***HCL Commerce V9.1***

***ElasticSearch - Ingest unstructured data***

**Implementation guide**

**Draft - Version 0.2**

**September 27, 2021**



**Document Source**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version #** | **Date** | **Version Description and Summary of changes** | **Editor** |
| 0.1 | July 23, 2021 | Draft - First version for review | Ravi Rai |
| 0.2 | September 27, 2021 | Draft – Updated for Commerce v9.1.6 | Craig Oakley |
| 0.3 | Dec 19, 2022 | Review and Update in v9.1.11 | Chaitanya Mulaguru |

DISCLAIMER:

The information contained and discussed in connection with this document is **confidential and proprietary to HCL**. The use and dissemination of this document and the information contained and discussed therewith are subject to confidentiality obligations and restrictions on use. Any information contained herein is subject to change at the sole discretion of HCL and its related entities and is not a binding commitment or representation on the development, release, or timing of any products, features, or functionality

This documented was created with the intent to showcase how-to work with NiFi. It was intended as a sample exercise of how to ingest unstructured content into elastic search through nifi container for HCL Commerce V9.1 Search based on Elasticsearch.

The documentation was created using HCL Commerce Developer v9.1.6 and is not currently intended to be updated beyond the specified released.

Business Scenario / Functional Requirements

Ingest unstructured data for MS Word .docx, PDFs, Excel .xlsx, CSV files, .txt files, HTML files, media metadata.

High Level Design (HLD)

Need to be enabled attachment in existing elastic search container and ingest unstructured data for MS Word .docx, PDFs, Excel .xlsx, CSV files, .txt files, HTML files, media metadata from nifi processes group and associate with existing product index or create new unstructured index and ingest data in new schema.

Implementation guide steps

**Step 1:**  Need to enable the attachment in elastic search container (docker.elastic.co/elasticsearch/elasticsearch:7.8.0), run the following command from terminal window

docker exec -it -u 0 commerce\_elasticsearch\_1 bash

**run the below command from container bash terminal**

elasticsearch-plugin install ingest-attachment

Restart the elasticsearch docker:

docker restart commerce\_elasticsearch\_1

**Note:** We can add attachment command in dockerfile, then we will need to build new elastic search image from base image.

**Step 2:** Create a directory into nifi container (commerce/search-nifi-app:9.1.6.0)

1. docker exec -it -u 0 commerce\_nifi\_1 bash
2. mkdir /opt/nifi/extDocs/
3. chown nifi:nifi /opt/nifi/extDocs
4. copy the files in the directory which we want to ingest into elsaticsearch, use the following command.

docker cp /home/ingestfiles/. commerce\_nifi\_1:/opt/nifi/extDocs/.

**Note:** For this example, we will be using the files:

SampleDocs-travel-laptop.docx  
SampleDocs-office-laptop.ppt.

**Step 3:** Import the following connectors into Runtime registry (commerce/search-registry-app:9.1.6.0).

1. docker cp custom-UnstructuredIndexSchemaUpdateConnector-attachment.json commerce\_registry\_1:/opt/nifi-registry/flows/.
2. docker cp custom-UnstructuredIndexSchemaUpdate.json commerce\_registry\_1:/opt/nifi-registry/flows/.
3. docker cp custom-UnstructuredIndexDatabaseConnectorPipe-Attachment.json commerce\_registry\_1:/opt/nifi-registry/flows/.

**Open the nifi registry container, run the following command**

docker exec -it -u 0 commerce\_registry\_1 bash

**Run the following commands from the registry terminal**

1. /opt/nifi-registry/scripts/import\_flow.sh custom-UnstructuredIndexSchemaUpdateConnector-attachment /opt/nifi-registry/flows/custom-UnstructuredIndexSchemaUpdateConnector-attachment.json
2. /opt/nifi-registry/scripts/import\_flow.sh custom-UnstructuredIndexSchemaUpdate /opt/nifi-registry/flows/custom-UnstructuredIndexSchemaUpdate.json
3. /opt/nifi-registry/scripts/import\_flow.sh custom-UnstructuredIndexDatabaseConnectorPipe-Attachment /opt/nifi-registry/flows/custom-UnstructuredIndexDatabaseConnectorPipe-Attachment.json

