What is Get Metadata Activity

“Get Metadata” Activity can be used to retrieve the metadata of any type of file, folder or relational database table in Azure Data Factory.

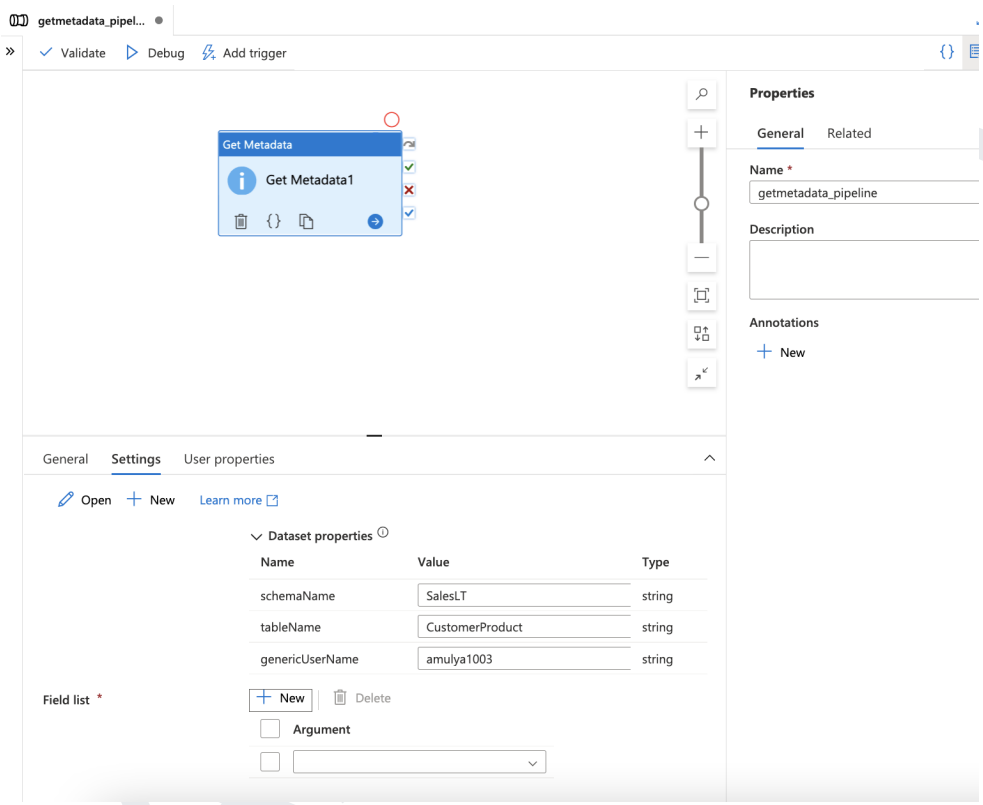
The Output of the “Get Metadata” Activity can be used in the “Conditional Expressions” to perform Validation, or, Consume the Metadata in the Subsequent Activities.

While working in Azure Data Factory, sometimes we need to retrieve metadata information, like the file name, file size, file existence, etc. We can use the Get Metadata activity to retrieve metadata information from the data set and then we can use that metadata information in subsequent activities.

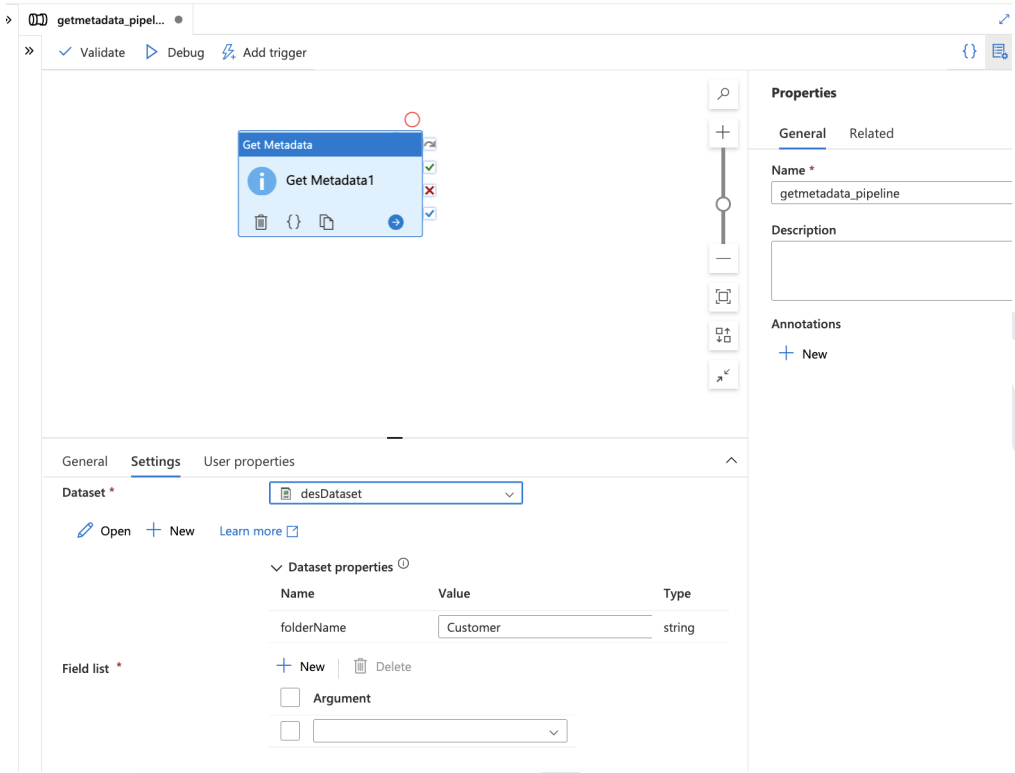
Attribute name	Data source type	Description
itemName	File storages	Name of the file or folder.
itemType	File storages	Type of the file or folder. The output value is File Folder.
size	File storages	Size of the file in bytes. Applicable to file only.
created	File storages	Created date/time of the file or folder.
lastModified	File storages	Last modified date/time of the file or folder.
childItems	File storages	List of sub-folders and files inside the given folder. Applicable to the folder object only. The output value is a list of name and type of each child item.
contentMD5	File storages	MD5 of the file. Applicable to file only.
structure	File and database systems	Data structure inside the file or relational database table. The output value is a list of column name and column type.
columnCount	File and database systems	The number of columns inside the file or relational table.
exists	File and database systems	Whether a file/folder/table exists or not. Note if "exists" is specified in the GetaMetadata field list, the activity will not fail even when the item (file/folder/table) does not exist; instead, it returns exists: false in the output.

