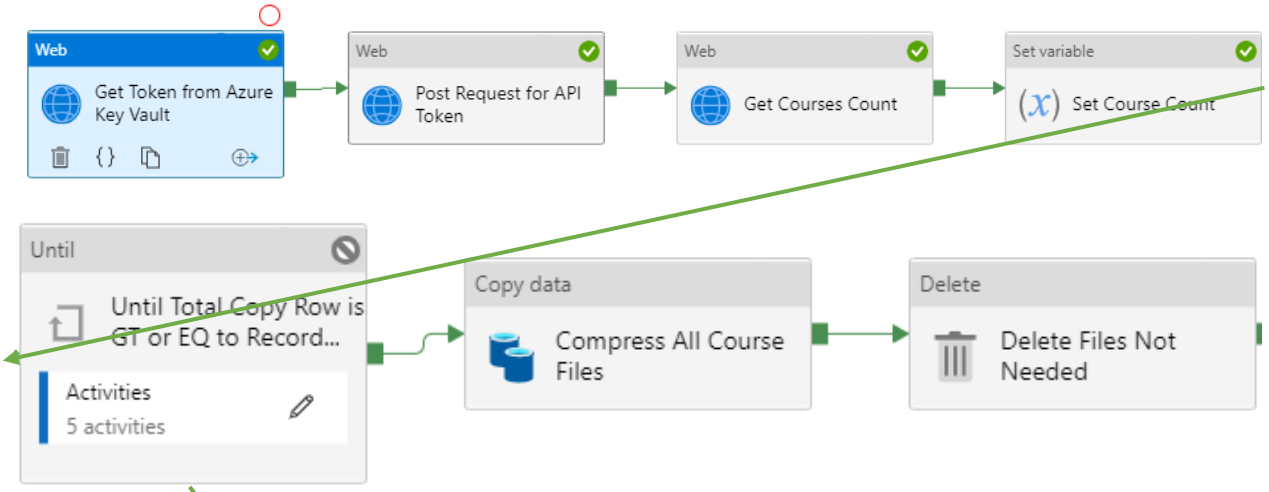
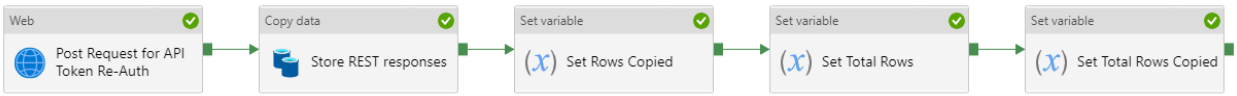
Copy and Page Through Large Ethos Integration Endpoint in Azure Data Factory and Store as JSON File

This example uses /api/courses Ethos endpoint.



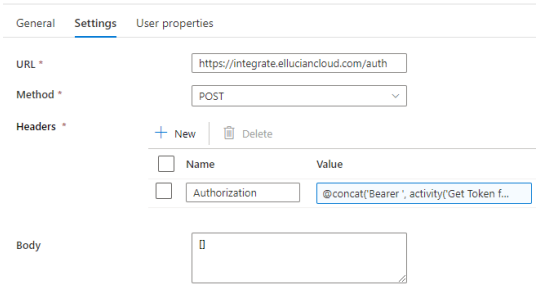
Until Activities



Step 1: Get Token from Azure Key Vault – We have a secret stored in AKV and use the AKV API to get the secret. Steps: [Use Azure Key Vault secrets in pipeline activities - Azure Data Factory | Microsoft Docs](). Set Secure Output on this step.

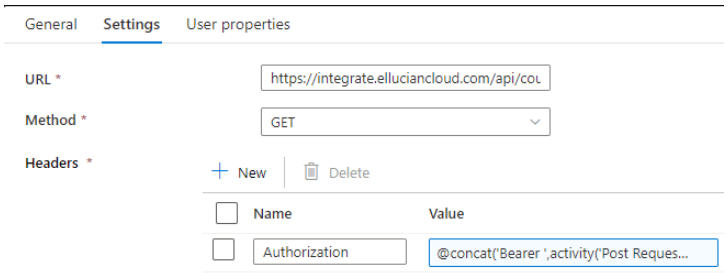
Step 2: Post Request for API Token – Takes token and attaches to auth header. Set Secure Input on this step.

