Search semi-structured data in Azure cloud Storage

rogara

# Search semi-structured data in cloud storage

In this two-part tutorial series, you learn how to search semi-structured and unstructured data. In this part, you search semi-structured data, such as JSON. Semi-structured data contains tags or markings which separate content within the data. It differs from structured data in that it is data that is not formally structured such as data models for relational databases.

In this part we cover how to:

[!div class="checklist"] \* Create an Azure Search Service using the REST API \* Use the Azure Search Service to search your container

## Prerequisites

To complete this tutorial:

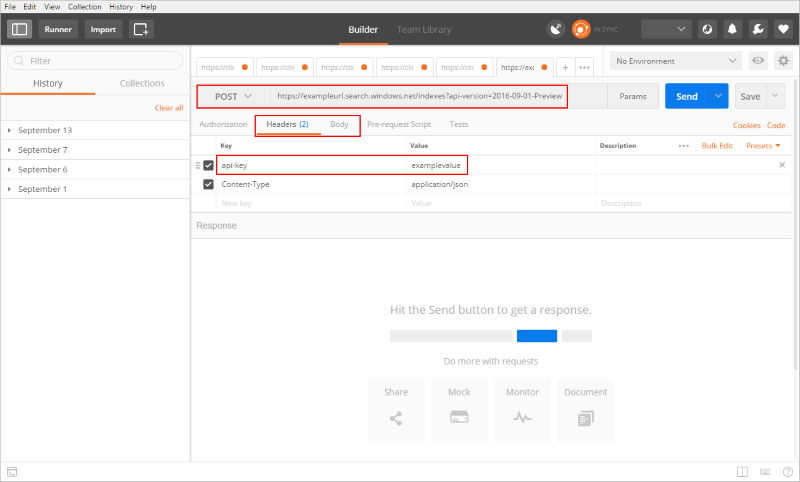
* Complete the [previous tutorial](./storage-unstructured-search.md)
* Have A REST client installed
* Have some familiarity with REST API queries

## Setup

To complete this tutorial you need a REST client. For the purposes of this tutorial, we are using [Postman](https://www.getpostman.com/). Feel free to use a different REST client if you're already comfortable with a particular one.

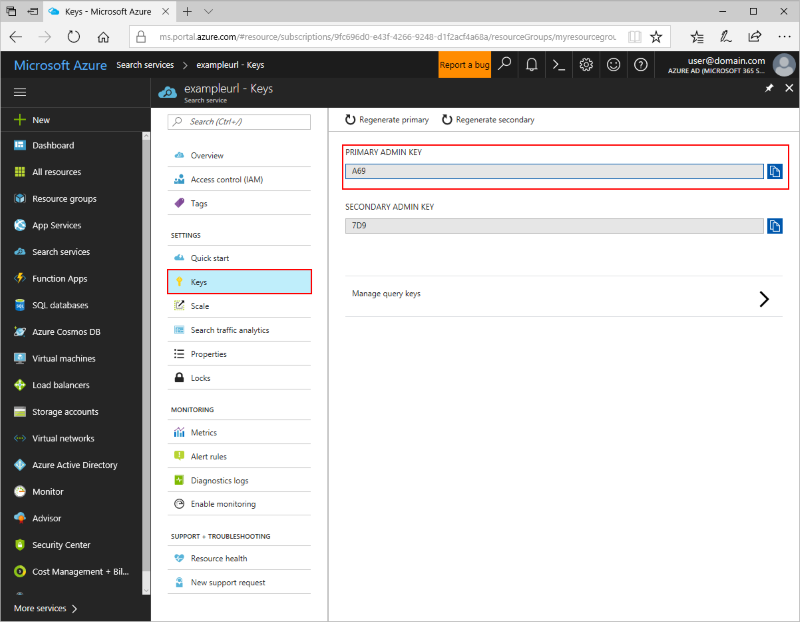
After installing postman, launch it.

If this is your first time making REST calls to Azure, here's a brief introduction of the important components for this tutorial: The request method for every call in this tutorial is "POST." The header keys, which are "Content-type" and "api-key." The values of the header keys, which are "application/json" and your "admin key" (this is a placeholder for your search primary key) respectively. The body is where you place the actual contents of your call. Depending on the client you're using, there may be some variations on how you construct your query but those are the basics.



Semi-structured search

For the calls covered in this tutorial, search api-key is required, it can be found under **Keys** inside your search service. This api-key must be in the header of every API call (place it where "admin key" in the preceding screenshot) this tutorial directs you to make. Retain this since you need it for each call.