**Step 4:** Create a Connector using Ingest Swagger. Open the postman and add the following details

URL - http://localhost:30800/connectors

Method - POST

Body - json body

{

"name": "auth.unstructured",

"description": "This is the connector for the unstructured processing",

"pipes": [

{

"name": "custom-UnstructuredIndexSchemaUpdate"

},

{

"name": "custom-UnstructuredIndexSchemaUpdateConnector-attachment"

},

{

"name": "custom-UnstructuredIndexDatabaseConnectorPipe-Attachment",

"properties": [

{

"name": "Database Driver Location(s)",

"value": "${AUTH\_JDBC\_DRIVER\_LOCATION}",

"scope": {

"name": "Database Connection Pool",

"type": "CONTROLLER\_SERVICE"

}

},

{

"name": "Database Connection URL",

"value": "${AUTH\_JDBC\_URL}",

"scope": {

"name": "Database Connection Pool",

"type": "CONTROLLER\_SERVICE"

}

},

{

"name": "Database User",

"value": "${AUTH\_JDBC\_USER\_NAME}",

"scope": {

"name": "Database Connection Pool",

"type": "CONTROLLER\_SERVICE"

}

},

{

"name": "Password",

"value": "${AUTH\_JDBC\_USER\_PASSWORD}",

"scope": {

"name": "Database Connection Pool",

"type": "CONTROLLER\_SERVICE"

}

}

]

},

{

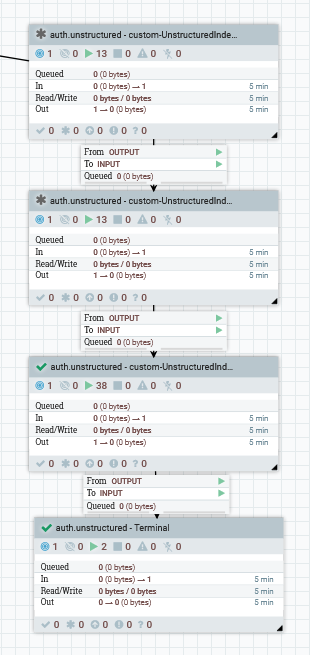
"name": "Terminal"

}   
 ]

}

**Step 5**: After running the connector, the following four process group pipes will be available in nifi, Add the process group in nifi dashboard and connect input/output ports as mentioned in the following image

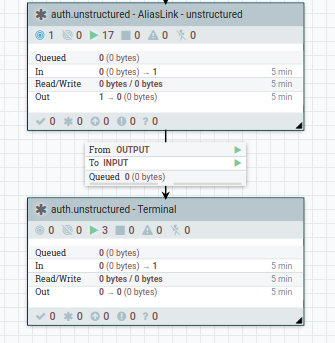
**a)**



Copy any alias link process group , ex : “auth.reindex – AliasLink – Product” and paste it in between In between 3rd and 4th process groups in the above pipeline. The new pipeline would look as below

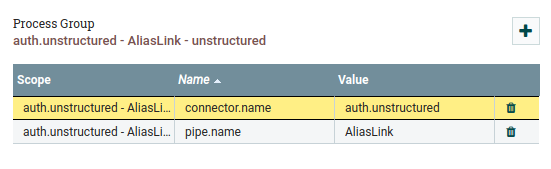
Graphical user interface, application

Description automatically generated

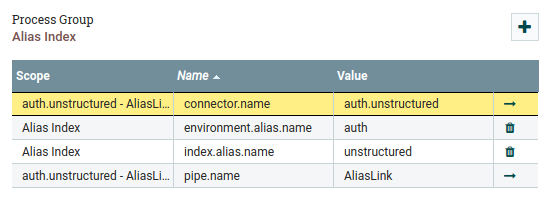


Right click the copied “**AliasLink**” process group and click on configure. Change the Process Group Name to “**auth.unstructured - AliasLink – unstructured**”.

Right click on “AliasLink” process group and click on “variables”. Add the data as shown in below image and remove others



Double click on “AliasLink” process group 🡪 right click on “Alias Index” and click on “variables”. Edit the data as shown in below image.