1. URL = https://integrate.elluciancloud.com/auth
2. Header:
    a. Name = Authorization
    b. Value = @concat('Bearer ', activity('Get Token from Azure Key Vault').output.value)
3. Body = []



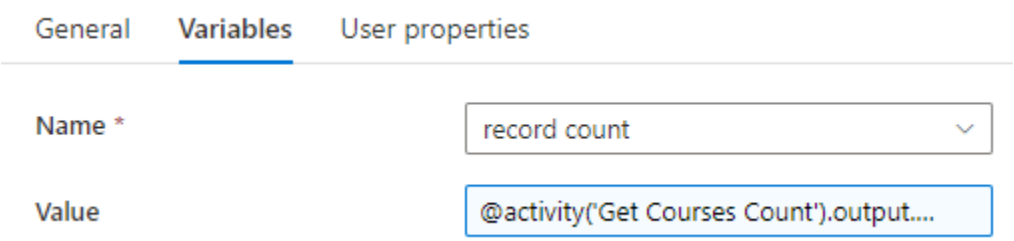
Step 3: Get Course Count – In this our step is to just call the end point and pick up the total number of records available from the endpoint.

1. URL = https://integrate.elluciancloud.com/api/courses
2. Headers: - used to get authorization token from POST and add Bearer key to header on GET call
    a. Name = Authorization
    b. Value = @concat('Bearer ',activity('Post Request for API Token').output.Response)



Step 4: Set Course Count – Our total is returned in the GET output as a header so we have to grab that value and set as a variable.

1. Value = @activity('Get Courses Count').output.ADFWebActivityResponseHeaders['x-total-count']



Step 5: Until Total Copy Row is GT or EQ to Record Count – Our endpoints can grow on the fly so we are cool with GT.

1. Expression = @greaterOrEquals(int(variables('total rows copied')),int(variables('record count')))

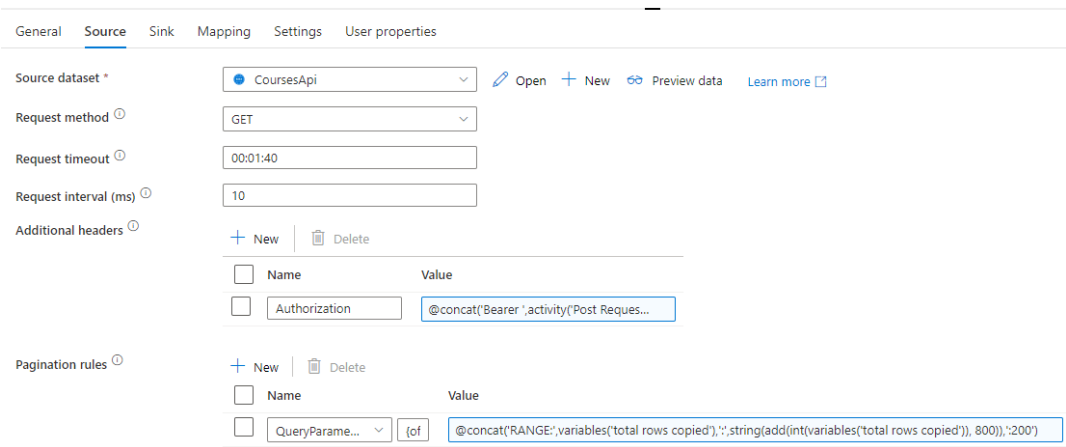
Until Activities:

UA Step 1: Post Request for API Token Re-Auth – We must reauthenticate due to timeout that we cannot react upon. Same as Step 2 above. Set Secure Input on this step.

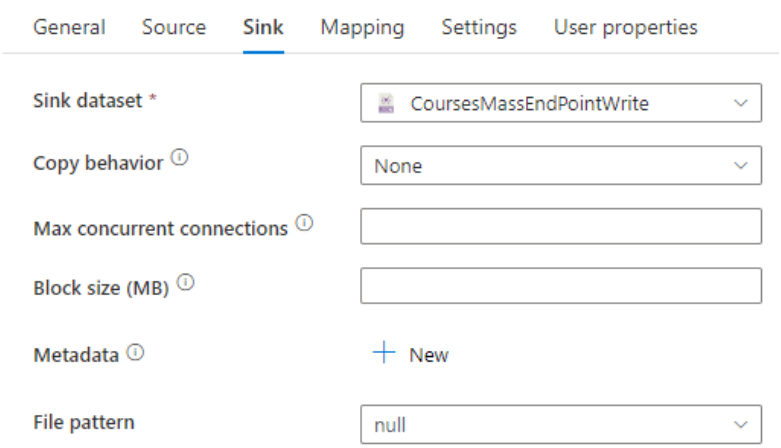
UA Step 2: Store REST responses – Addendum below with all dataset configurations to ADLS Gen 2 Storage. Pagination is used. This grabs 1000 records at a time and stores the response as a JSON file. Our endpoints are limited to 200 records at a time. Adjust these numbers as needed to get the number of records you want per call.

