again with a different (or same) JSON object; in fact Elasticsearch will replace (i.e. re-index) a new document on top of the existing one with the ID of “075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001”.

If, on the other hand, you use a different ID, a new document will be indexed and the existing document(s) already in the index remains untouched.

Retrieve a Document

To retrieve an existing document, you use the Get API, specifying the index, the type and the ID of document.

The basic syntax for retrieving a document is:

```
GET <Savision iQ Server>:9200/<Index>/<Type>/<ID>
```

The following example shows how use the curl tool to retrieve the Component document that was previously indexed:

```
curl -XGET 'localhost:9200/savisioniq_components_075261f3-e421-41d7-b71b-8320dccbf194/esentity/075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001?routing=075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001'
```
You need to specify the routing parameter in the request, and the routing parameter must equal the **key** property of the component.

Here is the response:

```json
{
    "_index": "savisioniq_components_075261f3-e421-41d7-b71b-8320dccbf194",
    "_type": "esentity",
    "_id": "075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001",
    "_version": 1,
    "found": true,
    "_source": {
        "joinKey": "parent",
        "name": "Testing Server 1",
        "typeEnum": 4,
        "key": "075261f3-e421-41d7-b71b-8320dccbf194|SRV0000001",
        "sourceId": "075261f3-e421-41d7-b71b-8320dccbf194",
        "sourceName": "Samanage 01",
        "sourceType": "VirtualConnector",
        "source": {
            "api": {
                "DisplayName": "Testing Server 1",
                "Id": "SRV0000001",
                "TimeAdded": "2017-02-07T18:04:24.4991017",
            }
        }
    }
}
```

The response field found, reports if the requested document was found, while the field _source returns the full JSON document that was previously indexed.

### Delete a Document

To delete an existing document, you use the [Delete API](#), specifying the index, the type and the ID of document.

The basic syntax for deleting a document is:

```
DELETE <Savision iQ Server>:9200/<Index>/<Type>/<ID>
```

The following example shows how to use the curl tool to delete the Component document that was previously indexed:
You need to specify the routing parameter in the request, and the routing parameter must equal the key property of the component.

See the _delete_by_query API to delete all documents matching a specific query. It is worth noting that it is much more efficient to delete a whole index instead of deleting all documents with the Delete By Query API.

The response field found, reports if the requested document was found, while the field _source returns the full JSON document that was previously indexed.

**Batch Processing**

In addition to being able to index and delete individual documents, Elasticsearch also provides the ability to perform any of the above operations in batches using the bulk API. This functionality is important in that it provides a very efficient mechanism to do multiple operations as fast as possible with as few network roundtrips as possible.

The basic syntax for indexing a document is:

```bash
POST <Savision IQ Server>:9200/<Index>/<Type>/_bulk
```

You must provide the following newline delimited JSON (NDJSON) structure in the body of the request:

```json
action_and_meta_data
optional_source

action_and_meta_data
optional_source

...

action_and_meta_data
optional_source
```

For example, to index two Component documents in one bulk operation:
curl -XPOST 'localhost:9200/savisioniq_components_075261f3-e421-41d7-b71b-8320dcbff194/esentity/_bulk' -H 'Content-Type: application/json' -d

{"index":{"_id":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000002","routing":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000002"}}

{"joinKey":"parent","name":"Testing Server 2","typeEnum":4,"key":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000002"}

{"index":{"_id":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000003","routing":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000003"}}

{"joinKey":"parent","name":"Testing Server 3","typeEnum":4,"key":"075261f3-e421-41d7-b71b-8320dcbff194|SRV0000003"}

Note in the example above that for the delete action, there is no corresponding source document after it since deletes only require the ID of the document to be deleted.

The Bulk API does not fail due to failures in one of the actions. If a single action fails for whatever reason, it will continue to process the remainder of the actions after it. When the bulk API returns, it will provide a status for each action (in the same order it was sent in) so that you can check if a specific action failed or not.

Search Documents

To search the documents persisted into Elasticsearch, you use the Search API. Refer to the Elasticsearch documentation for details.
List all Indexes

To get information about all the indexes, you use the cat indices API. The basic syntax is:

```
GET <Savision iQ Server>:9200/_cat/indices
```

Using the curl tool, the basic syntax is:

```
curl -XGET 'localhost:9200/_cat/indices'
```

Delete an Index

To delete an existing index, you use the Delete Index API, specifying the index.

The basic syntax for deleting an index is:

```
DELETE <Savision iQ Server>:9200/<Index>
```

The following example shows how to use the curl tool to delete an index that was previously created:

```
curl -XDELETE 'localhost:9200/savisioniq_components_075261f3-e421-41d7-b71b-8320dccbf194'
```