**Process Groups(pipes) :** In the above image using the four process group.

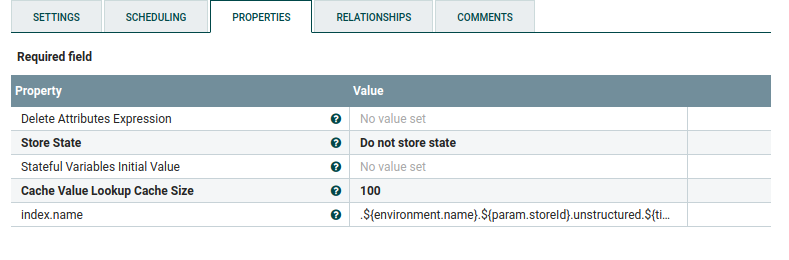
1. **custom-UnstructuredIndexSchemaUpdate :** This process group will be used to set new schema, it will skip if the schema will already be available in elastic search.

We can use existing schema also, just update the index name in the following properties file for processor set unstructured schema name.

Go to the process group “Set unstructured index name” and change the index name

**NOTE:** Change index name as below

**Index.name** - .${environment.name}.${param.storeId}.unstructured.${time.id}



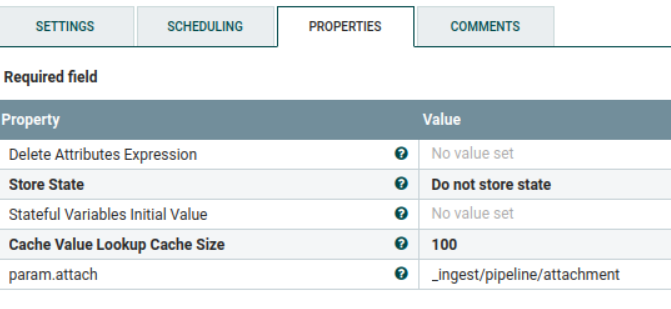
Go to process group **“Populate Unstructured Index schema**” process group and update the replace value with below one



1. **custom-UnstructuredIndexSchemaUpdateConnector-attachment:**  This process group will used for enabling the attachment setting in elastic search, it will skip if the attachment setting will already be available.

The following setting has been used by default; we can update as per our requirement.

The param.attach is available in ‘Set unstructured attachment ’ processor.



The following json is available in ‘Populate unstructured Index schema’ processor.

Here we can add/update keyword, this keyword will be used for ingesting/searching unstructured data.

**{**

**"description" : "Extract attachment information",**

**"processors" : [**

**{**

**"attachment" : {**

**"field" : "data",**

**"indexed\_chars\_field" : "max\_size",**

**"properties": [ "content", "title", "keywords", "content\_type", "content\_length" ]**

**}**

**}**

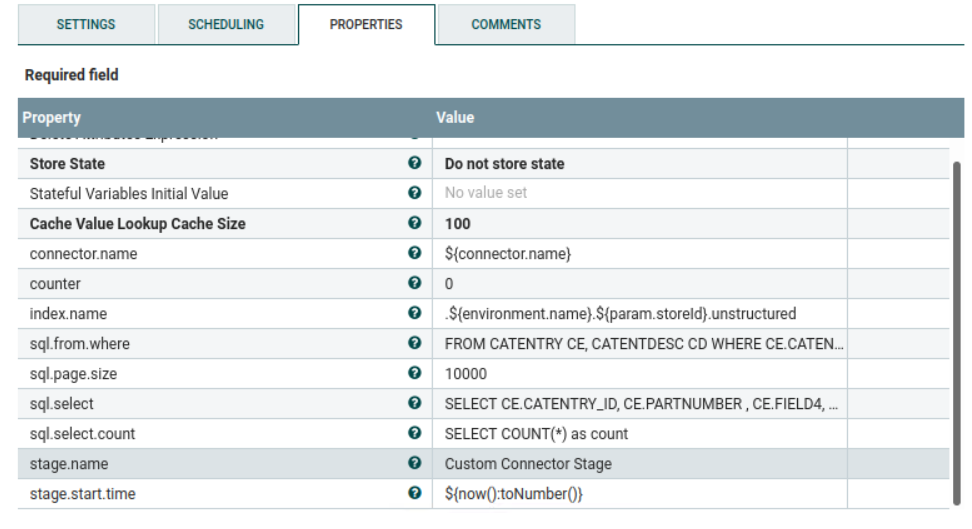
**]**

**}**

1. **custom-UnstructuredIndexDatabaseConnectorPipe-Attachment :** This process group are being used for fetching file location from db, read file content from directory mentioned in db, encode into base64 and ingest file content into the elastic search.