please note that the **childItems** attribute from this list is applicable to folders only and is designed to provide list of files and folders nested within the source folder.

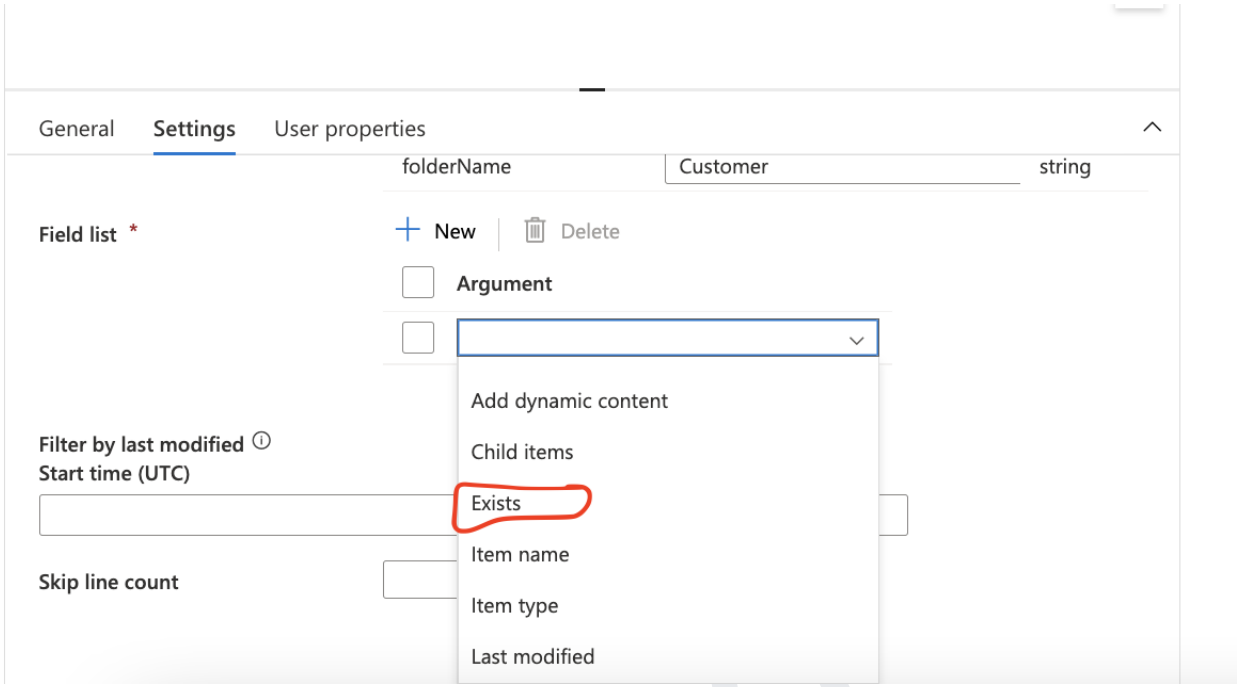
In this scenario, I will retrieve the metadata information of a folder that is stored in a data lake folder.



Get Metadata connected to folder in ADLS:

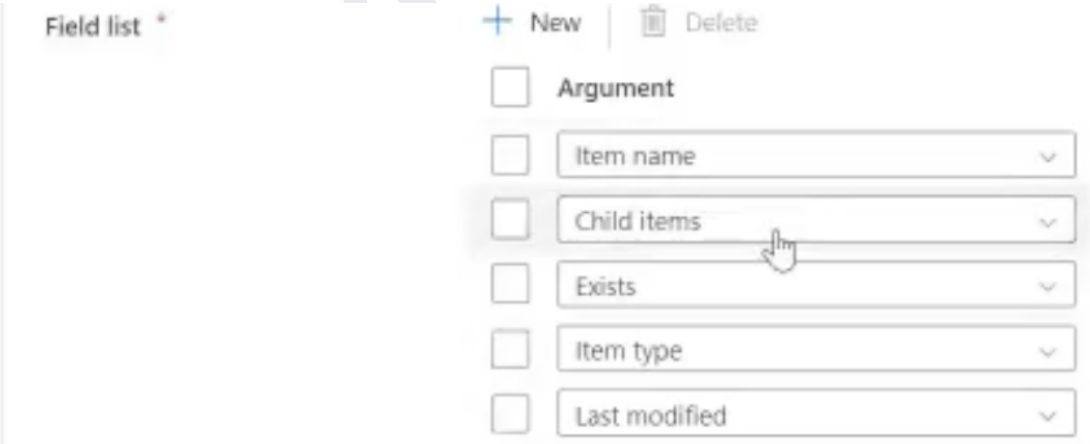


IN Settings tab-> It checks for folder “Exists”, If Exists it will give true , otherwise it gives false.