1. Header:
    a. Name = Authorization
    b. Value = @concat('Bearer ',activity('Post Request for API Token Re-Auth').output.Response)
2. Pagination Rules:
    a. Name = QueryParameters
    b. {offset}
    c. Value = @concat('RANGE:',variables('total rows copied'),':',string(add(int(variables('total rows copied')), 800)),':200')

Source:

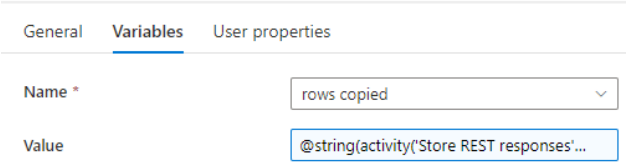


Sink:



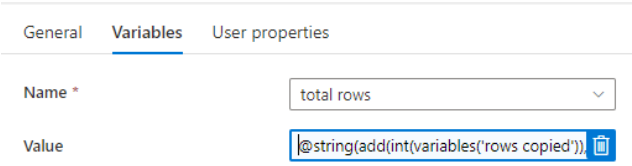
UA Step 3: Set Rows Copied – You need this so you can keep moving up your range each loop.

1. Name = rows copied
2. Value = @string(activity('Store REST responses').output['rowsCopied'])

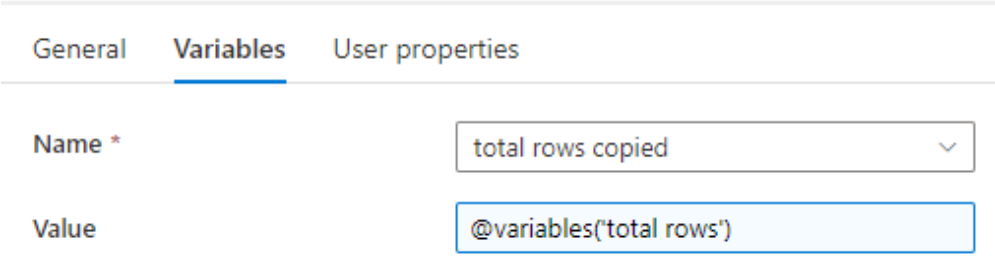


UA Step 4: Set Total Rows – You need this to add up and get a new total row to assign to Total Rows Copied. Variables cannot self-reference themselves.

1. Name = total rows
2. Value = @string(add(int(variables('rows copied')),int(variables('total rows copied'))))

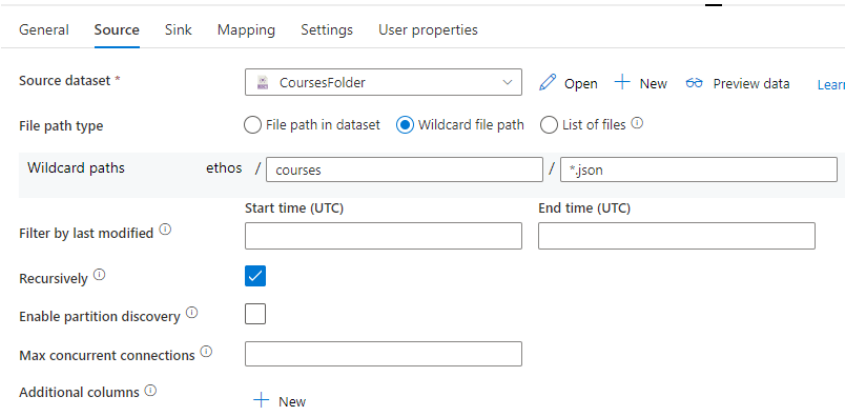


UA Step 5: Set Total Rows Copied

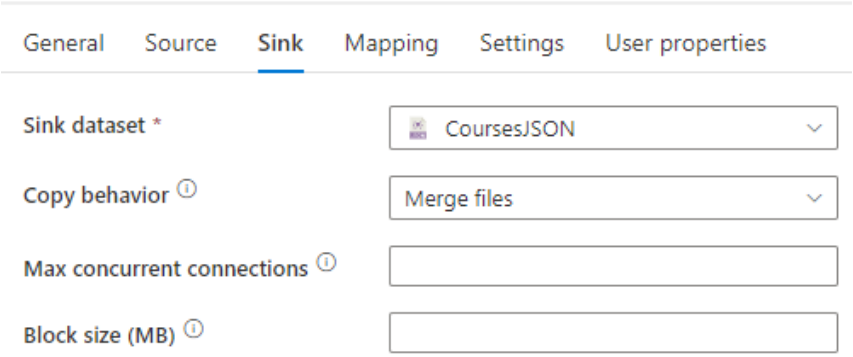


Step 6: Compress All Course Files – This will take all the individual files and merge into one large file called courses.json.