Semi-structured search

## Download the sample data

A sample data set has been prepared for you. **Download** [**clinical-trials-json.zip**](https://github.com/roygara/storage-blob-integration-with-cdn-search-hdi/raw/master/clinical-trials-json.zip) and unzip it to its own folder.

Contained in the sample are example JSON files, which were originally text files obtained from [clinicaltrials.gov](https://clinicaltrials.gov/ct2/results). We have converted them to JSON for your convenience.

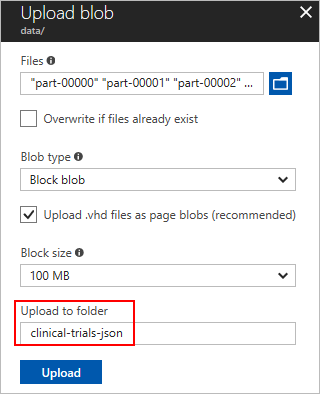
## Log in to Azure

Log in to the [Azure portal](http://portal.azure.com).

## Upload the sample data

Navigate back to the storage account from the previous tutorial, open the **data** container, and click **Upload**.

Click **Advanced** and enter "clinical-trials-json", then upload all of the JSON files you downloaded.



Semi-structured search

After the upload completes, the files should appear in their own subfolder inside the data container.

## Connect your search service to your blob storage

We are using the REST API to perform the connection because the UI does not currently support JSON indexing.

The querystring must contain **api-version=2016-09-01-Preview** and each call should return a **201 Created**. The generally available api-version does not yet have the capability to handle json as a jsonArray, currently only the preview api-version does.

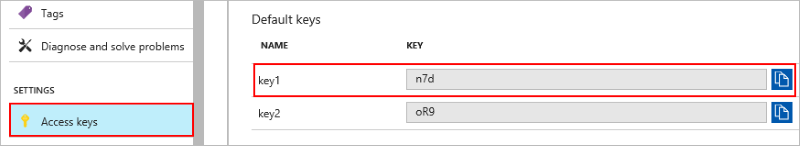
Execute the following three API calls from your REST client.

### Create a datasource

A data source is what you use to specify what data to index.

The endpoint of this call is https://[service name].search.windows.net/datasources?api-version=2016-09-01-Preview, replace [service name] with the name of your search service.

For this call, you need the name of your storage account and your storage account key, the storage account key can be found in your storage account under **Access Keys**. Location pictured following:



Semi-structured search

Make sure to replace the [storage account name] and [storage account key] in the body of your call before executing the call.

{  
 "name" : "clinical-trials-json",  
 "type" : "azureblob",  
 "credentials" : { "connectionString" : "DefaultEndpointsProtocol=https;AccountName=[storage account name];AccountKey=[storage account key];" },  
 "container" : { "name" : "data", "query" : "clinical-trials-json" }  
}

The response should look like:

{  
 "@odata.context": "https://exampleurl.search.windows.net/$metadata#datasources/$entity",  
 "@odata.etag": "\"0x8D505FBC3856C9E\"",  
 "name": "clinical-trials-json",  
 "description": null,  
 "type": "azureblob",  
 "subtype": null,  
 "credentials": {  
 "connectionString": "DefaultEndpointsProtocol=https;AccountName=[mystorageaccounthere];AccountKey=[[myaccountkeyhere]]];"  
 },  
 "container": {  
 "name": "data",  
 "query": "clinical-trials-json"  
 },  
 "dataChangeDetectionPolicy": null,  
 "dataDeletionDetectionPolicy": null  
}

### Create an index

The second API call creates an index, it sets all the parameters and their attributes.

The URL for this call is https://[service name].search.windows.net/indexes?api-version=2016-09-01-Preview, replace [service name] with the name of your search service.

Replace the URL, then copy and paste the following code into your body and run the query.