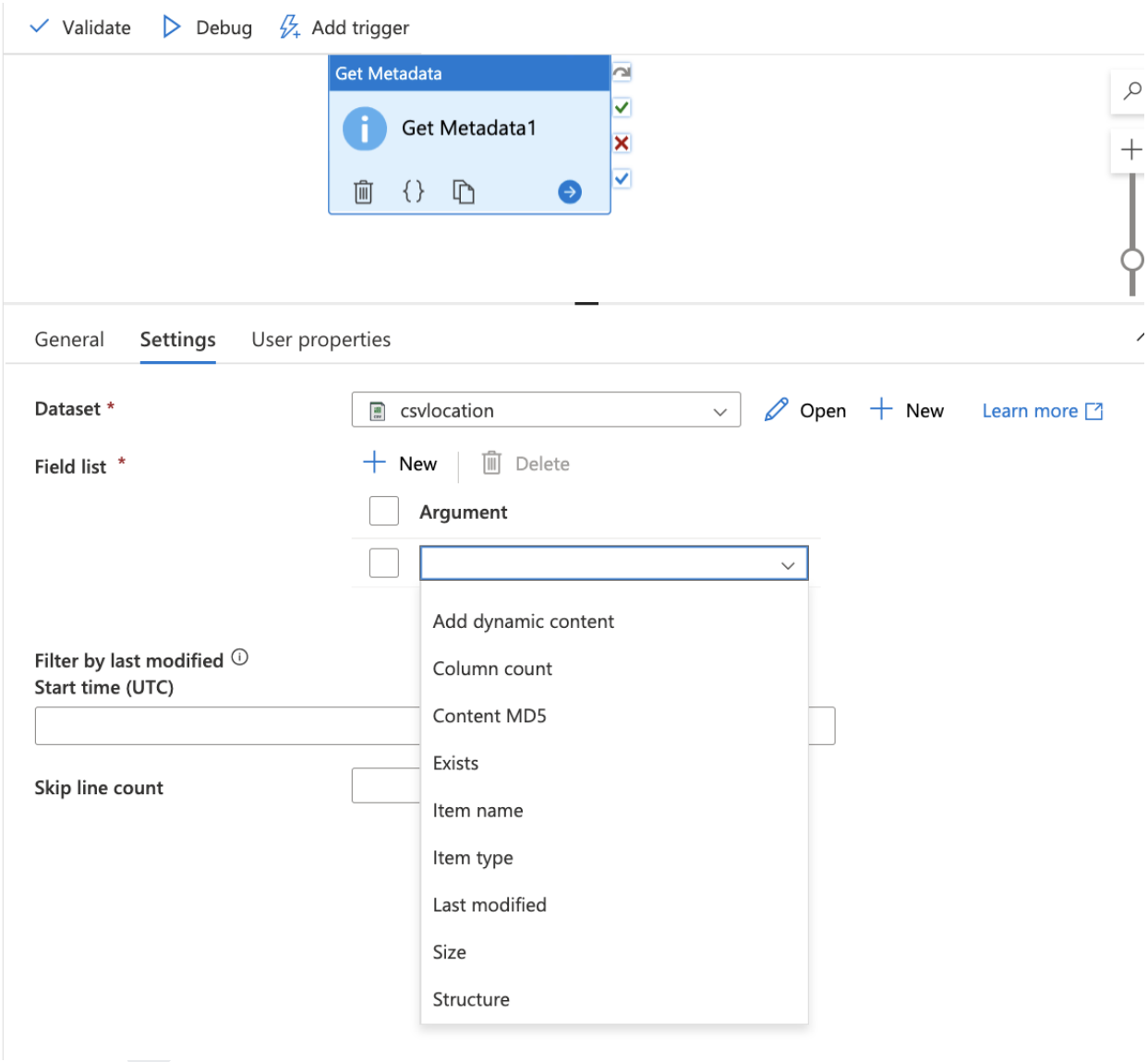
ItemName : it will gives the folder Name.

Instead of selecting the single item, we can click on “New” under the field List and add multiple items.

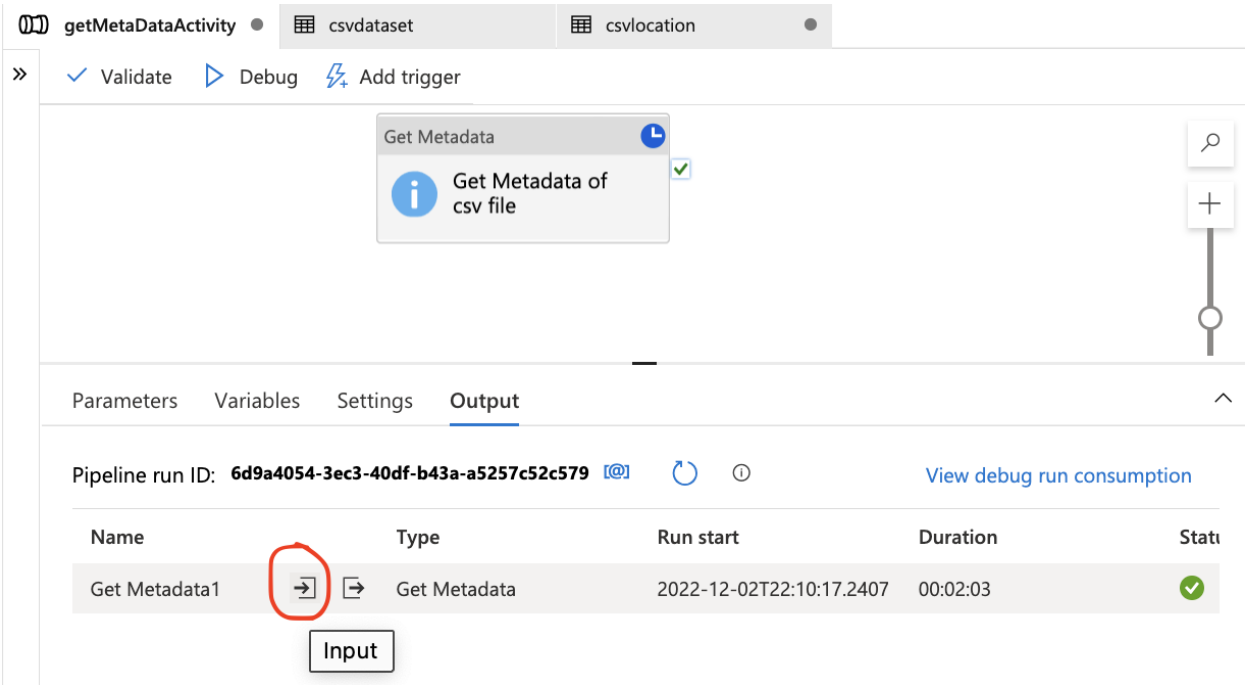


For folder location , we can get these many options under the field list

Get Metadata connected to file in ADLS:

