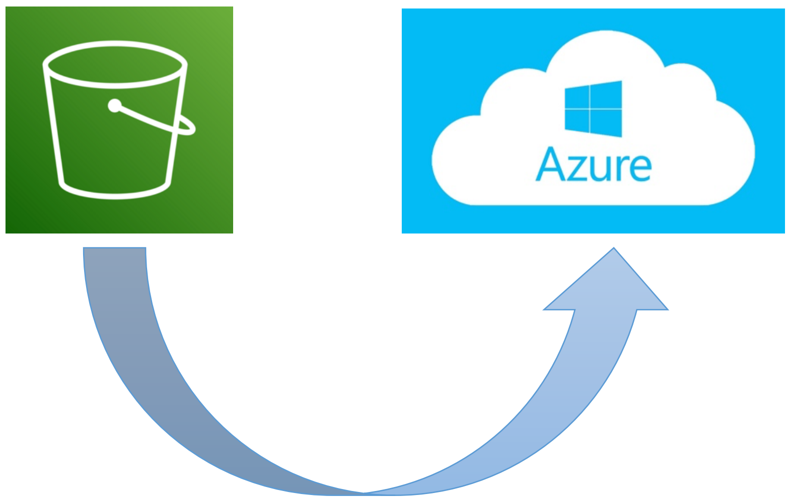


Smart Vehicles IOT Data Migration

Migrating Telemetry Data from Amazon S3 to Azure SQL Server

Project By

Rohit Amalnerkar

Objective:

The objective of this use case is to facilitate the seamless transfer of telemetry data from a third-party Internet of Things (IoT) device, connected to a vehicle, to Azure cloud infrastructure. The data, transmitted in JSON format, will first be validated for completeness and correctness before being stored in a SQL database. This validated data will serve as a valuable resource for Customer's Data Science team for analysis and insights.

Overview:

1. **Data Source:**

- Third-Party IoT Device Vehicles are equipped with third-party IoT devices capable of collecting and transmitting telemetry data.
- Telemetry data is sent in JSON format.

2. **Data Transmission: AWS Cloud**

- The telemetry data is initially sent to the AWS cloud, acting as an intermediary.

3. **Data Transfer: AWS to Azure Cloud**

- The primary task is to move data from the third-party AWS cloud to Azure cloud infrastructure.
- This transfer process ensures that data remains securely and efficiently accessible to Customer.

4. **Data Validation: JSON Integrity**

- Before the telemetry data is processed further, it undergoes validation.
- The validation step checks for completeness and correct JSON formatting.
- Any incomplete or incorrectly formatted JSON data is rejected.

5. **Data Storage: SQL Database**

- Validated telemetry data is stored in a SQL database.
- This structured storage facilitates organized and efficient data management.

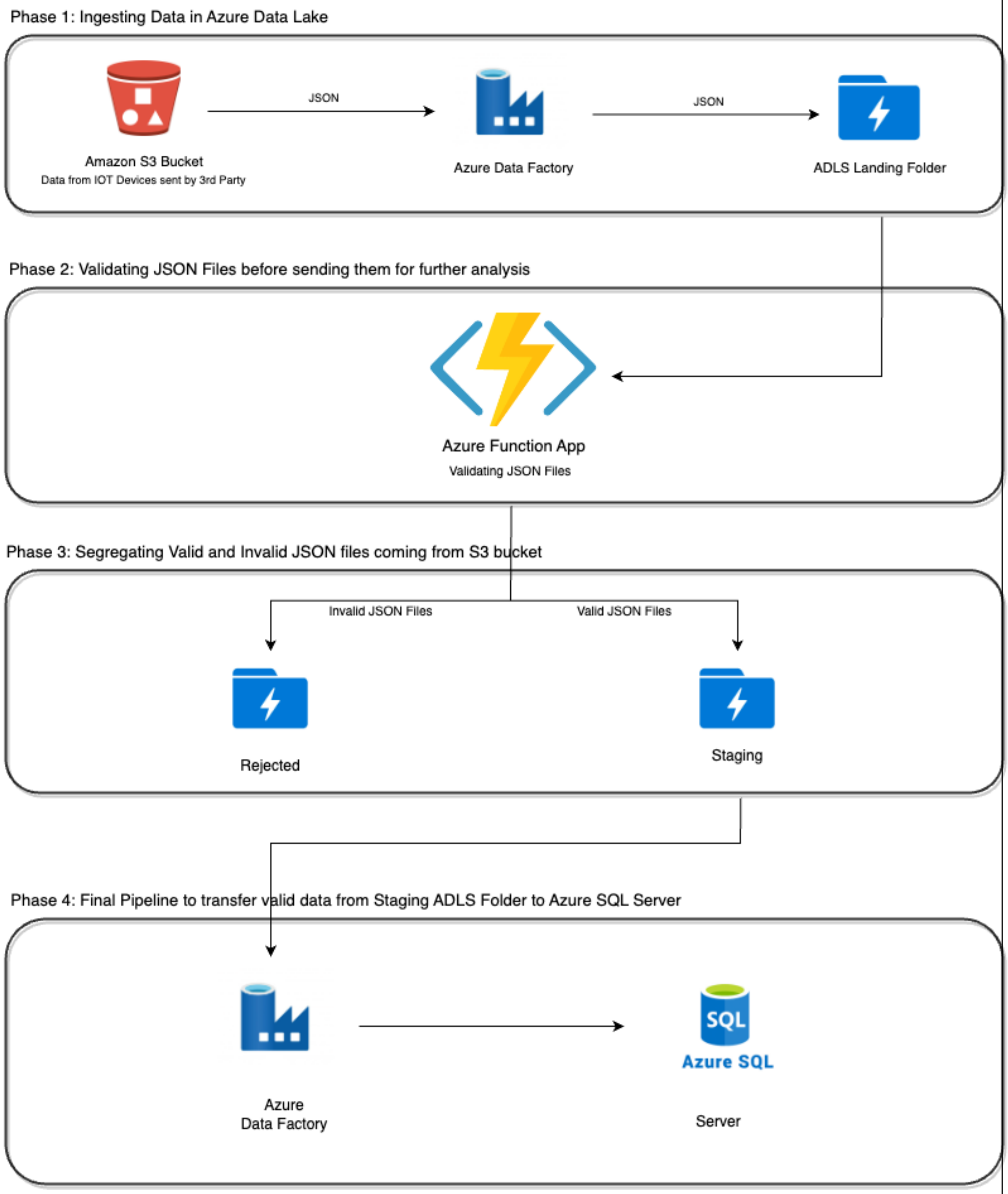
6. **Utilization: Data Science Team**

- The stored data in the SQL database serves as a crucial resource for Data Science team.
- Data analysts and scientists utilize this data for various analyses, insights, and decision-making processes.

Benefits:

- Seamless integration of telemetry data from IoT devices to Customer's Azure cloud.
- Ensured data integrity through validation, minimizing the risk of incorrect or incomplete data.
- Centralized storage of telemetry data in a SQL database for easy access and analysis.
- Empowering the Data Science team with valuable data for informed decision-making.

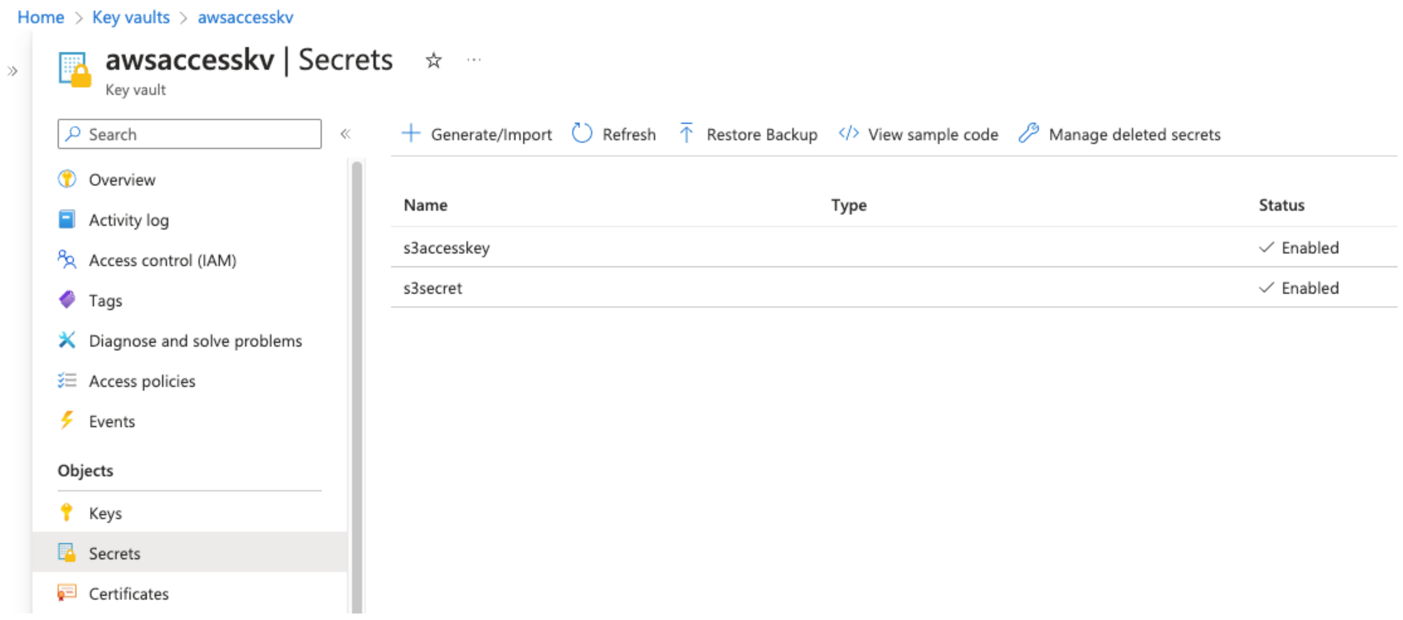
End to End Data Migration Architecture


Phase 1: Ingesting Data in Azure Data Lake
Amazon S3 Bucket
Data from IOT Devices sent by 3rd Party
JSON
Azure Data Factory
JSON
ADLS Landing Folder
Phase 2: Validating JSON Files before sending them for further analysis
Azure Function App
Validating JSON Files
Phase 3: Segregating Valid and Invalid JSON files coming from S3 bucket
Invalid JSON Files
Valid JSON Files
Rejected
Staging
Phase 4: Final Pipeline to transfer valid data from Staging ADLS Folder to Azure SQL Server
Azure
Data Factory
SQL
Azure SQL
Server

Phase 1

Step 1: Creating Linked Service to Connect to Amazon S3 bucket in Azure Data Factory

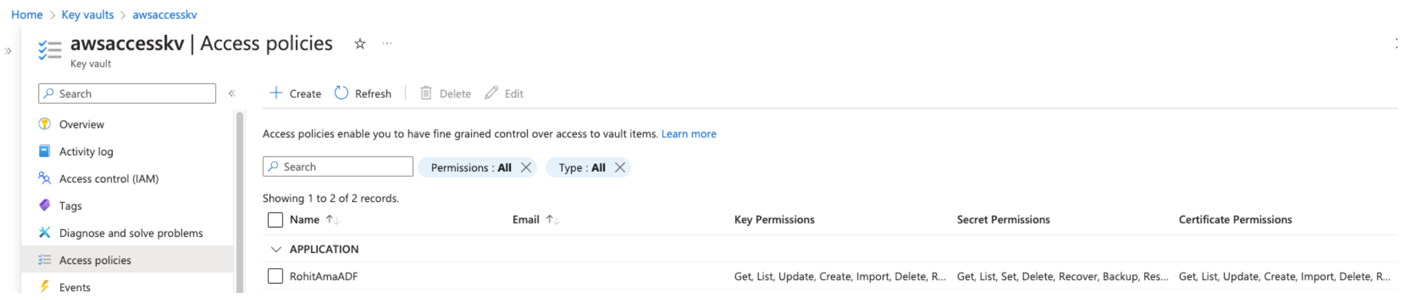
1. Getting S3 Access key ID and Access Secret from Amazon Bucket and Storing them in Key Vault