This process group are having multiple processors to ingest the files into elastic search.

The ‘Set Attribute’ processor are being used for setting the parameter which will be used for processing the file, we can update as per our need.



**IMP:** Go to “Update Attribute” process group and add the below property

fileABSPath - ${file.path:substringAfterLast('/')}

Go to “**AttributesToJson**” process group and update the property “Attributes List” with below value

filename,fileABSPath,file.path,productId,sku,product.name,short.desc,data,title,file\_type

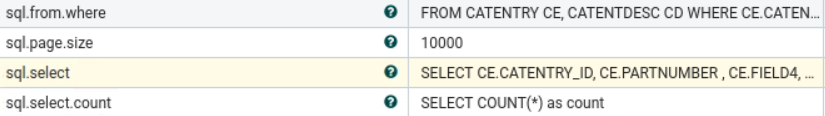
1. **Terminal :** This process group finishes the process flow and is required to terminate the output of the previous process.

**Step 6:** Run the following queries for setting the file location in Catentry.field4 (this table are currently being used in the process group), we can modify query and add multiple transaction as per need to setting the file location for catalog entries.

UPDATE catentry SET FIELD4 = '/opt/nifi/extDocs/SampleDocs-travel-laptop.docx' WHERE PARTNUMBER = 'CLA022\_2203'

UPDATE catentry SET FIELD4 = '/opt/nifi/extDocs/SampleDocs-office-laptop.ppt' WHERE PARTNUMBER = 'CLA022\_2205'

**Note :** If we want to set the file location in different table then we will have to modify the following properties in ‘Set Attrubute’ processor under **custom-UnstructuredIndexDatabaseConnectorPipe-Attachment.**



***Currently the below queries are being used, we can update/modify as per need.***

**SELECT COUNT(\*) as count FROM CATENTRY CE, CATENTDESC CD**

**WHERE CE.CATENTRY\_ID = CD.CATENTRY\_ID AND CD.LANGUAGE\_ID =-1 AND CE.MARKFORDELETE =0 AND CE.BUYABLE =1 AND CD.PUBLISHED =1 AND ce.FIELD4 IS not NULL**

**AND CE.CATENTRY\_ID IN (SELECT C.CATENTRY\_ID FROM CATGPENREL R, CATENTRY C**

**WHERE R.CATALOG\_ID IN**

**(SELECT CATALOG\_ID FROM STORECAT WHERE STOREENT\_ID IN**

**(SELECT RELATEDSTORE\_ID FROM STOREREL**

**WHERE STATE = 1 AND STRELTYP\_ID = -4 AND STORE\_ID = ${param.storeId}))**

**AND R.CATENTRY\_ID = C.CATENTRY\_ID AND C.MARKFORDELETE = 0 AND C.CATENTTYPE\_ID <> 'ItemBean')**

**SELECT CE.CATENTRY\_ID, CE.PARTNUMBER , CE.FIELD4, CD.NAME, CD.SHORTDESCRIPTION , CD.PUBLISHED FROM CATENTRY CE, CATENTDESC CD**

**WHERE CE.CATENTRY\_ID = CD.CATENTRY\_ID AND CD.LANGUAGE\_ID =-1 AND CE.MARKFORDELETE =0 AND CE.BUYABLE =1 AND CD.PUBLISHED =1 AND ce.FIELD4 IS not NULL**