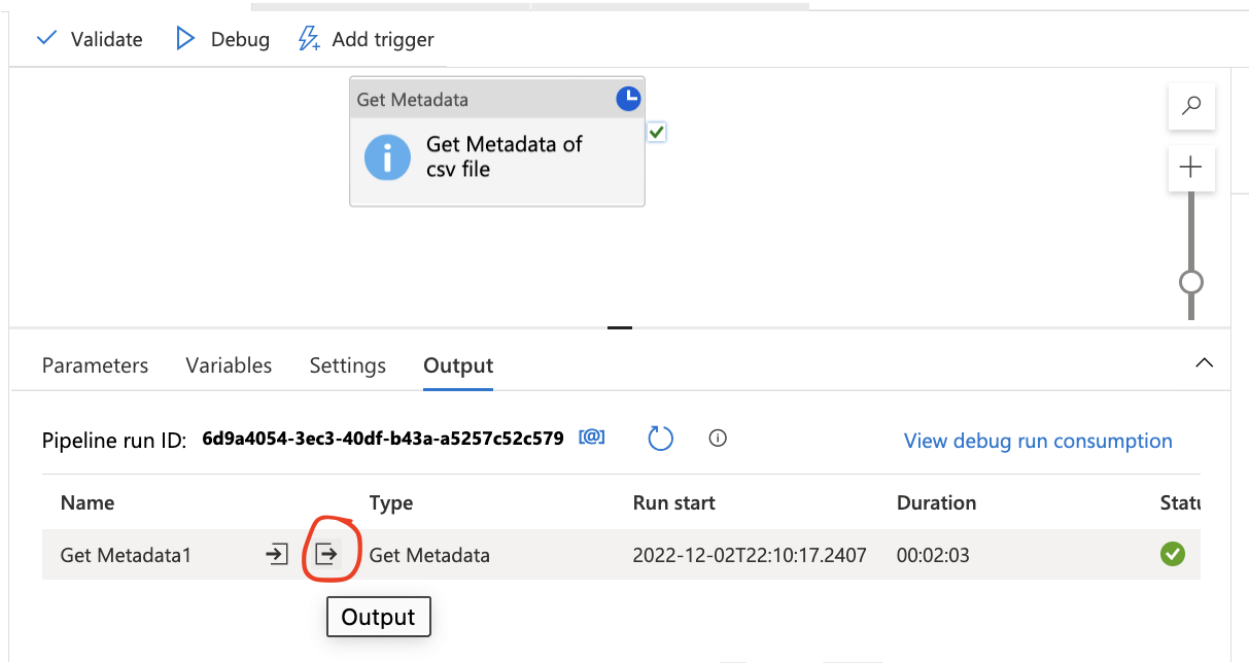


This is the Input , we are passing:



Input


```
{
    "dataset": {
        "referenceName": "csvlocation",
        "type": "DatasetReference",
        "parameters": {}
    },
    "fieldList": [
        "columnCount",
        "contentMD5",
        "exists",
        "itemName",
        "itemType",
        "lastModified",
        "size",
        "structure"
    ],
    "storeSettings": {
        "type": "AzureBlobFSReadSettings",
        "enablePartitionDiscovery": false
    },
    "formatSettings": {
        "type": "DelimitedTextReadSettings"
    }
}
```




```
{
    "contentMD5": null,
    "exists": true,
    "itemName": "SalesLT.Product.csv",
    "itemType": "File",
    "lastModified": "2022-12-01T01:51:07Z",
    "size": 1358360,
    "structure": [
        {
            "name": "Prop_0",
            "type": "String"
        },
        {
            "name": "Prop_1",
            "type": "String"
```