Home > Key vaults > awsaccesskv

awsaccesskv | Secrets
Key vault

Name	Type	Status
s3accesskey		✓ Enabled
s3secret		✓ Enabled

2. Giving Key Vault Access to ADF


Home > Key vaults > awsaccesskv

awsaccesskv | Access policies
Key vault

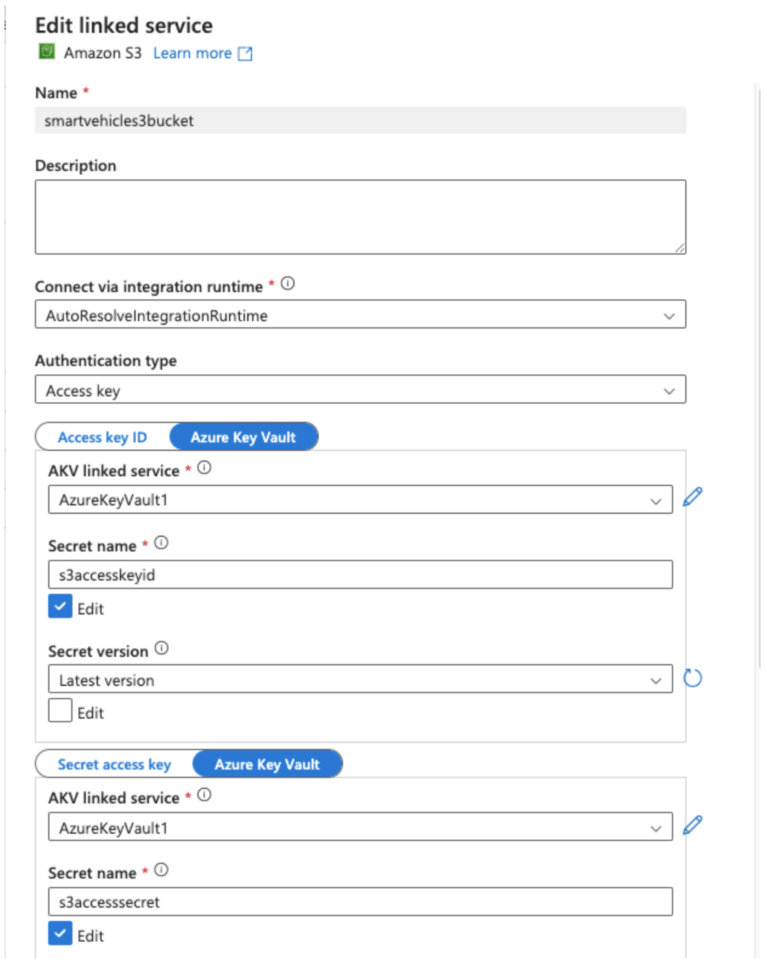
Access policies enable you to have fine grained control over access to vault items. Learn more

Search | Permissions : All | Type : All

Showing 1 to 2 of 2 records.

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions
APPLICATION				
RohitAmaADF		Get, List, Update, Create, Import, Delete, R...	Get, List, Set, Delete, Recover, Backup, Res...	Get, List, Update, Create, Import, Delete, R...

3. Using these credentials while creating the Linked Service for Source (S3 Bucket)



Edit linked service

Amazon S3 Learn more

Name *

smartvehicles3bucket

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Access key

Access key ID | Azure Key Vault

AKV linked service *

AzureKeyVault1

Secret name *

s3accesskeyid

☑ Edit

Secret version

Latest version

☐ Edit

Secret access key | Azure Key Vault

AKV linked service *

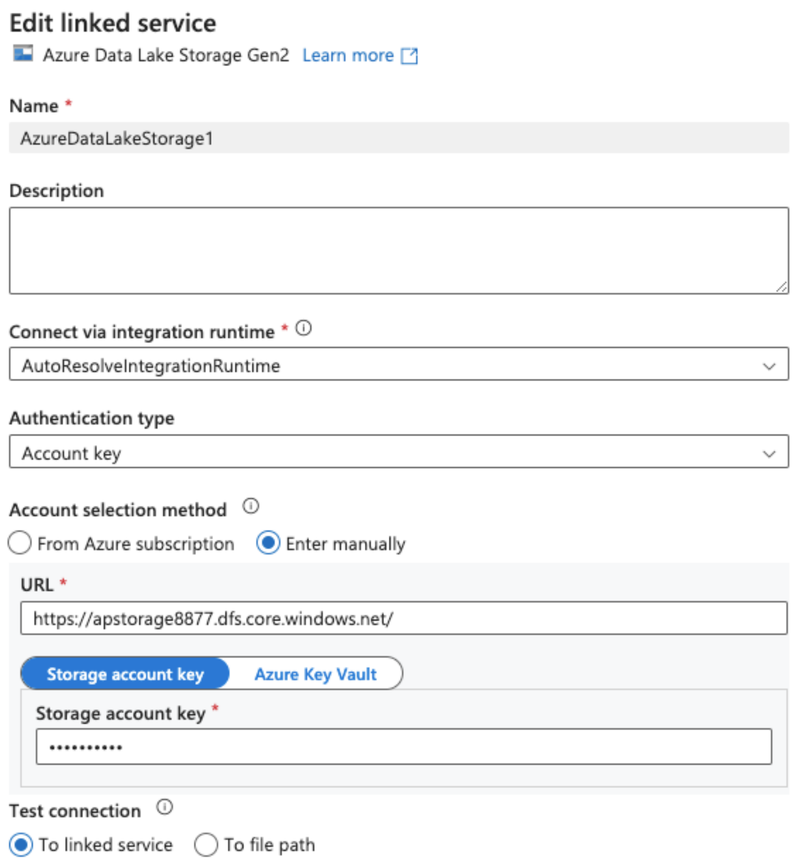
AzureKeyVault1

Secret name *

s3accesssecret

☑ Edit

4. Creating Linked Service for Sink (ADLS)



Edit linked service

Azure Data Lake Storage Gen2 Learn more

Name *

AzureDataLakeStorage1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method

From Azure subscription [✓ Enter manually]

URL *

https://apstorage8877.dfs.core.windows.net/

Storage account key | Azure Key Vault

Storage account key *

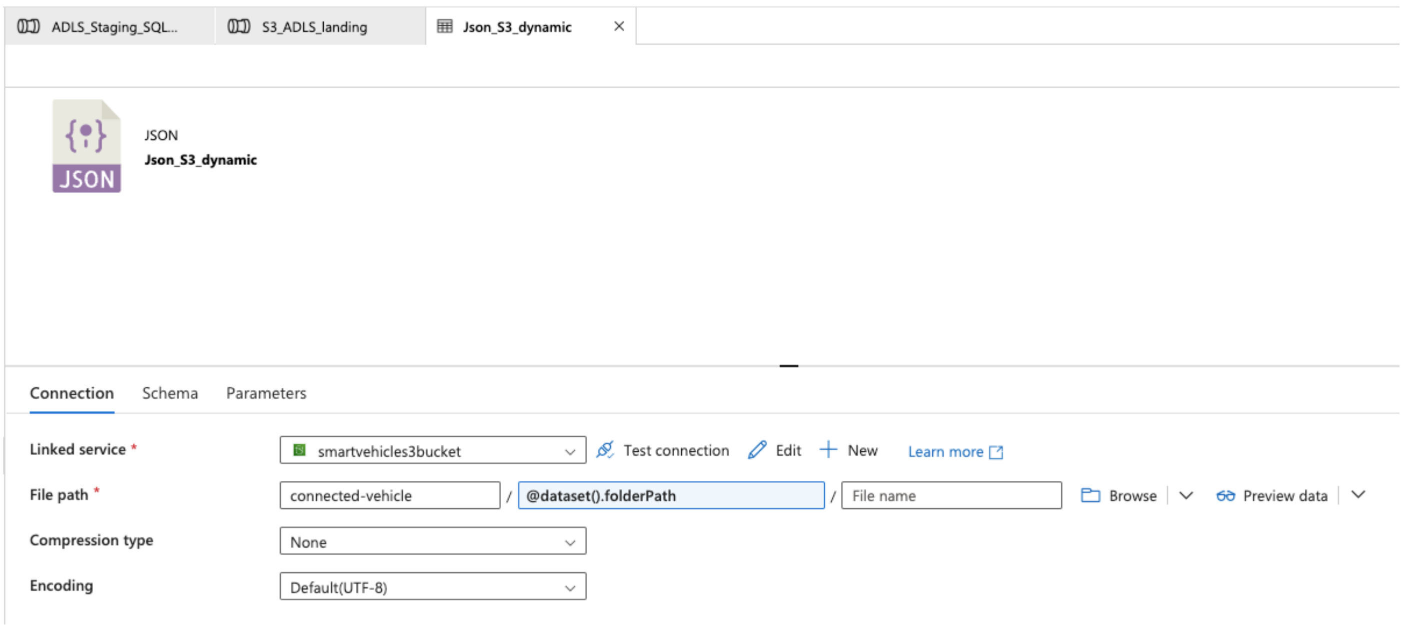
••••••••••

Test connection

[✓ To linked service] To file path

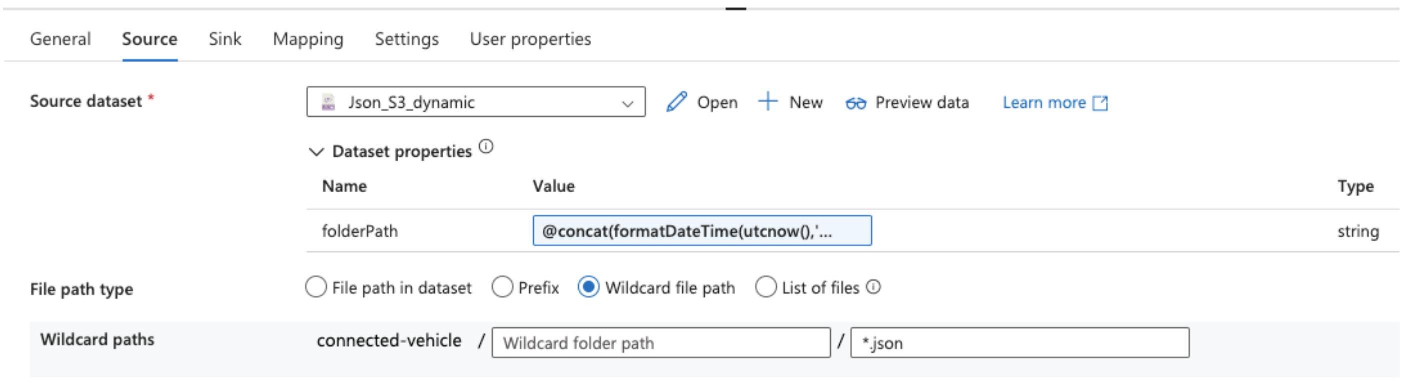
Step 2: Creating Pipeline to ingest Data from S3 Bucket to ADLS Landing Folder

1. Creating Datasets for Source in Copy Activity



Linked service *	smartvehicles3bucket
File path *	connected-vehicle / @dataset().folderPath / File name
Compression type	None
Encoding	Default(UTF-8)

2. Source Settings – We keep a wildcard file path because we want it to fetch daily data which is JSON format because we will be running this pipeline on a daily basis



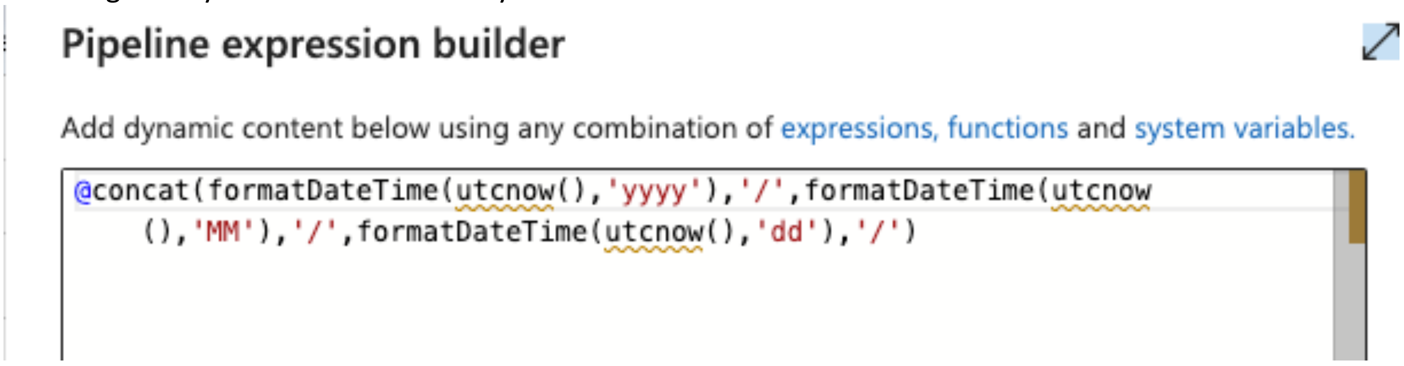
Source dataset * Json_S3_dynamic

Name	Value	Type
folderPath	@concat(formatDateTime(utcnow(),'...	string

File path type: ( ) File path in dataset ( ) Prefix [✓ Wildcard file path] ( ) List of files