{  
 "name": "clinical-trials-json-index",   
 "fields": [  
 {"name": "FileName", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": true},  
 {"name": "Description", "type": "Edm.String", "searchable": true, "retrievable": false, "facetable": false, "filterable": false, "sortable": false},  
 {"name": "MinimumAge", "type": "Edm.Int32", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": true},  
 {"name": "Title", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": true},  
 {"name": "URL", "type": "Edm.String", "searchable": false, "retrievable": false, "facetable": false, "filterable": false, "sortable": false},  
 {"name": "MyURL", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": false},  
 {"name": "Gender", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},  
 {"name": "MaximumAge", "type": "Edm.Int32", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": true},  
 {"name": "Summary", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": false, "sortable": false},  
 {"name": "NCTID", "type": "Edm.String", "key": true, "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": true},  
 {"name": "Phase", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},  
 {"name": "Date", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": false, "filterable": false, "sortable": true},  
 {"name": "OverallStatus", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},  
 {"name": "OrgStudyId", "type": "Edm.String", "searchable": true, "retrievable": true, "facetable": false, "filterable": true, "sortable": false},  
 {"name": "HealthyVolunteers", "type": "Edm.String", "searchable": false, "retrievable": true, "facetable": true, "filterable": true, "sortable": false},  
 {"name": "Keywords", "type": "Collection(Edm.String)", "searchable": true, "retrievable": true, "facetable": true, "filterable": false, "sortable": false},  
 {"name": "metadata\_storage\_last\_modified", "type":"Edm.DateTimeOffset", "searchable": false, "retrievable": true, "filterable": true, "sortable": false},  
 {"name": "metadata\_storage\_size", "type":"Edm.String", "searchable": false, "retrievable": true, "filterable": true, "sortable": false},  
 {"name": "metadata\_content\_type", "type":"Edm.String", "searchable": true, "retrievable": true, "filterable": true, "sortable": false}  
 ],  
 "suggesters": [  
 {  
 "name": "sg",  
 "searchMode": "analyzingInfixMatching",  
 "sourceFields": ["Title"]  
 }  
 ]  
}

The response should look like:

{  
 "@odata.context": "https://exampleurl.search.windows.net/$metadata#indexes/$entity",  
 "@odata.etag": "\"0x8D505FC00EDD5FA\"",  
 "name": "clinical-trials-json-index",  
 "fields": [  
 {  
 "name": "FileName",  
 "type": "Edm.String",  
 "searchable": false,  
 "filterable": false,  
 "retrievable": true,  
 "sortable": true,  
 "facetable": false,  
 "key": false,  
 "indexAnalyzer": null,  
 "searchAnalyzer": null,  
 "analyzer": null,  
 "synonymMaps": []  
 },  
 {  
 "name": "Description",  
 "type": "Edm.String",  
 "searchable": true,  
 "filterable": false,  
 "retrievable": false,  
 "sortable": false,  
 "facetable": false,  
 "key": false,  
 "indexAnalyzer": null,  
 "searchAnalyzer": null,  
 "analyzer": null,  
 "synonymMaps": []  
 },  
 ...  
 "scoringProfiles": [],  
 "defaultScoringProfile": null,  
 "corsOptions": null,  
 "suggesters": [],  
 "analyzers": [],  
 "tokenizers": [],  
 "tokenFilters": [],  
 "charFilters": []  
}

### Create an index

An indexer connects the data source to the target search index and (optionally) provides a schedule to automate the data refresh.

The URL for this call is https://[service name].search.windows.net/indexers?api-version=2016-09-01-Preview, replace [service name] with the name of your search service.

Replace the URL, then copy and paste the following code into your body and run the query.

{  
 "name" : "clinical-trials-json-indexer",  
 "dataSourceName" : "clinical-trials-json",  
 "targetIndexName" : "clinical-trials-json-index",  
 "parameters" : { "configuration" : { "parsingMode" : "jsonArray" } }  
}

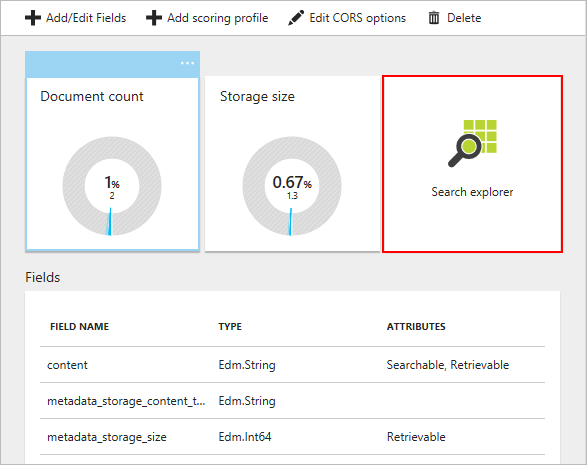
The response should look like:

{  
 "@odata.context": "https://exampleurl.search.windows.net/$metadata#indexers/$entity",  
 "@odata.etag": "\"0x8D505FDE143D164\"",  
 "name": "clinical-trials-json-indexer",  
 "description": null,  
 "dataSourceName": "clinical-trials-json",  
 "targetIndexName": "clinical-trials-json-index",  
 "schedule": null,  
 "parameters": {  
 "batchSize": null,  
 "maxFailedItems": null,  
 "maxFailedItemsPerBatch": null,  
 "base64EncodeKeys": null,  
 "configuration": {  
 "parsingMode": "jsonArray"  
 }  
 },  
 "fieldMappings": [],  
 "enrichers": [],  
 "disabled": null  
}

## Search your JSON files

Now that the search service has been connected to your data container you can begin searching your files.