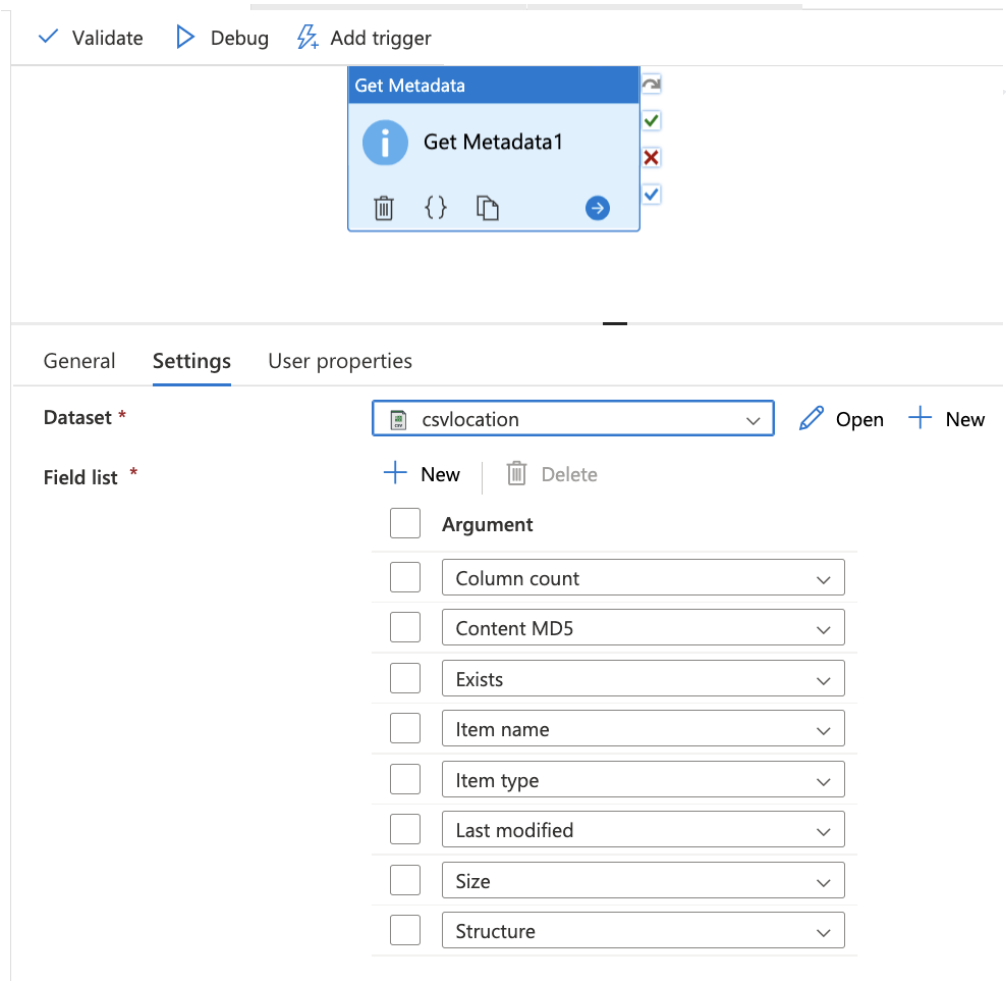
```
},
{
    "name": "Prop_2",
    "type": "String"
},
{
    "name": "Prop_3",
    "type": "String"
},
{
    "name": "Prop_4",
    "type": "String"
},
{
    "name": "Prop_5",
    "type": "String"
},
{
    "name": "Prop_6",
    "type": "String"
},
{
    "name": "Prop_7",
    "type": "String"
},
```

{
"name": "Prop_8",
"type": "String"
},
{
"name": "Prop_9",
"type": "String"
},
{
"name": "Prop_10",
"type": "String"
},
{
"name": "Prop_11",
"type": "String"
},
{
"name": "Prop_12",
"type": "String"
},
{
"name": "Prop_13",
"type": "String"
},
{

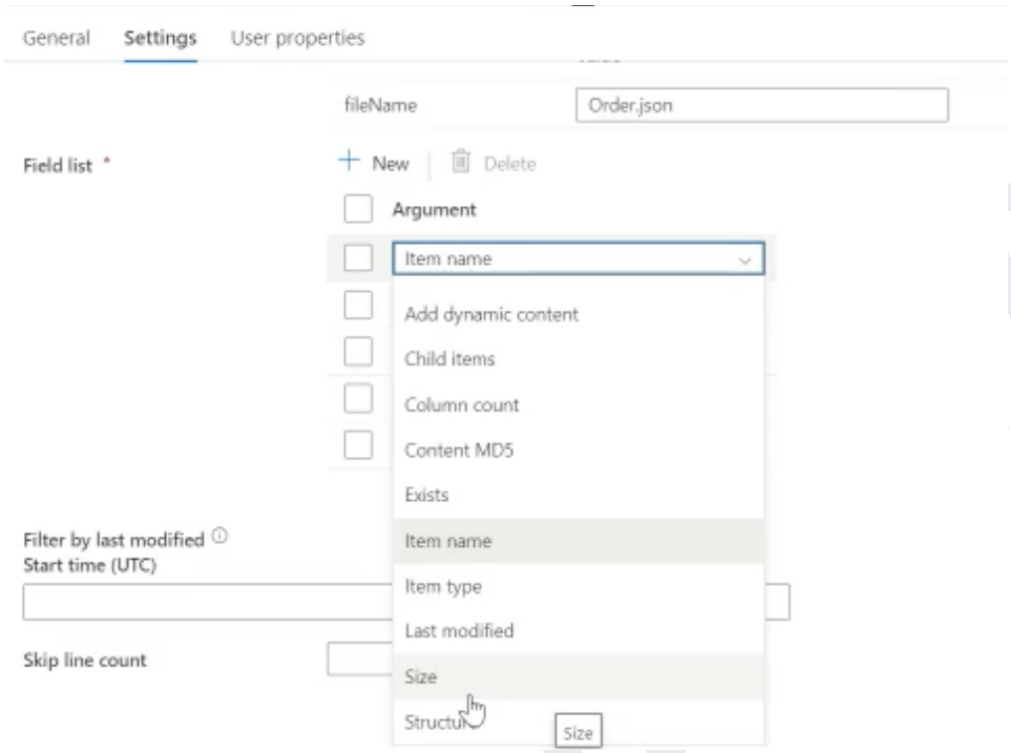
"name": "Prop_14",
"type": "String"
},
{
"name": "Prop_15",
"type": "String"
},
{
"name": "Prop_16",
"type": "String"
}
],
"columnCount": 17,
"effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (East US)",
"executionDuration": 3,
"durationInQueue": {
"integrationRuntimeQueue": 114
},
"billingReference": {
"activityType": "PipelineActivity",
"billableDuration": [
{
"meterType": "ManagedVNetIR",
"duration": 0.016666666666666666,