Wildcard paths: connected-vehicle / Wildcard folder path / *.json

We use the following dynamic folder path because of the way the folders are structured at the Source side and we get daily data in a different day's folder –

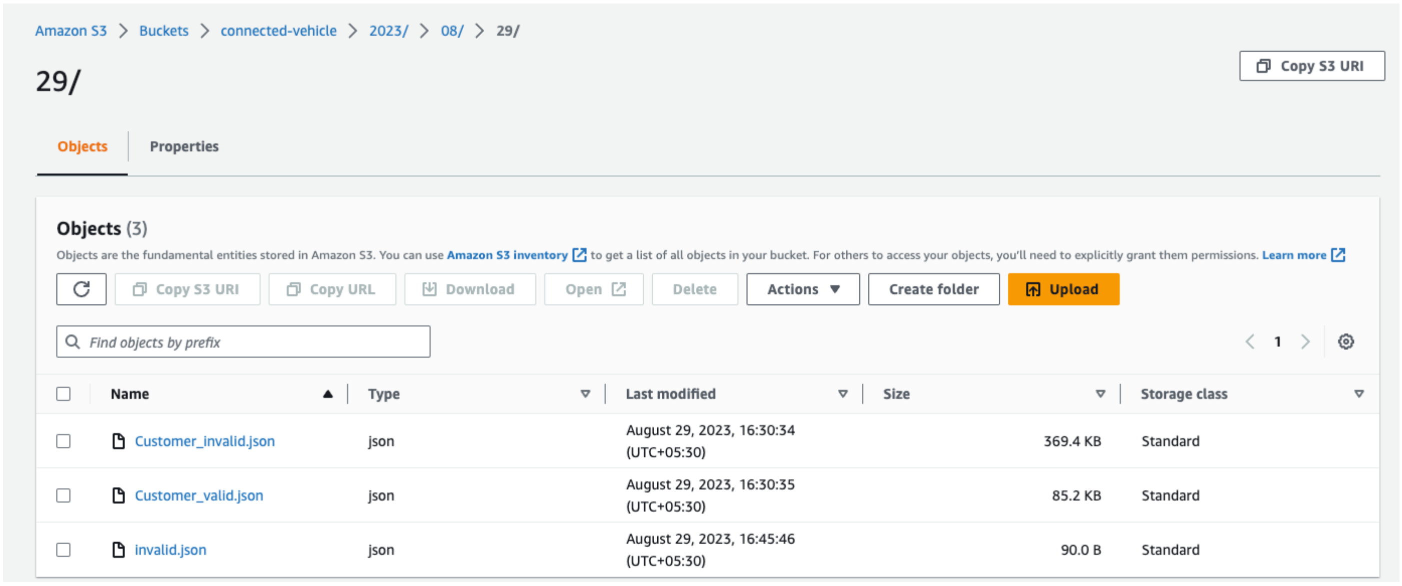


Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat(formatDateTime(utcnow(),'yyyy'),'/',formatDateTime(utcnow
    (),'MM'),'/',formatDateTime(utcnow(),'dd'),'/')
```

Source side Bucket's folder structure for reference –


Amazon S3 > Buckets > connected-vehicle > 2023/ > 08/ > 29/

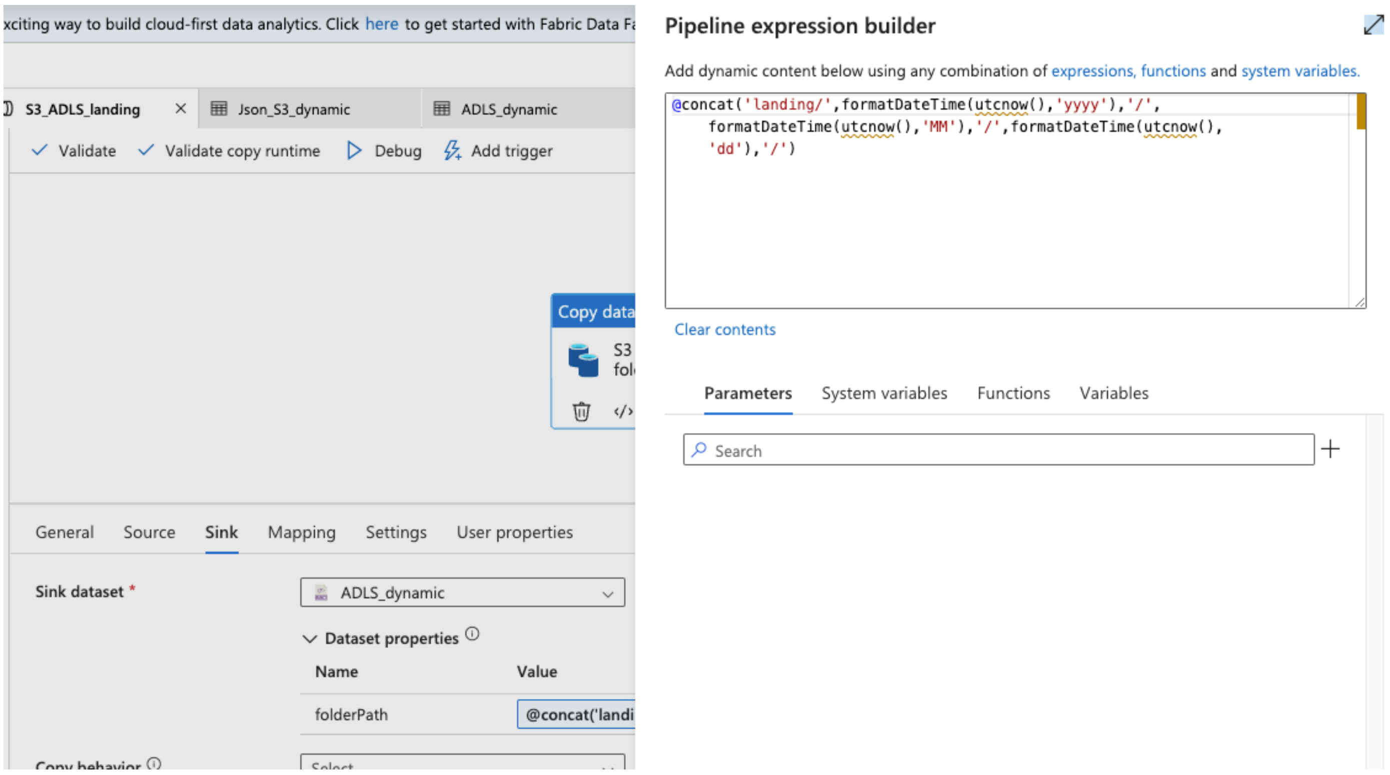
29/

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Customer_invalid.json	json	August 29, 2023, 16:30:34 (UTC+05:30)	369.4 KB	Standard
Customer_valid.json	json	August 29, 2023, 16:30:35 (UTC+05:30)	85.2 KB	Standard
invalid.json	json	August 29, 2023, 16:45:46 (UTC+05:30)	90.0 B	Standard

3. Creating Sink Dataset



Sink dataset * ADLS_dynamic

Dataset properties

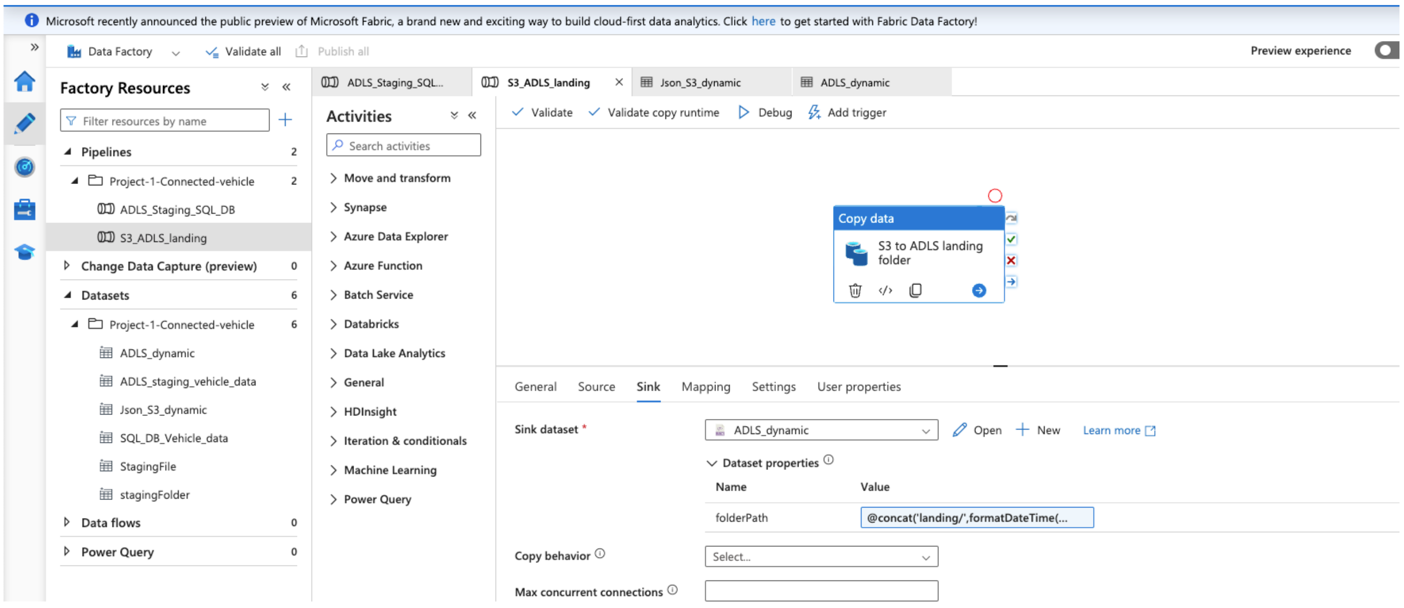
Name	Value
folderPath	@concat('landi

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat('landing/',formatDateTime(utcnow(),'yyyy'),'/',
    formatDateTime(utcnow(),'MM'),'/',formatDateTime(utcnow(),
    'dd'),'/')
```

We follow the same structure here as we want the files to be stored in ADLS in the same format as they are stored in S3 bucket

Step 3: Run the pipeline and check the ADLS Landing Folder


Microsoft recently announced the public preview of Microsoft Fabric, a brand new and exciting way to build cloud-first data analytics. Click here to get started with Fabric Data Factory!

Sink dataset * ADLS_dynamic

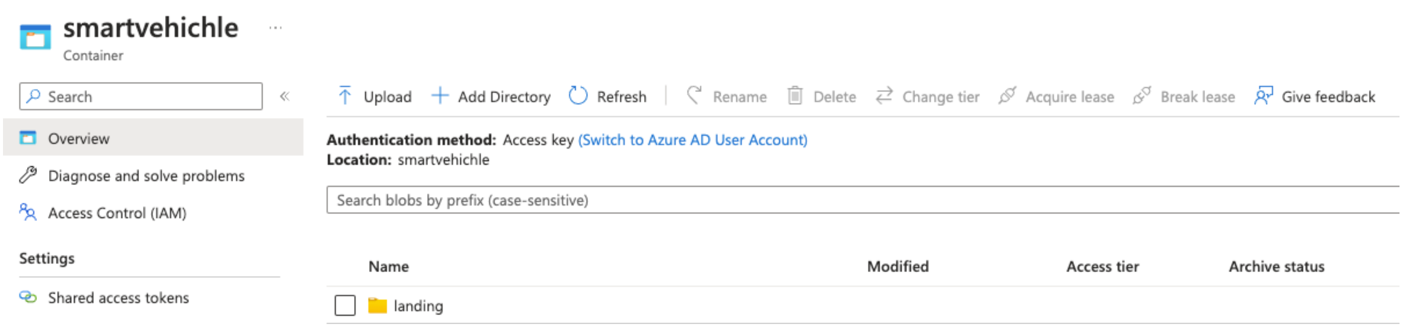
Dataset properties

Name	Value
folderPath	@concat('landing/',formatDateTime(...

Copy behavior: Select...