Source:

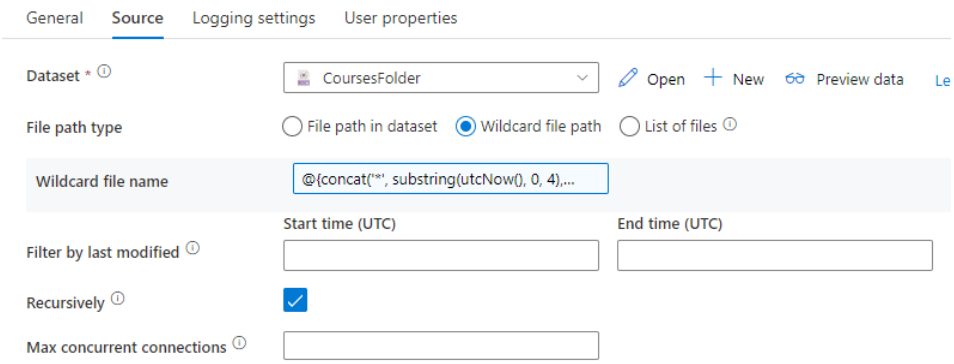


Sink:



Step 7: Delete Files Not Needed – This step is used to clean up all the small files. This will save money on storage. The below concat just gets the year in the file name to clean up smaller files.

1. Wildcard file name = @{concat('*', substring(utcNow(), 0, 4),'*')}



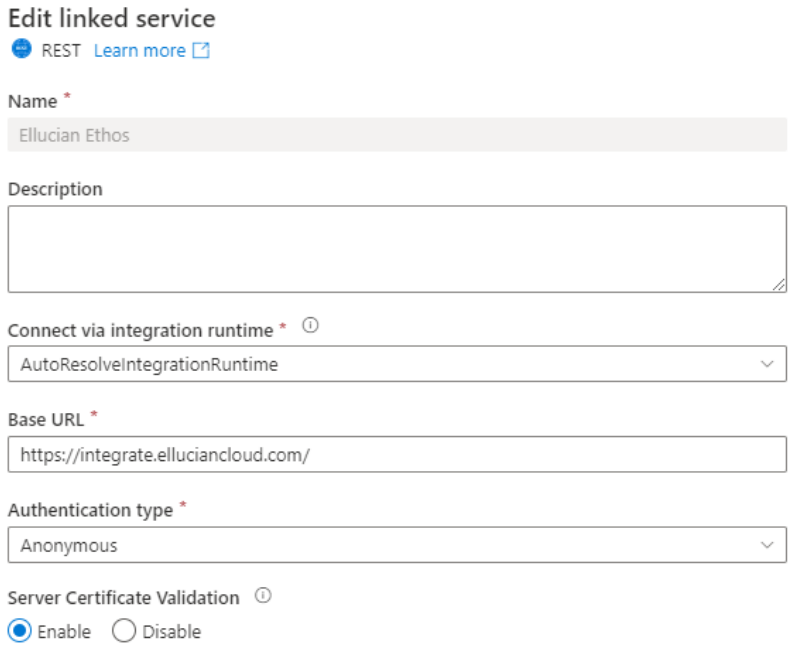
Additional Thoughts:

1. Since our endpoints can grow during initial calls it may be worth adding an additional step after the UNTIL to count endpoint again for total records. Subtract (total rows copied – total records) and determine if variable rows copied is smaller than the value returned from subtraction. If it is do something like grab additional records and copy to directory, else merge, delete, end.
2. We are using courses in this example but we need to parameterize folders and endpoints (datasets) so you can have one pipeline that flows through every endpoint. Not sure if this is doable since we are already have an iteration and nesting is not allowed. Article discussing parameterizing things that could be insightful. [Dynamic Datasets in Azure Data Factory | Under the kover of business intelligence (sqlkover.com)]
3. When compressing and creating large files we probably will need to be able to split these files by size for better performance in our zones that are higher up.