**AND CE.CATENTRY\_ID IN (SELECT C.CATENTRY\_ID FROM CATGPENREL R, CATENTRY C**

**WHERE R.CATALOG\_ID IN**

**(SELECT CATALOG\_ID FROM STORECAT WHERE STOREENT\_ID IN**

**(SELECT RELATEDSTORE\_ID FROM STOREREL**

**WHERE STATE = 1 AND STRELTYP\_ID = -4 AND STORE\_ID = ${param.storeId}))**

**AND R.CATENTRY\_ID = C.CATENTRY\_ID AND C.MARKFORDELETE = 0 AND C.CATENTTYPE\_ID <> 'ItemBean')**

**Note :** If we want to update schema name where the file attachment will be ingest then we will have to modify the following properties in ‘Set Attrubute’ processor under **custom-UnstructuredIndexDatabaseConnectorPipe-Attachment.**



**Step 7:** Connect the **auth.unstructured - custom-UnstructuredIndexSchemaUpdate** process group with **‘Routing Service’** process group with INPUT **auth.unstructured**.

**Note:** This process should already be connected to the routing service if connector process is followed. Verify that the auth.unstructured route is available.

**Step 8:**  Navigate the following process group.



Select the any ‘Execute SQL’ processor, right click and select the ‘View Configuration’ then select arrow button at right side in the following property.

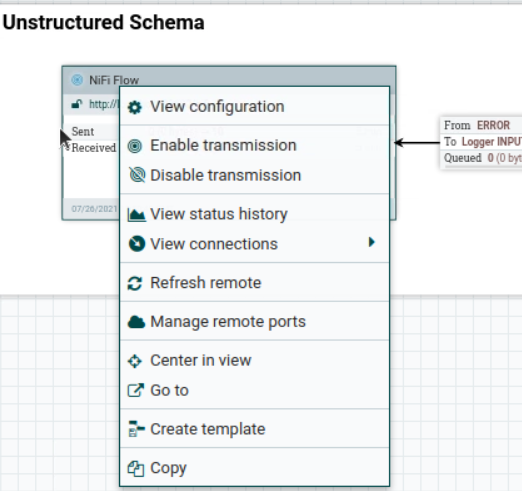


Make sure the Database Connection pool service is enabled.



**Step 9:** Start all the four process group, then navigate to inside each process group and right click on the Nifi Flow then select the ‘Enable Transmission’.

**Note:** transmissions may already be enabled.



**Steps 10:** After starting the process group run the following url from postman

POST- <https://localhost:5443/wcs/resources/admin/index/dataImport/build?connectorId=auth.unstructured&storeId=1>

For checking status:

GET- <https://localhost:5443/wcs/resources/admin/index/dataImport/status?jobStatusId=1036>

**Steps 11:**  Now, we can verify indexed unstructured data, we can pass keyword as what we have given during setting the attachment:

POST - localhost:30200/.auth.1.unstructured/\_search

Body: With Content available in file

{ "query": { "bool": {

"must": [

{

"query\_string": {

"query": "lightweight"

}

}

]

} } }

Body – with SKU (partnumber)

{ "query": { "bool": {

"must": [

{

"query\_string": {

"query": "CLA022\_2205"

}

}

]

} } }

Body – with file extension

{ "query": { "bool": {

"must": [

{

"query\_string": {

"query": "docx"

}

}

]

} } }

**Step 12:**  Now, we will create a new search API to access the unstructured index data

1. Create a new rest resource “UnstructuredContentResource”.



1. Create a new resource property file “**unstructured-resources.properties**”



1. Create a new custom postprocessor “**SearchWebContentPostprocessor**”



1. Select all these files and export them as jar.
2. Patch the jar in **search-query-app** and **search-query-appdata** containers

Copy the jar to the directory **/patches/** in both the containers.

Go to **/SETUP/bin/**

Run **./patch.sh ../../patches/{jarname}**

1. Create the new custom profile using below API.

**POST** - { **baseUrl** }/api/v2/documents/profiles/ **HCL\_findUnstructuredContentsBySearchTerm**



1. Create a new field mapping “**UnstructuredContentResponseFieldMapping**”. Pass the below json to the below api.

**POST - {{baseUrl}}/api/v2/configuration?nodeName=component&envType=auth**



1. Restart both **search-query-app** and **search-query-appdata** containers
2. Hit the new custom API to validate the data in response. This API has to be integrated in frontend

**http://localhost:3737/search/resources/store/1/sitecontent/unstructuredContentsBySearchTerm/{searchTerm}**

Text

Description automatically generated