Open up the Azure portal and navigate back to your search service. Just like you did in the previous tutorial.

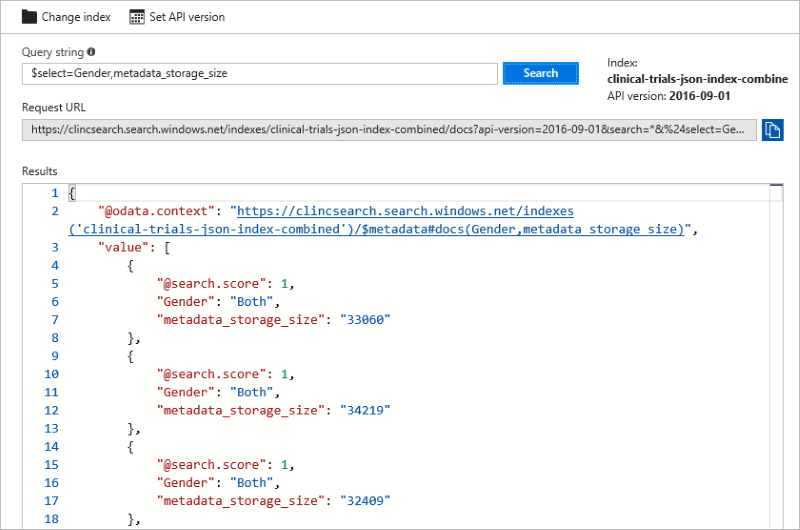


Unstructured search

### User-defined metadata search

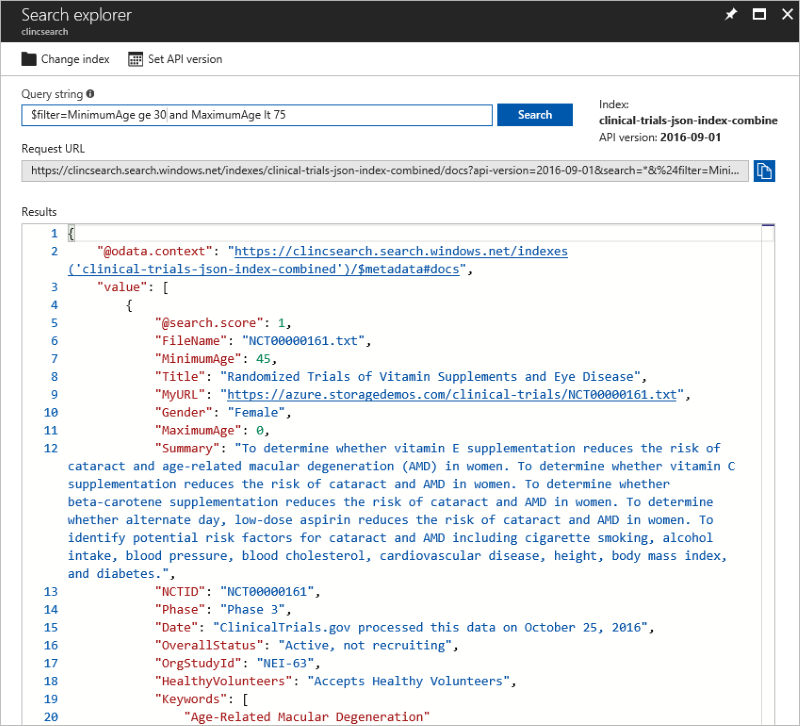
As before, the data can be queried in a number of ways: full text search, system properties, or user-defined metadata. Both system properties and user-defined metadata may only be searched with the select parameter if they were marked as retrievable during creation of the target index. Once parameters are created in the target index they may not be altered, though additional parameters may be added.

A basic query would be $select=Gender,metadata\_storage\_size, which limits the return to those two parameters.



Semi-structured search

A more complex query would be $filter=MinimumAge ge 30 and MaximumAge lt 75, which returns only results where the parameters MinimumAge is greater than or equal to 30 and MaximumAge is less than 75.



Semi-structured search

If you'd like to experiment and try a few more queries yourself, feel free to do so. Know that Logical operators (and, or, not) work as well as comparison operators (eq, ne, gt, lt, ge, le). String comparisons are case-sensitive.

The $filter parameter only works with metadata that were marked filterable at the creation of your index.

## Next steps

In this tutorial, you learned about searching semi-structured data such as how to:

[!div class="checklist"] \* Create an Azure Search Service using the REST API \* Use the Azure Search Service to search your container

Follow this link to learn more about search.

[!div class="nextstepaction"](https://docs.microsoft.com/en-us/azure/search/search-what-is-azure-search)