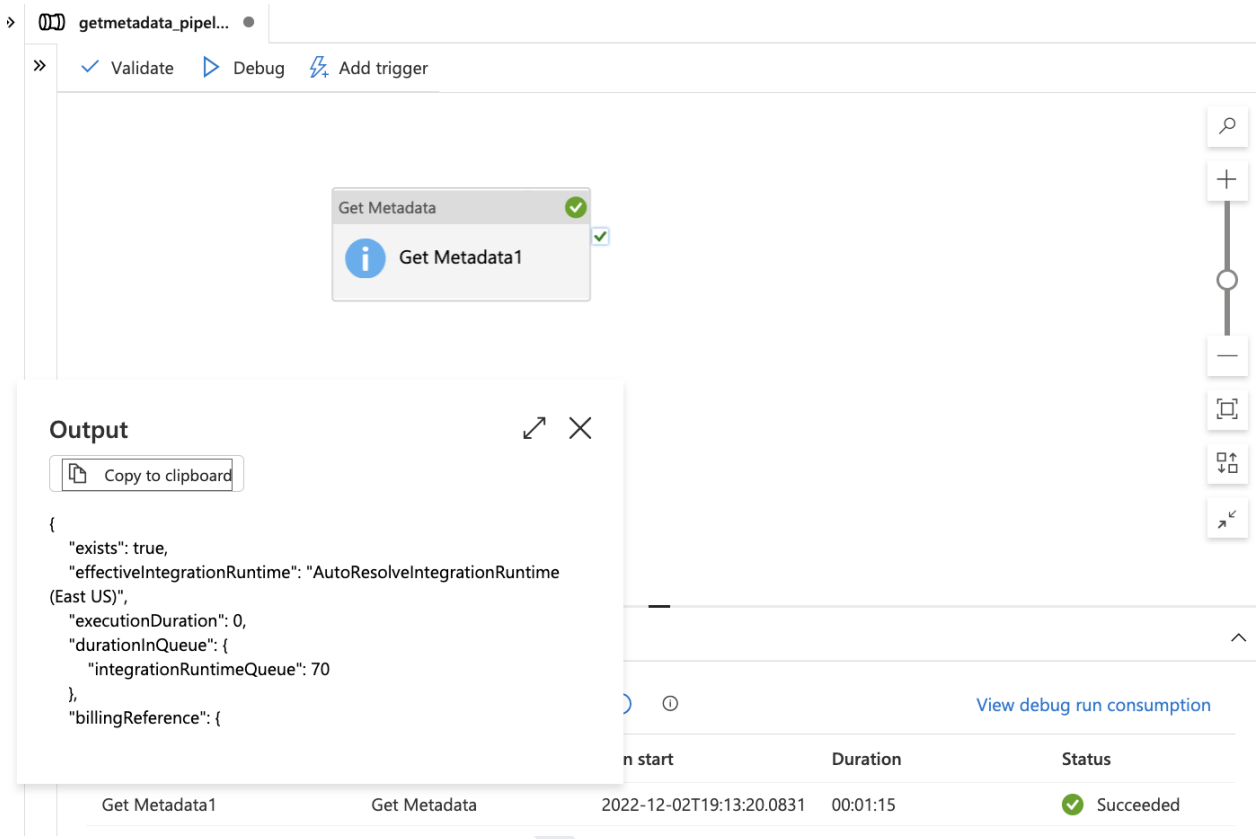
```
            "unit": "Hours"
        }
    ]
  }
}
```

Let's add all arguments and check the output:



Now debug and see the output:

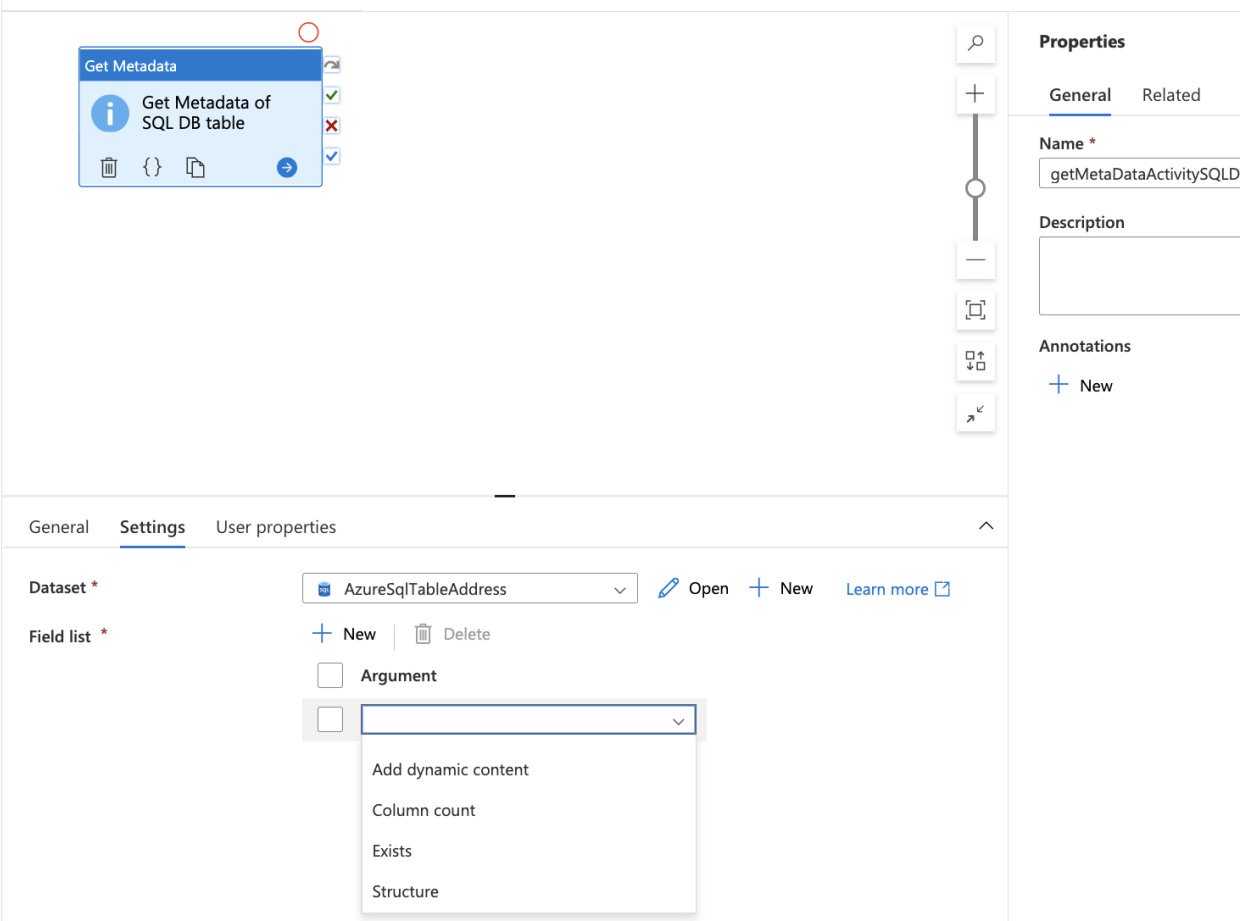


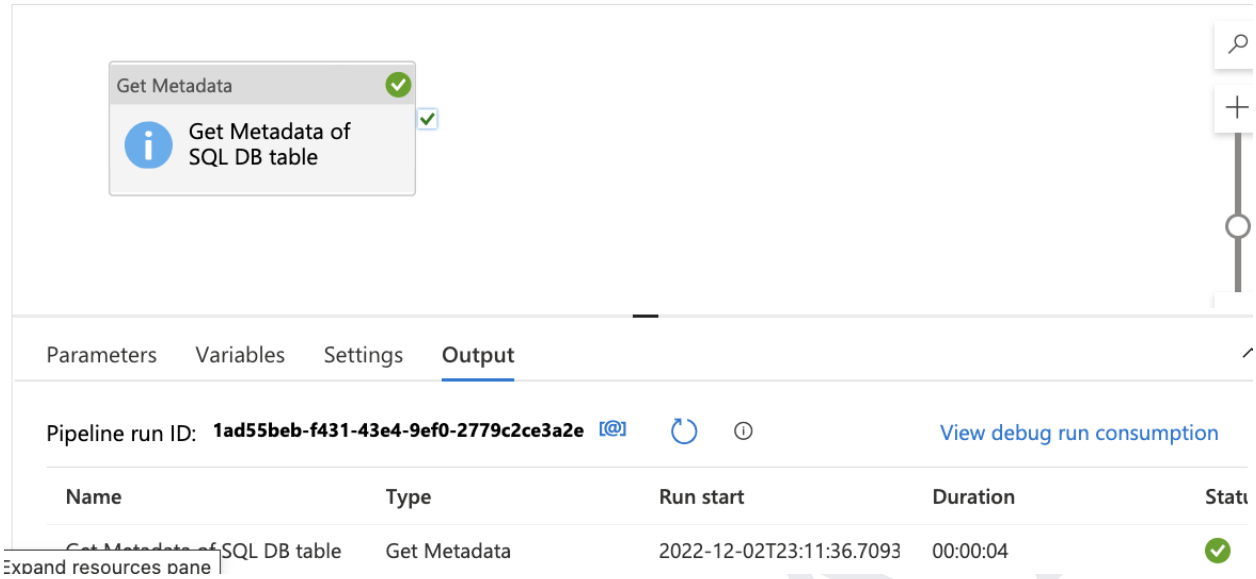


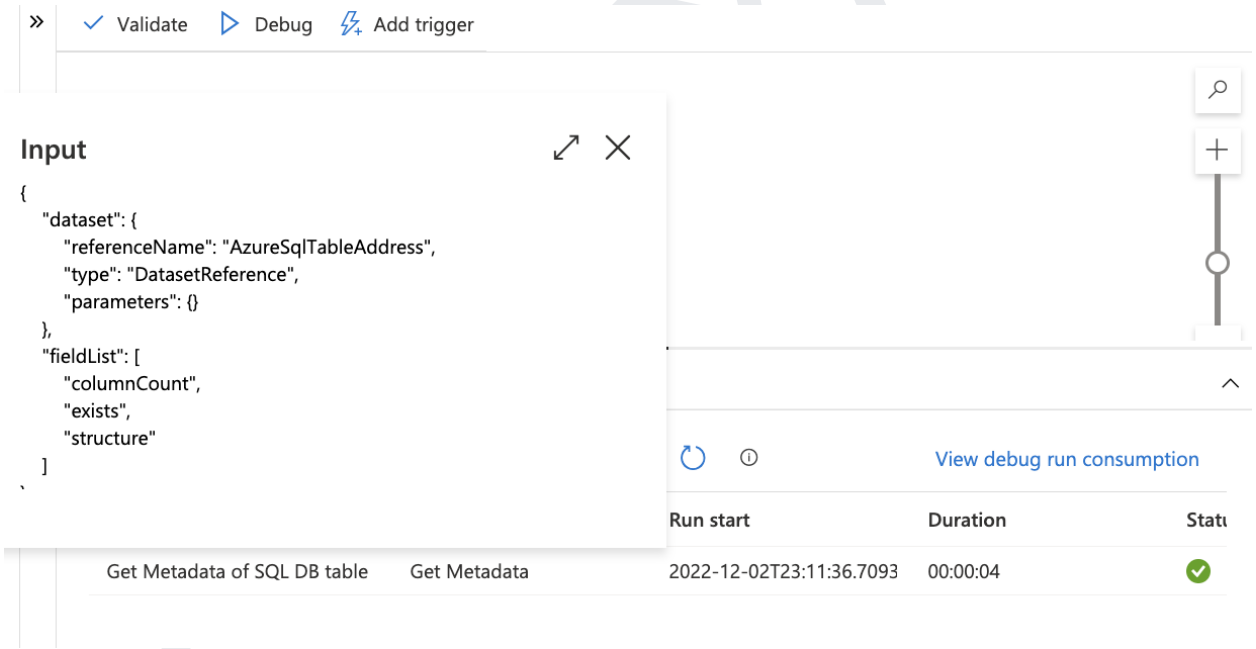

```
{
    "exists": true,
    "effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (East US)",
    "executionDuration": 0,
    "durationInQueue": {
        "integrationRuntimeQueue": 70
    },
    "billingReference": {
        "activityType": "PipelineActivity",
        "billableDuration": [
            {
```