Max concurrent connections:

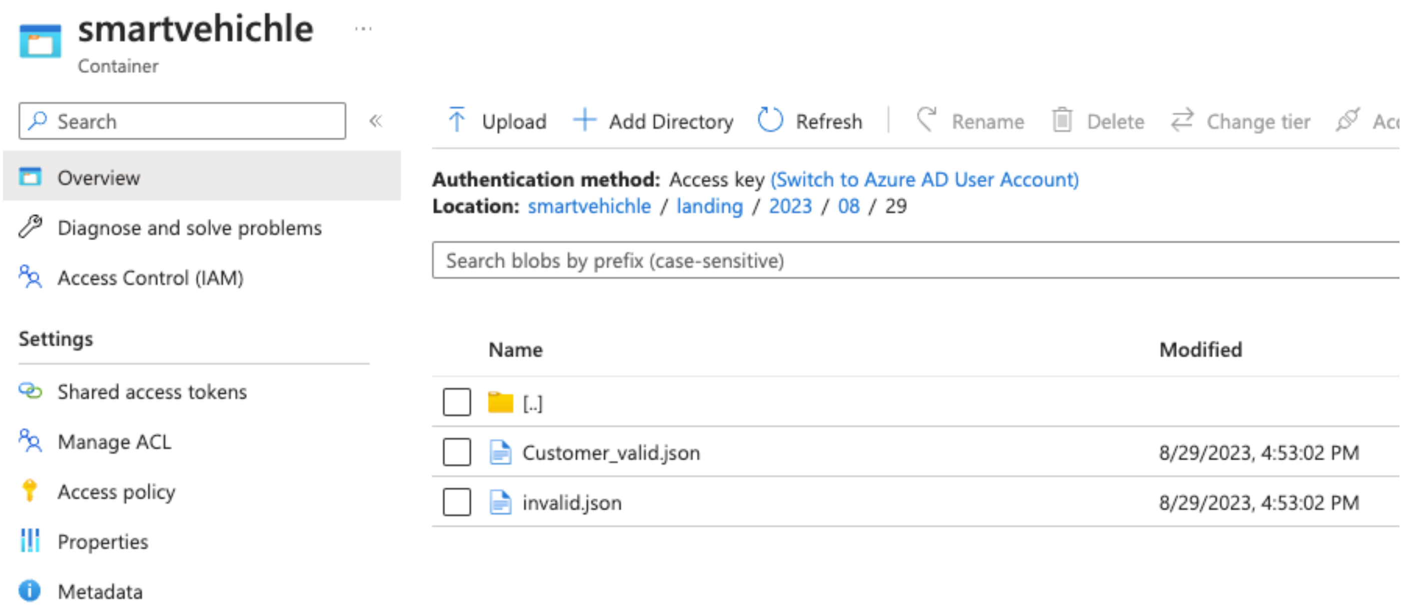
ADLS Container:



Authentication method: Access key (Switch to Azure AD User Account)
Location: smartvehichle

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status
landing			



Authentication method: Access key (Switch to Azure AD User Account)
Location: smartvehichle / landing / 2023 / 08 / 29

Search blobs by prefix (case-sensitive)

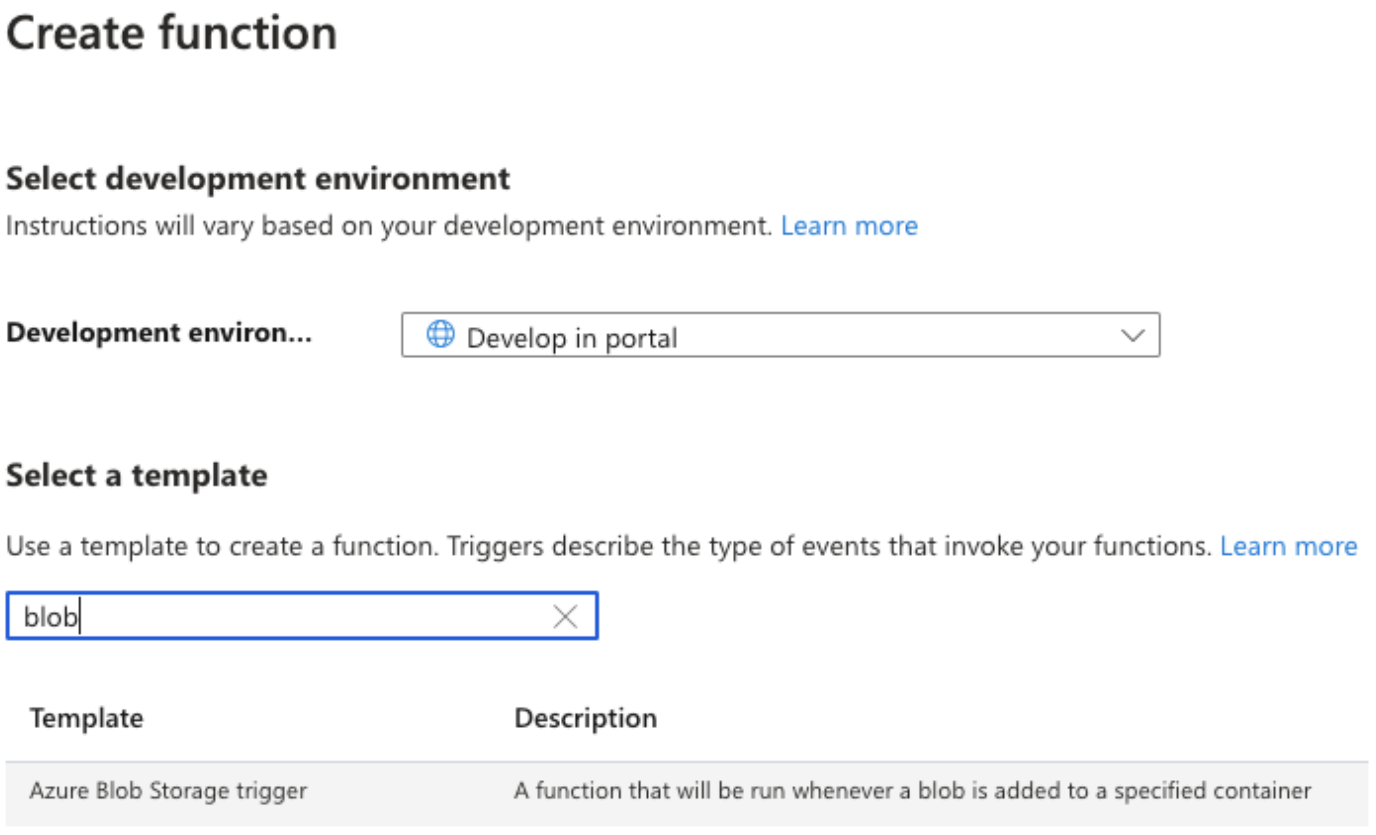
Name	Modified
[..]	
Customer_valid.json	8/29/2023, 4:53:02 PM
invalid.json	8/29/2023, 4:53:02 PM

Step 5: We can add a trigger to make this pipeline run daily

Phase 2: Validating the JSON files coming in the Landing Folder from S3 Bucket

Step 1: Creating an Azure Function to validate the JSON file

1. Create a Function App and create a Blob Storage Trigger function inside it



Create function

Select development environment

Instructions will vary based on your development environment. Learn more

Development environ... Develop in portal

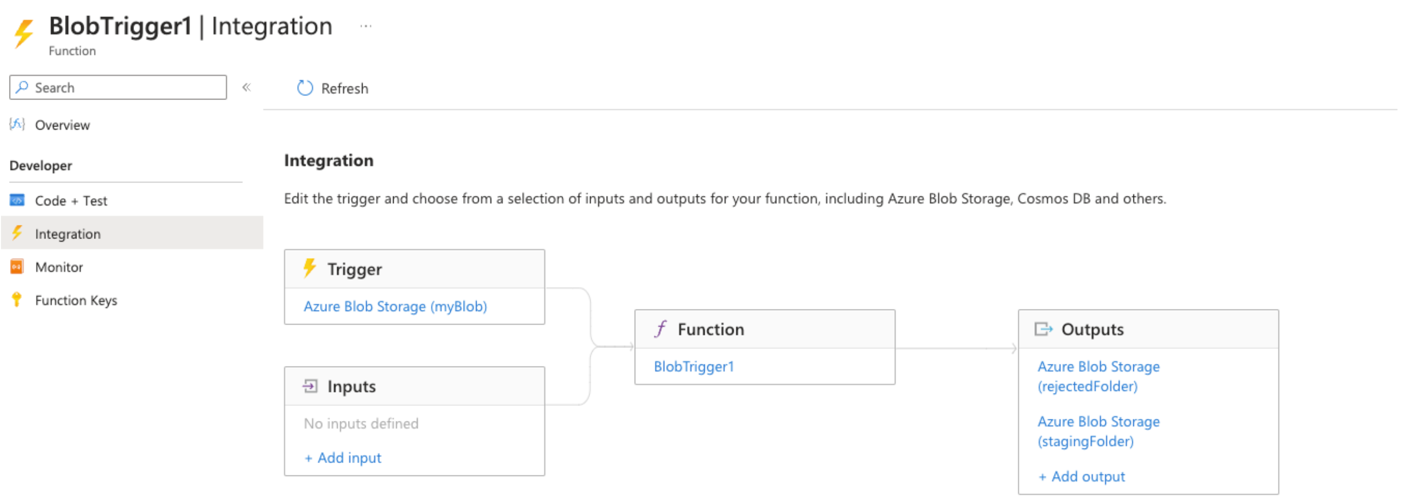
Select a template

Use a template to create a function. Triggers describe the type of events that invoke your functions. Learn more

blob

Template	Description
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container

2. Setup Trigger in Integration Tab



BlobTrigger1 | Integration

Function

Integration

Edit the trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.

Trigger: Azure Blob Storage (myBlob)

Inputs: No inputs defined

+ Add input

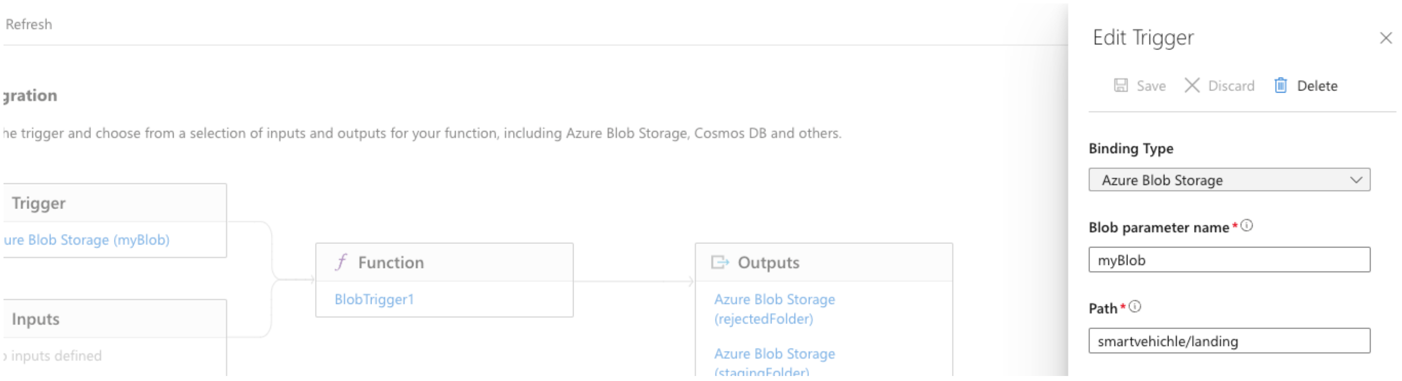
Function: BlobTrigger1

Outputs: Azure Blob Storage (rejectedFolder)

Azure Blob Storage (stagingFolder)

+ Add output

3. Setup Bindings for Input along with the storage account


Refresh

gration

he trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.