Addendum Linked Services and Datasets:

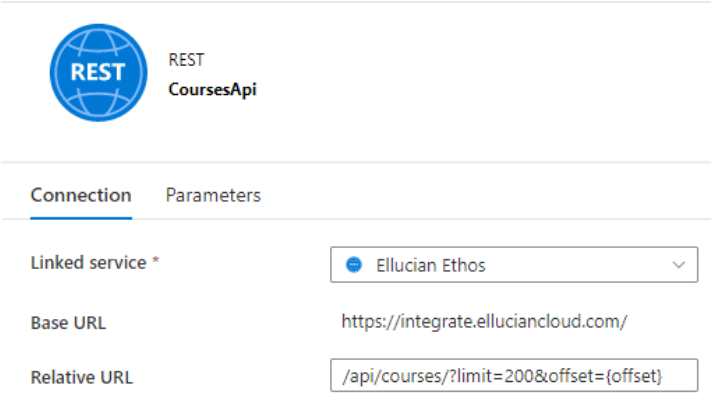
Linked Services

1. Create Linked Service to ADLS Gen2.
    a. How to Create an Azure Data Lake Gen2 Storage Linked Service: Part III | Tallan
2. Create a Linked Service for your RESTful service – Ours is for Ethos that uses a unique API token to request a bearer token that last for 5 minutes only. This is how ours is setup.
    a. Click New and filter for REST.

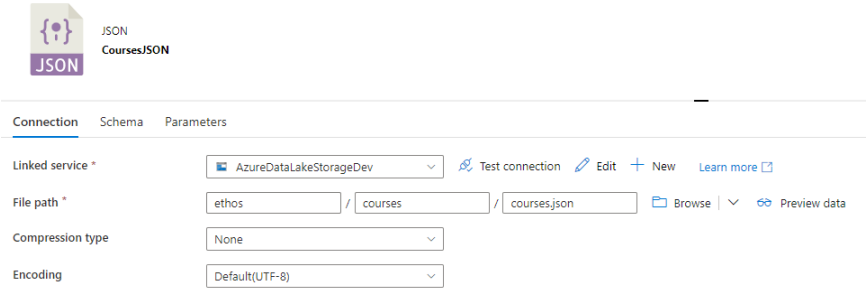


Datasets

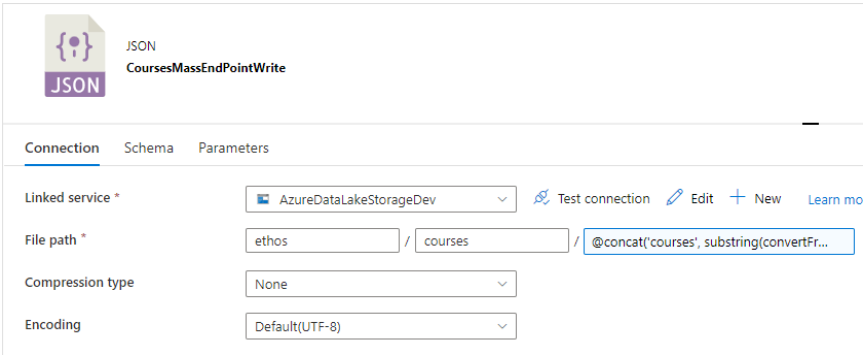
Note: Keep in mind your structure may be different.



Your variable above may be different. We cannot pull more than 200 records at a time. The offset matches up to the variables passed in the Copy Data activity pagination header.

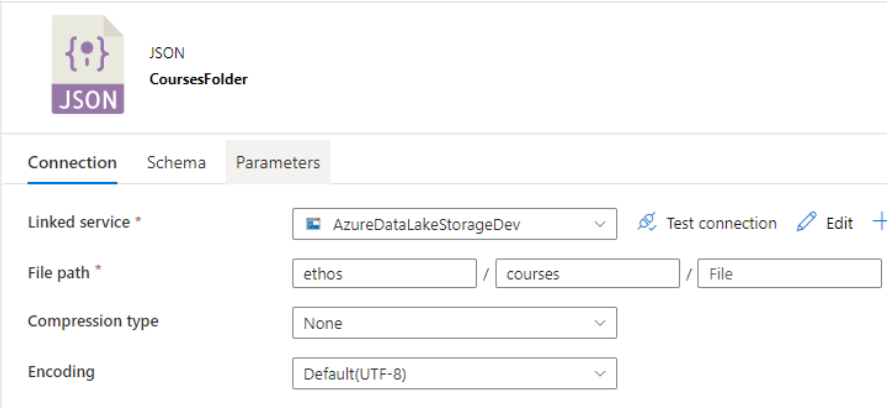


CoursesJSON dataset is just the main file that all smaller files are merged into.



This @concat allows for file to be named with a timestamp all the way out to the second. Feel free to tweak substring. Remember that year is used in the Delete activity wildcard.

@concat('courses', substring(convertFromUtc(utcNow(), 'Eastern Standard Time'), 0, 19),'.json')



The folder is needed for the Merge of the Copy Data to merge all smaller files.