Trigger
ure Blob Storage (myBlob)

Inputs
› inputs defined

Function
BlobTrigger1

Outputs
Azure Blob Storage (rejectedFolder)
Azure Blob Storage (stagingFolder)

Edit Trigger

Save | Discard | Delete

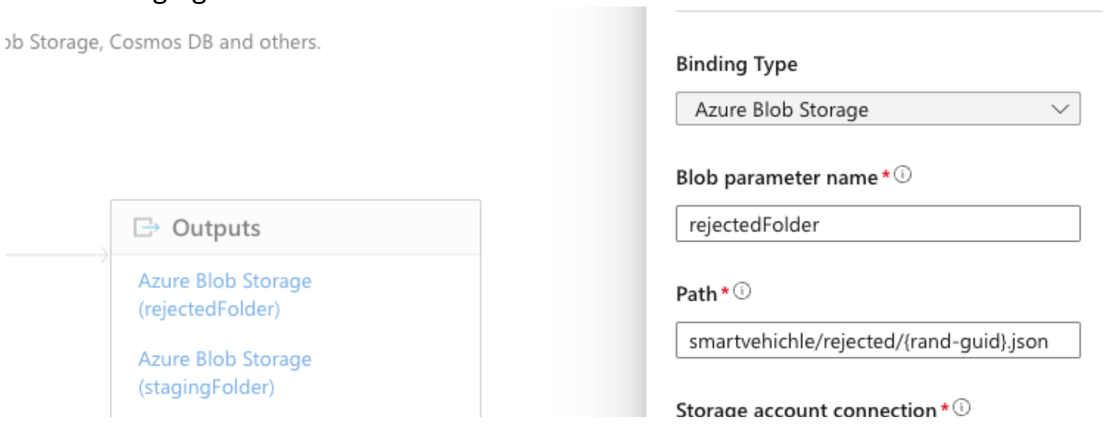
Binding Type: Azure Blob Storage

Blob parameter name*: myBlob

Path*: smartvehichle/landing

4. Setup Bindings for Output along with the storage account

a. One for Staging which will have valid JSON files


›b Storage, Cosmos DB and others.

Outputs
Azure Blob Storage (rejectedFolder)
Azure Blob Storage (stagingFolder)

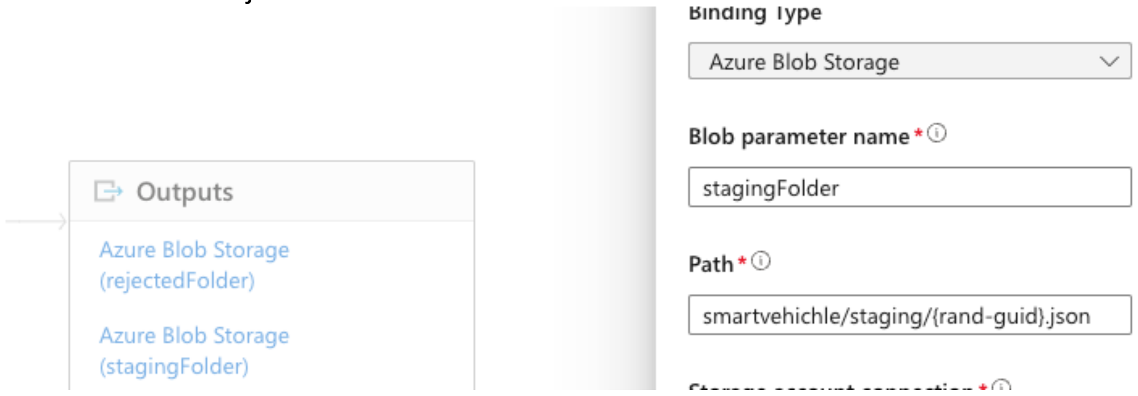
Binding Type: Azure Blob Storage

Blob parameter name*: rejectedFolder

Path*: smartvehichle/rejected/{rand-guid}.json

Storage account connection*

b. Another for Rejected for invalid JSON files


Binding Type: Azure Blob Storage

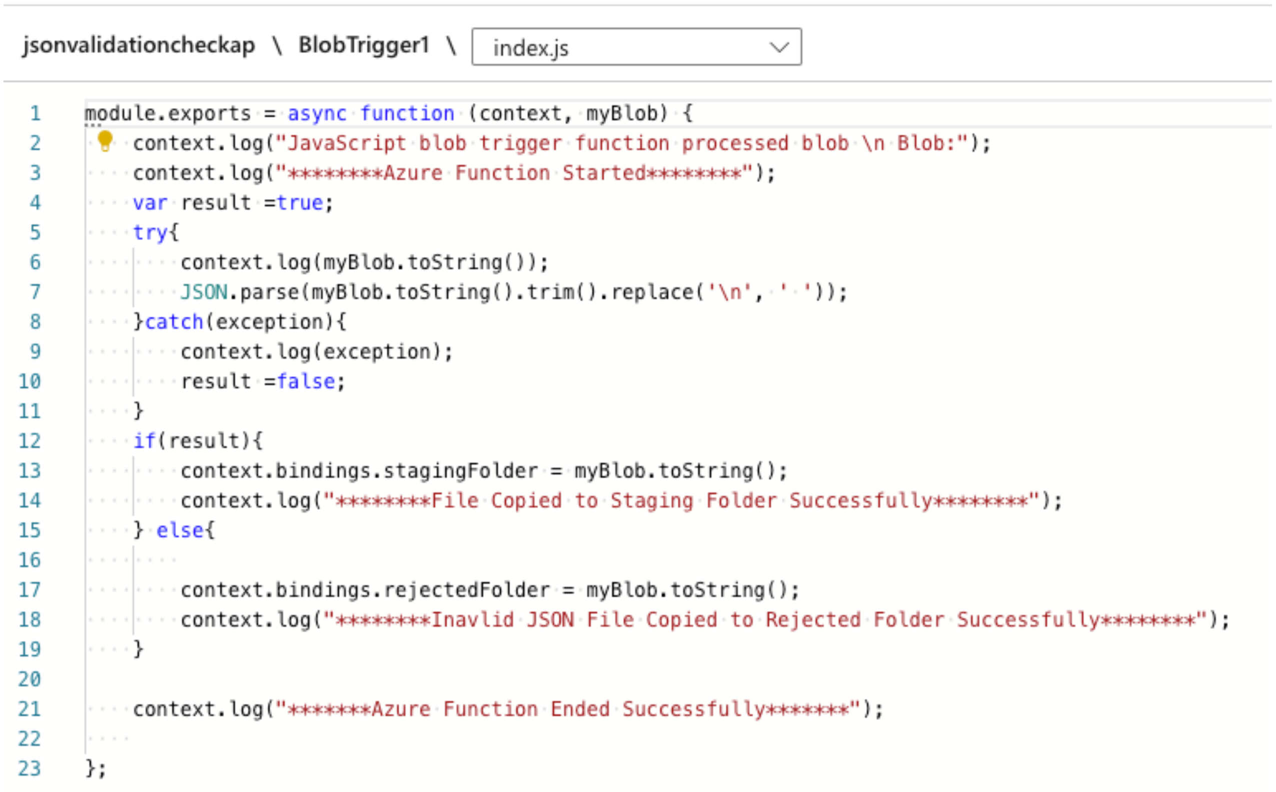
Blob parameter name*: stagingFolder

Path*: smartvehichle/staging/{rand-guid}.json

Outputs
Azure Blob Storage (rejectedFolder)
Azure Blob Storage (stagingFolder)

5. Writing the code to check if any error given while parsing, send the file to rejected folder else send it to Staging Folder which we will use for final migration

Code Snippet –



jsonvalidationcheckap \ BlobTrigger1 \ index.js

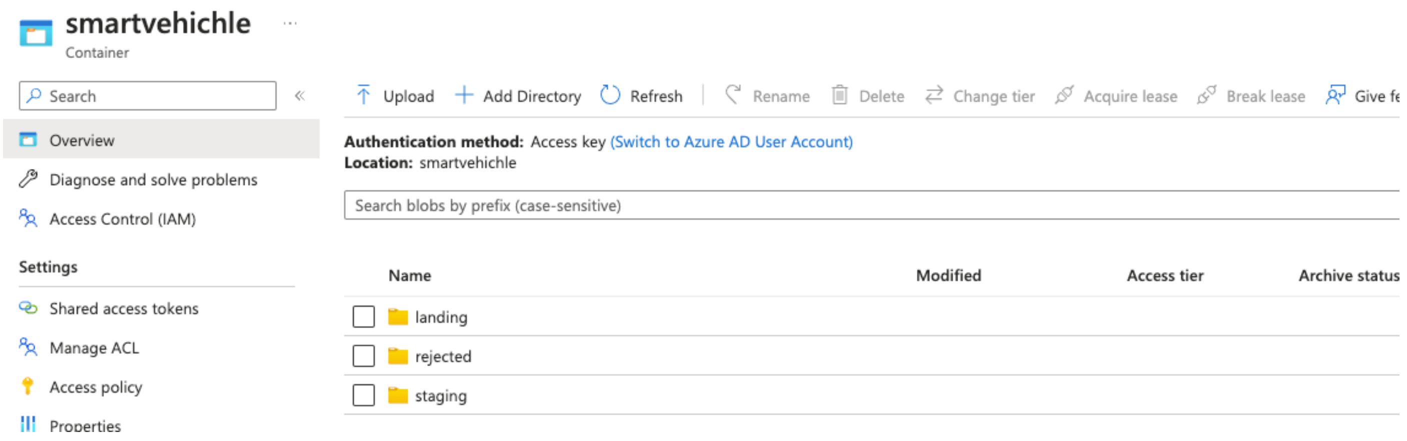
```
module.exports = async function (context, myBlob) {
    context.log("JavaScript blob trigger function processed blob \n Blob:");
    context.log("*******Azure Function Started*******");
    var result =true;
    try{
        context.log(myBlob.toString());
        JSON.parse(myBlob.toString().trim().replace('\n', ' '));
    }catch(exception){
        context.log(exception);
        result =false;
    }
    if(result){
        context.bindings.stagingFolder = myBlob.toString();
        context.log("********File Copied to Staging Folder Successfully********");
    } else{
        
        context.bindings.rejectedFolder = myBlob.toString();
        context.log("********Inavlid JSON File Copied to Rejected Folder Successfully********");
    }

    context.log("*******Azure Function Ended Successfully*******");
    
};
```

Once we get a new file in Landing Folder, this app will trigger and the files will be segregated based on their validity check

Phase 3: Data Segregation and Validation

Here we can see 2 folders being created having 1 file each –

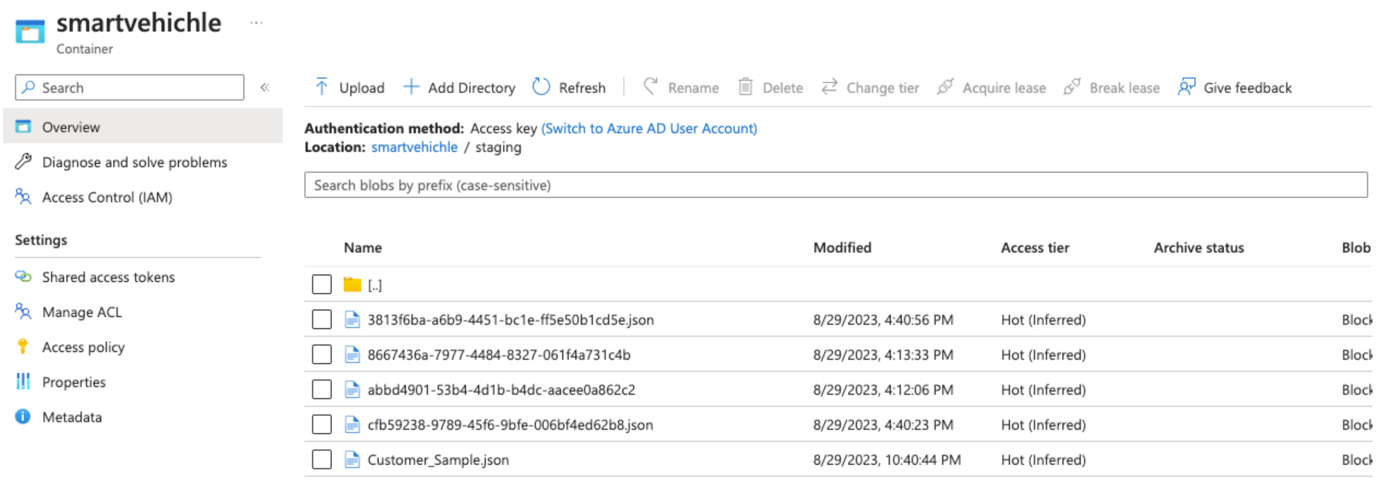


smartvehichle
Container

Authentication method: Access key (Switch to Azure AD User Account)
Location: smartvehichle

Name	Modified	Access tier	Archive status
landing			
rejected			
staging			

Staging –

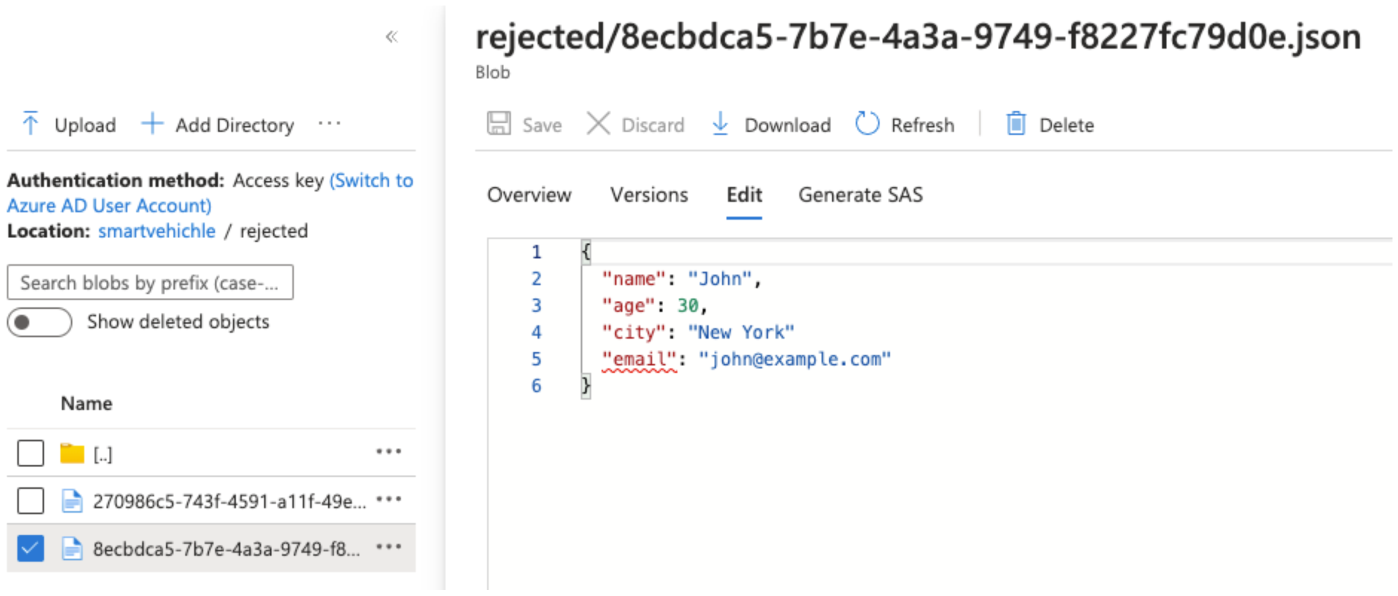


Authentication method: Access key (Switch to Azure AD User Account)
Location: smartvehichle / staging

Name	Modified	Access tier	Archive status	Blob
[..]				
3813f6ba-a6b9-4451-bc1e-ff5e50b1cd5e.json	8/29/2023, 4:40:56 PM	Hot (Inferred)		Block
8667436a-7977-4484-8327-061f4a731c4b	8/29/2023, 4:13:33 PM	Hot (Inferred)		Block
abbd4901-53b4-4d1b-b4dc-aacee0a862c2	8/29/2023, 4:12:06 PM	Hot (Inferred)		Block
cfb59238-9789-45f6-9bfe-006bf4ed62b8.json	8/29/2023, 4:40:23 PM	Hot (Inferred)		Block
Customer_Sample.json	8/29/2023, 10:40:44 PM	Hot (Inferred)		Block

Rejected –

The function works because we purposely added an invalid JSON format file and it shows up in the rejected folder as seen below –



Authentication method: Access key (Switch to Azure AD User Account)
Location: smartvehichle / rejected

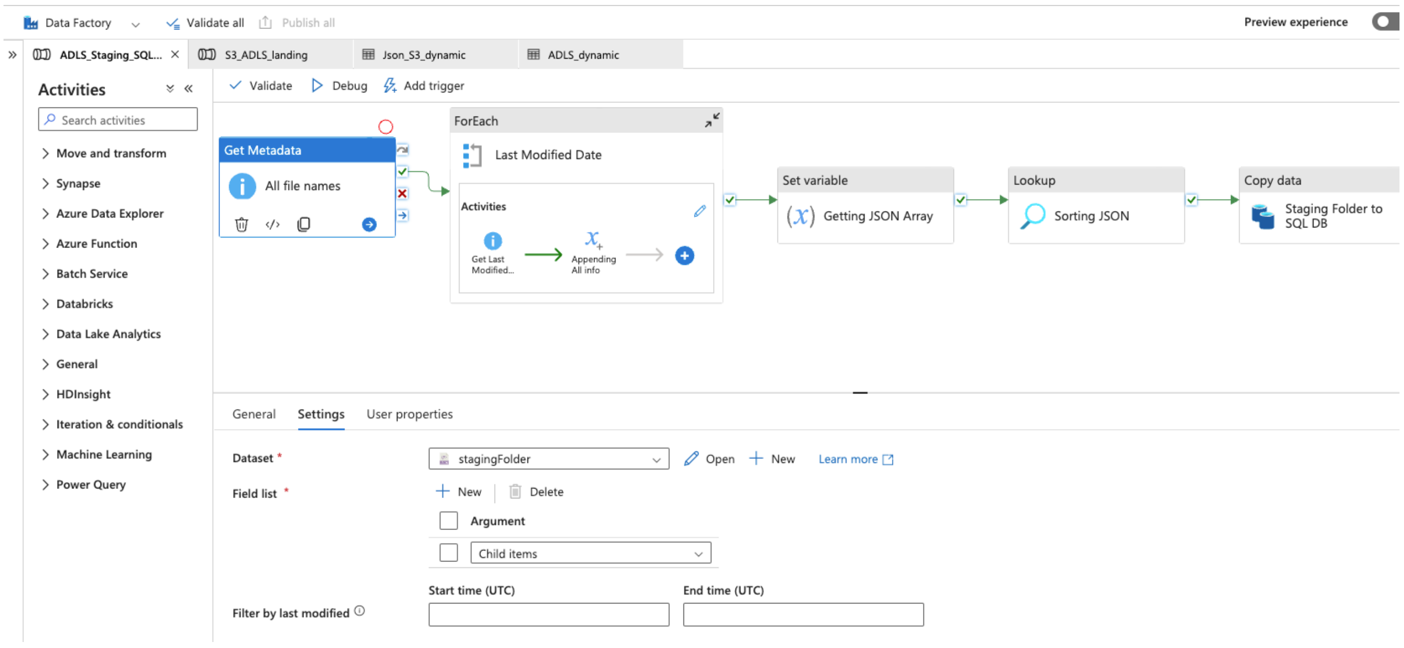
Name
[..]
270986c5-743f-4591-a11f-49e...
8ecbdca5-7b7e-4a3a-9749-f8...

```
{
 "name": "John",
 "age": 30,
 "city": "New York"
 "email": "john@example.com"
}
```

Phase 4: Final Pipeline to send files from Staging Folder to Azure SQL Server

This is the crucial part of the pipeline as we will be getting files on a daily basis in the Staging Folder and each time we get a new file in this folder, we would want our pipeline to take the most recent file and send it to SQL Server.

Step 1: Take all the filenames from the Staging Folder

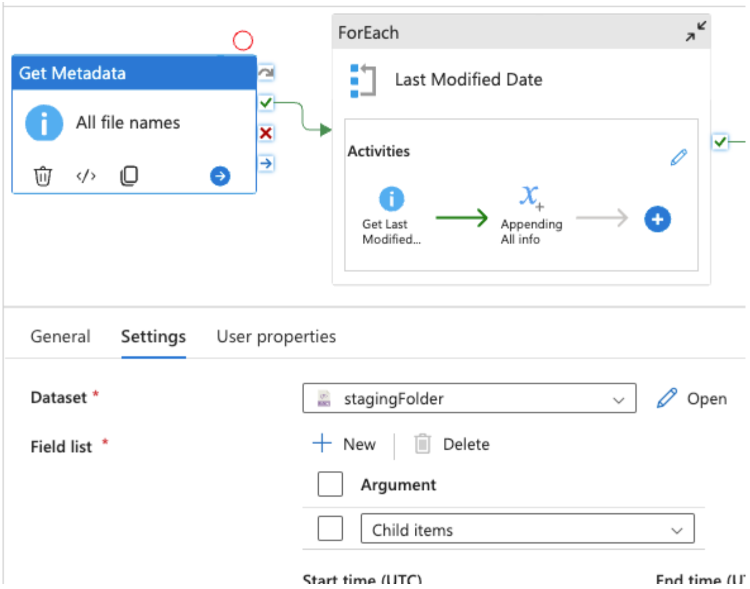


Dataset * stagingFolder

Field list * Argument: Child items

Filter by last modified — Start time (UTC): End time (UTC):

Get Metadata –

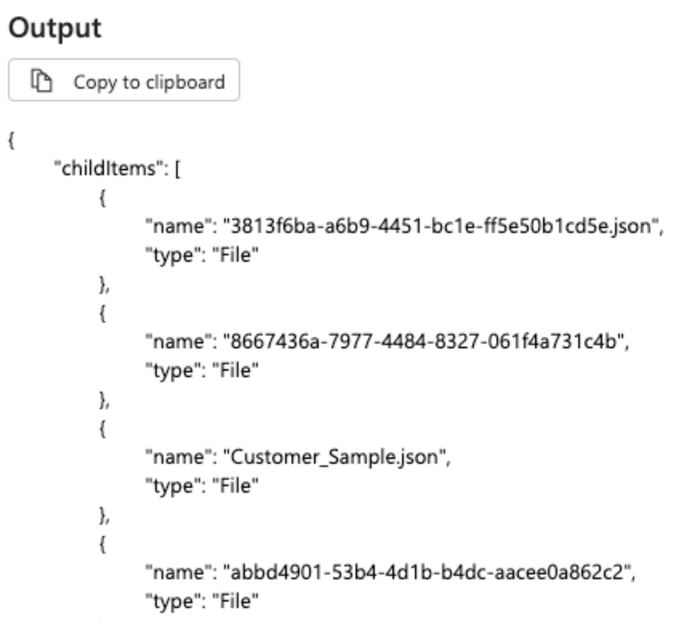


Dataset * stagingFolder

Field list * Argument: Child items

Start time (UTC) End time (U

2. Now that we have the filenames, we can get the Last Modified date for each file

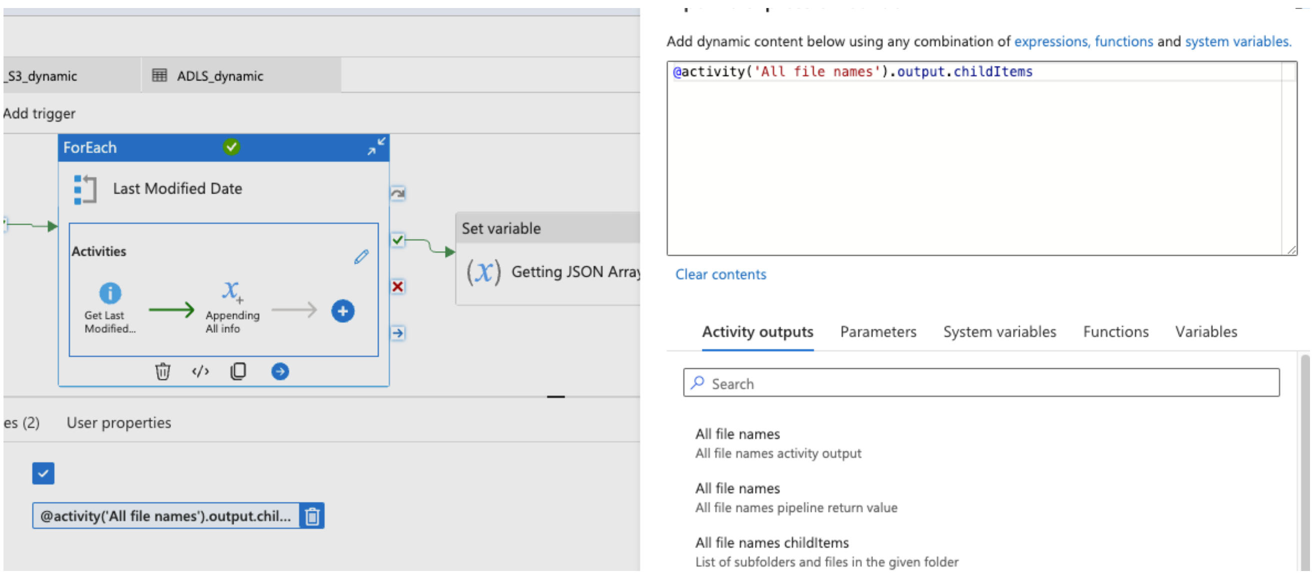


Output

Copy to clipboard

```
{
    "childItems": [
        {
            "name": "3813f6ba-a6b9-4451-bc1e-ff5e50b1cd5e.json",
            "type": "File"
        },
        {
            "name": "8667436a-7977-4484-8327-061f4a731c4b",
            "type": "File"
        },
        {
            "name": "Customer_Sample.json",
            "type": "File"
        },
        {
            "name": "abbd4901-53b4-4d1b-b4dc-aacee0a862c2",
            "type": "File"
        }
```

3. Applying For each activity and adding a Get Metadata activity inside for getting the Last Modified Date for each file



@activity('All file names').output.chil...

Add dynamic content below using any combination of expressions, functions and system variables.

```
@activity('All file names').output.childItems
```

Clear contents

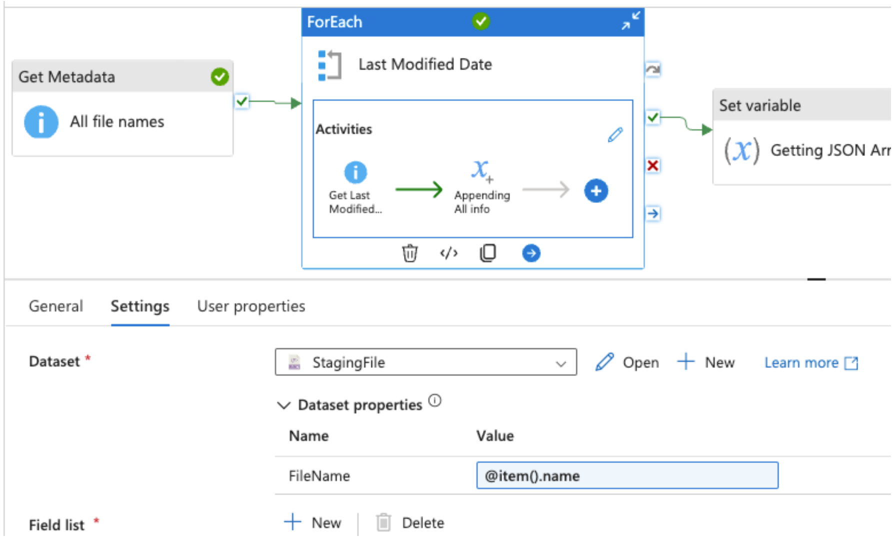
Activity outputs | Parameters | System variables | Functions | Variables

Search

All file names
All file names activity output

All file names
All file names pipeline return value

All file names childItems
List of subfolders and files in the given folder



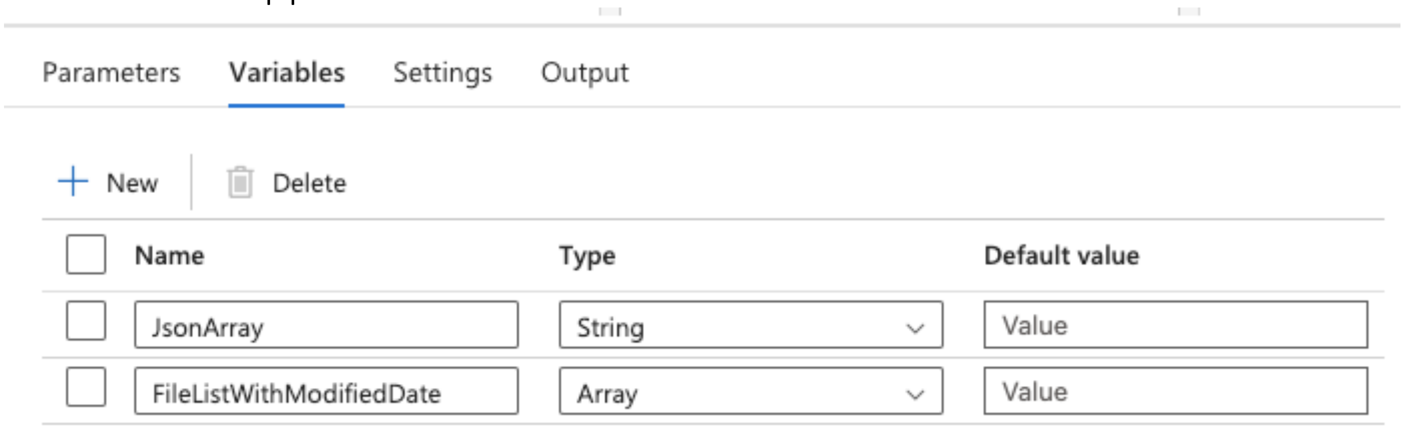
Dataset * StagingFile — Open + New Learn more

Dataset properties

Name	Value
FileName	@item().name

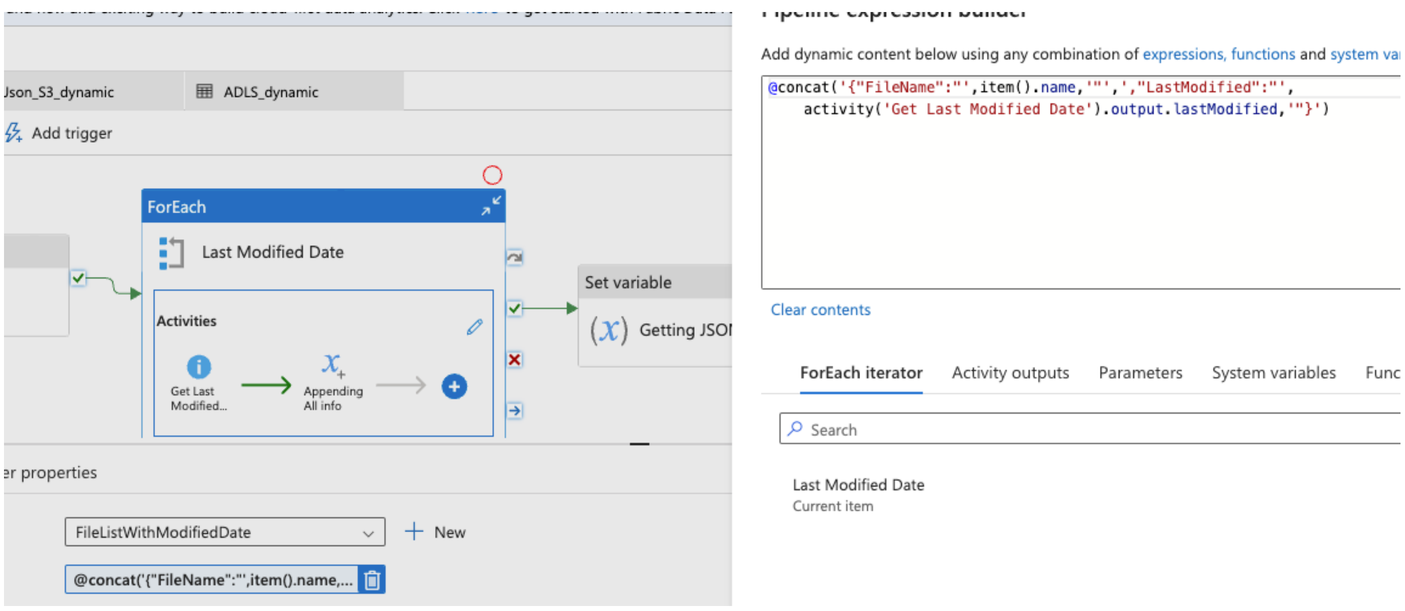
Field list * + New Delete

4. We define 2 pipeline variables



Name	Type	Default value
JsonArray	String	Value
FileListWithModifiedDate	Array	Value

5. We use the array variable above to append our outputs from both Get Metadata Activity by using Append Variable Activity



Add dynamic content below using any combination of expressions, functions and system va

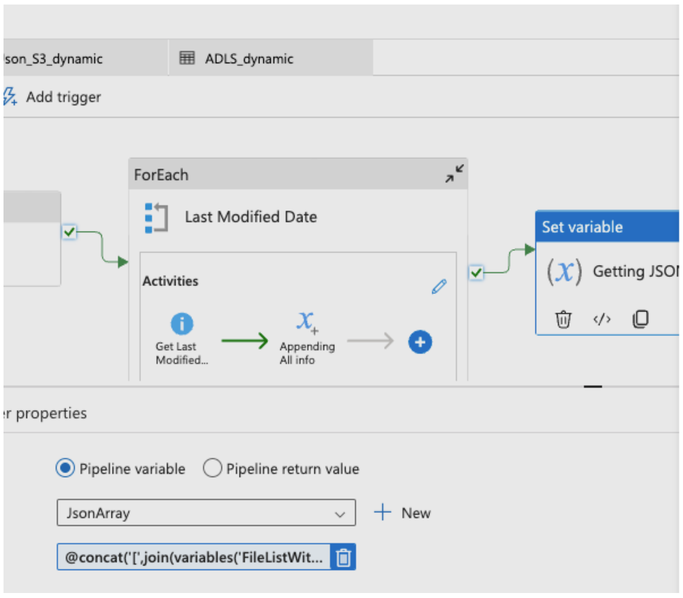
```
@concat('{"FileName":"',item().name,'"',',"LastModified":"',
    activity('Get Last Modified Date').output.lastModified,'"}')
```

@concat('{"FileName":"',item().name,...

Output would look like –

{\"FileName\":\"3813f6ba-a6b9-4451-bc1e-ff5e50b1cd5e.json\",\"LastModified\":\"2023-08-29T11:10:56+00:00\"}
{\"FileName\":\"8667436a-7977-4484-8327-061f4a731c4b\",\"LastModified\":\"2023-08-29T10:43:33+00:00\"}
{\"FileName\":\"Customer_Sample.json\",\"LastModified\":\"2023-08-29T17:10:44+00:00\"}
{\"FileName\":\"abbd4901-53b4-4d1b-b4dc-aacee0a862c2\",\"LastModified\":\"2023-08-29T10:42:06+00:00\"}
{\"FileName\":\"cfb59238-9789-45f6-9bfe-006bf4ed62b8.json\",\"LastModified\":\"2023-08-29T11:10:23+00:00\"}

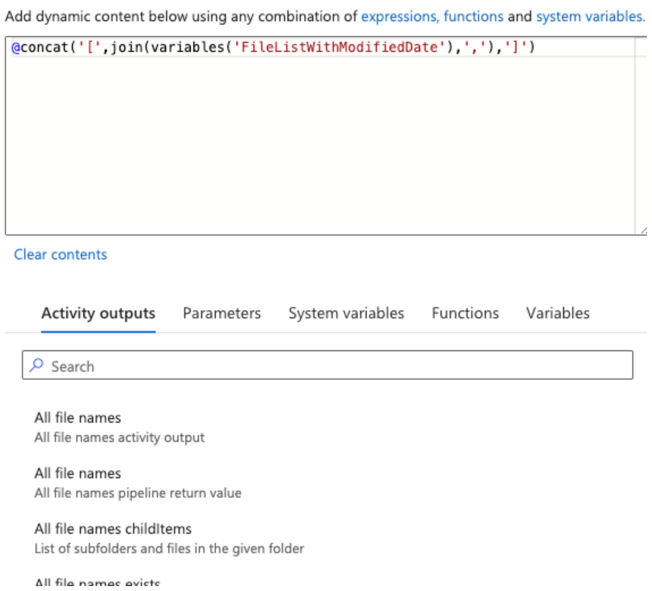
6. Using Set variable activity to assign the above JSON array with a name –



[✓ Pipeline variable] Pipeline return value

JsonArray + New

@concat('[',join(variables('FileListWit...



Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat('[',join(variables('FileListWithModifiedDate'),','),']')
```

All file names
All file names activity output

All file names
All file names pipeline return value

All file names childItems
List of subfolders and files in the given folder

All file names exists

Output –

```
{
	"name": "JsonArray",
	"value": "[{\"FileName\":\"3813f6ba-a6b9-4451-bc1e-ff5e50b1cd5e.json\",\"LastModified\":\"2023-08-29T11:10:56+00:00\"},{\"FileName\":\"8667436a-7977-4484-8327-061f4a731c4b\",\"LastModified\":\"2023-08-29T10:43:33+00:00\"},{\"FileName\":\"Customer_Sample.json\",\"LastModified\":\"2023-08-29T17:10:44+00:00\"},{\"FileName\":\"abbd4901-53b4-4d1b-b4dc-aacee0a862c2\",\"LastModified\":\"2023-08-29T10:42:06+00:00\"},{\"FileName\":\"cfb59238-9789-45f6-9bfe-006bf4ed62b8.json\",\"LastModified\":\"2023-08-29T11:10:23+00:00\"}]"
}
```

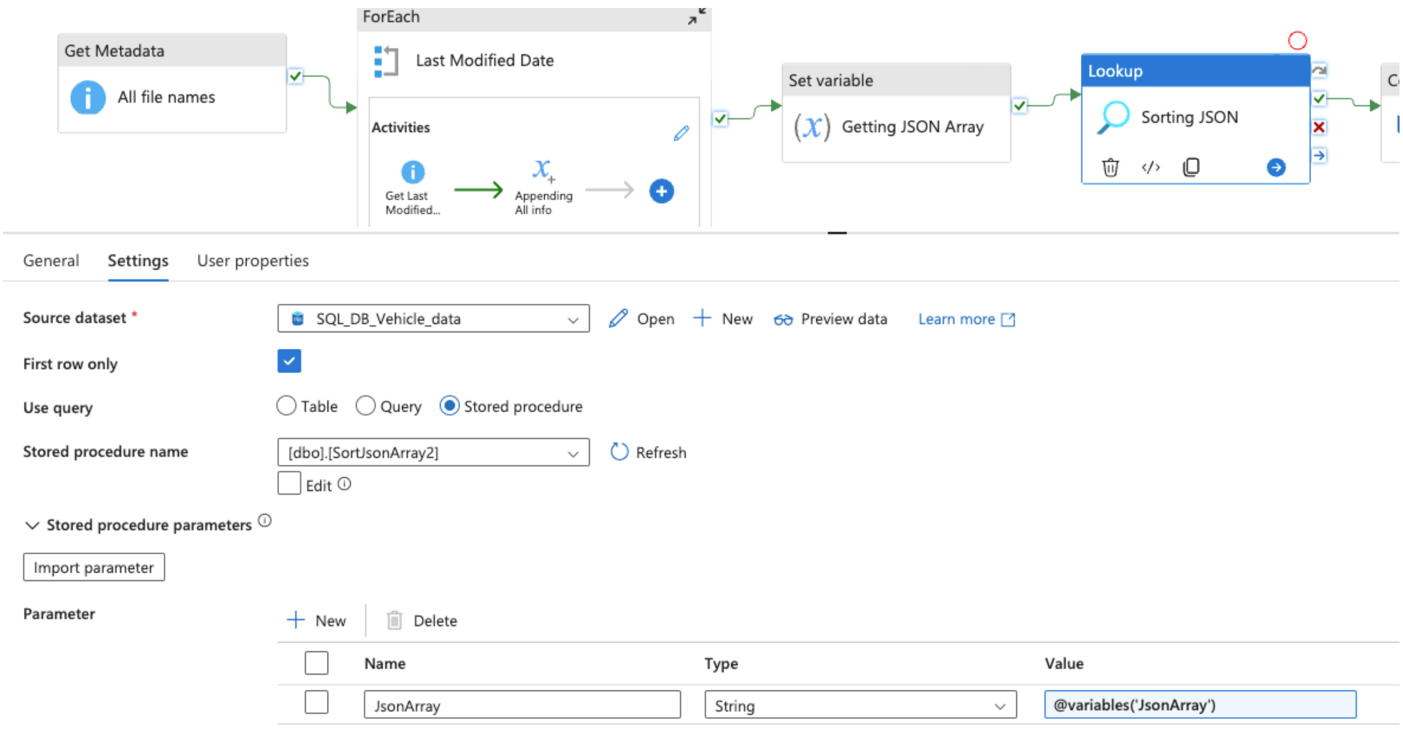
7. Creating a stored procedure in SQL server to sort this JSON array based on Last Modified Date

```sql
CREATE PROCEDURE SortJsonArray2
  @JsonArray NVARCHAR(MAX)
AS
BEGIN
  DECLARE @OutputJson NVARCHAR(MAX)

  SELECT @OutputJson = (
    SELECT FileName, LastModified
    FROM (
      SELECT FileName, CONVERT(DATETIME2, LastModified, 127) AS LastModified
      FROM OPENJSON(@JsonArray)
      WITH (
        FileName NVARCHAR(MAX),
        LastModified NVARCHAR(50) -- Use NVARCHAR to retain the original string
      )
    ) AS Subquery
    ORDER BY LastModified DESC
    FOR JSON PATH
  )

  SELECT @OutputJson AS OutputJson
END
```

8. Using Lookup Activity and the pipeline variable we created to pass in this Stored Procedure to get the latest file based on Last Modified Date



Source dataset * SQL_DB_Vehicle_data

First row only ✓

Use query: Table Query [✓ Stored procedure]

Stored procedure name: [dbo].[SortJsonArray2]

Edit

Stored procedure parameters

Import parameter

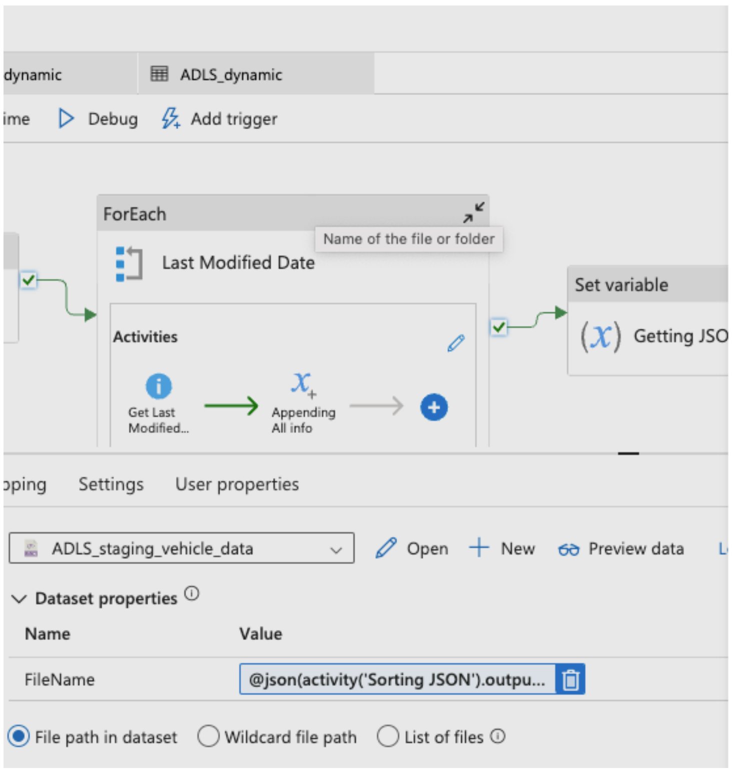
Parameter

Name	Type	Value
JsonArray	String	@variables('JsonArray')

Sorted JSON –

{
	"firstRow": {
		"OutputJson": "[{\"FileName\":\"3813f6ba-a6b9-4451-bc1e-ff5e50b1cd5e.json\",\"LastModified\":\"2023-08-29T11:10:56\"},{\"FileName\":\"cfb59238-9789-45f6-9bfe-006bf4ed62b8.json\",\"LastModified\":\"2023-08-29T11:10:23\"},{\"FileName\":\"8667436a-7977-4484-8327-061f4a731c4b\",\"LastModified\":\"2023-08-29T10:43:33\"},{\"FileName\":\"abbd4901-53b4-4d1b-b4dc-aacee0a862c2\",\"LastModified\":\"2023-08-29T10:42:06\"}]"
	},
	"effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (East US)",
	"billingReference": {
		"activityType": "PipelineActivity",
		"billableDuration": [
			{
				"meterType": "AzureIR",
				"duration": 0.016666666666666666,
				"unit": "Hours"
			}
		]
	},
	"durationInQueue": {
		"integrationRuntimeQueue": 1
	}
}

9. As seen above, we have our latest file as the 1st element of the JSON Array and we will use Copy Activity now to have it fetch the first element's file name

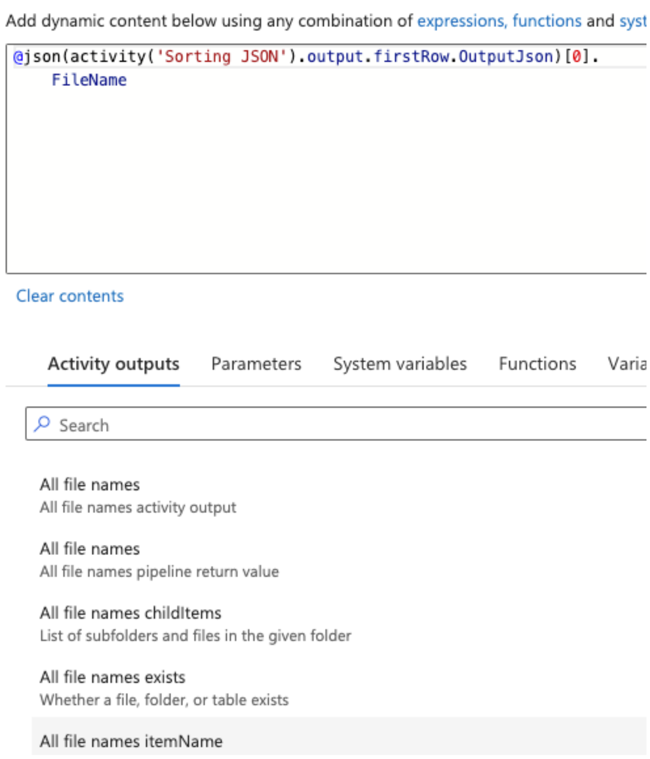


ADLS_staging_vehicle_data | Open | New | Preview data

Dataset properties

Name	Value
FileName	@json(activity('Sorting JSON').outpu...

[✓ File path in dataset] Wildcard file path List of files



Add dynamic content below using any combination of expressions, functions and syst

```
@json(activity('Sorting JSON').output.firstRow.OutputJson)[0].
    FileName
```

All file names
All file names activity output

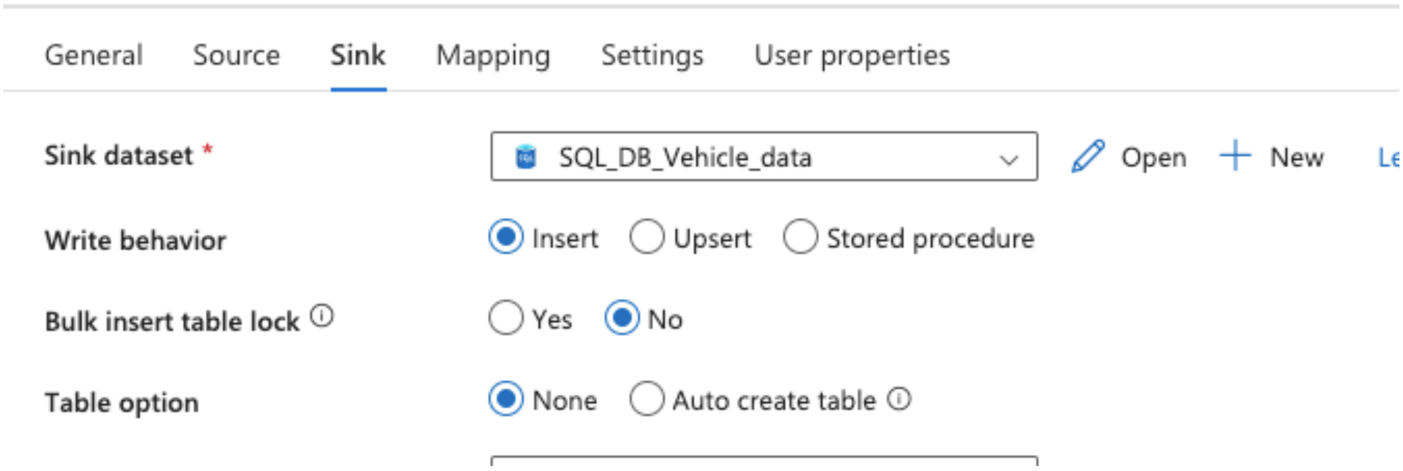
All file names
All file names pipeline return value

All file names childItems
List of subfolders and files in the given folder

All file names exists
Whether a file, folder, or table exists

All file names itemName

10. Sink settings –



Sink dataset *	SQL_DB_Vehicle_data
Write behavior	[✓ Insert] Upsert Stored procedure
Bulk insert table lock	Yes [✓ No]
Table option	[✓ None] Auto create table

Now after adding a Storage Event based trigger for this pipeline and by setting it to Staging Folder, each time it gets a new file, this pipeline will get triggered and this pipeline will get the latest file and put it in the SQL Server Table.