"meterType": "ManagedVNetIR",
"duration": 0.0166666666666666666,
"unit": "Hours"
}
]
}
}

Get Metadata connected to SQL database:




Get Metadata
Get Metadata of SQL DB table
Parameters
Variables
Settings
Output
Pipeline run ID: 1ad55beb-f431-43e4-9ef0-2779c2ce3a2e
View debug run consumption
Name
Type
Run start
Duration
Statu
SQL DB table
Get Metadata
2022-12-02T23:11:36.7093
00:00:04
Expand resources pane


Validate
Debug
Add trigger
Input
{
"dataset": {
"referenceName": "AzureSqlTableAddress",
"type": "DatasetReference",
"parameters": {}
},
"fieldList": [
"columnCount",
"exists",
"structure"
]
View debug run consumption
Run start
Duration
Statu
Get Metadata of SQL DB table
Get Metadata
2022-12-02T23:11:36.7093
00:00:04

Output

Copy to clipboard


```
{
    "exists": true,
    "structure": [
        {
            "physicalName": "AddressID",
            "type": "Int32",
            "logicalType": "Int32",
            "name": "AddressID",
            "physicalType": "int",
```


View debug run consumption

		n start	Duration	Statu
Get Metadata of SQL DB table	Get Metadata	2022-12-02T23:11:36.7093	00:00:04	✓

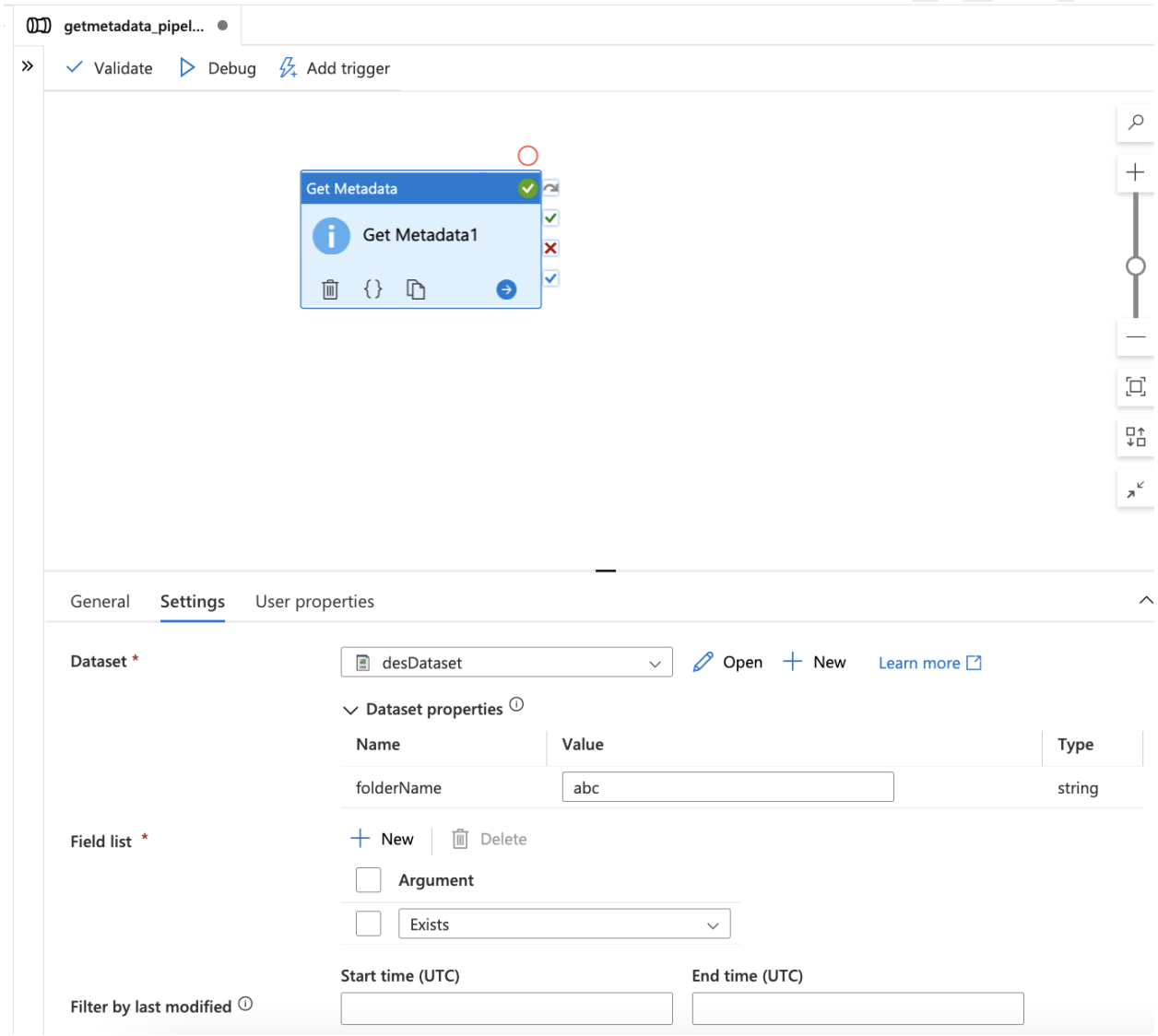
Output shows all the columns:

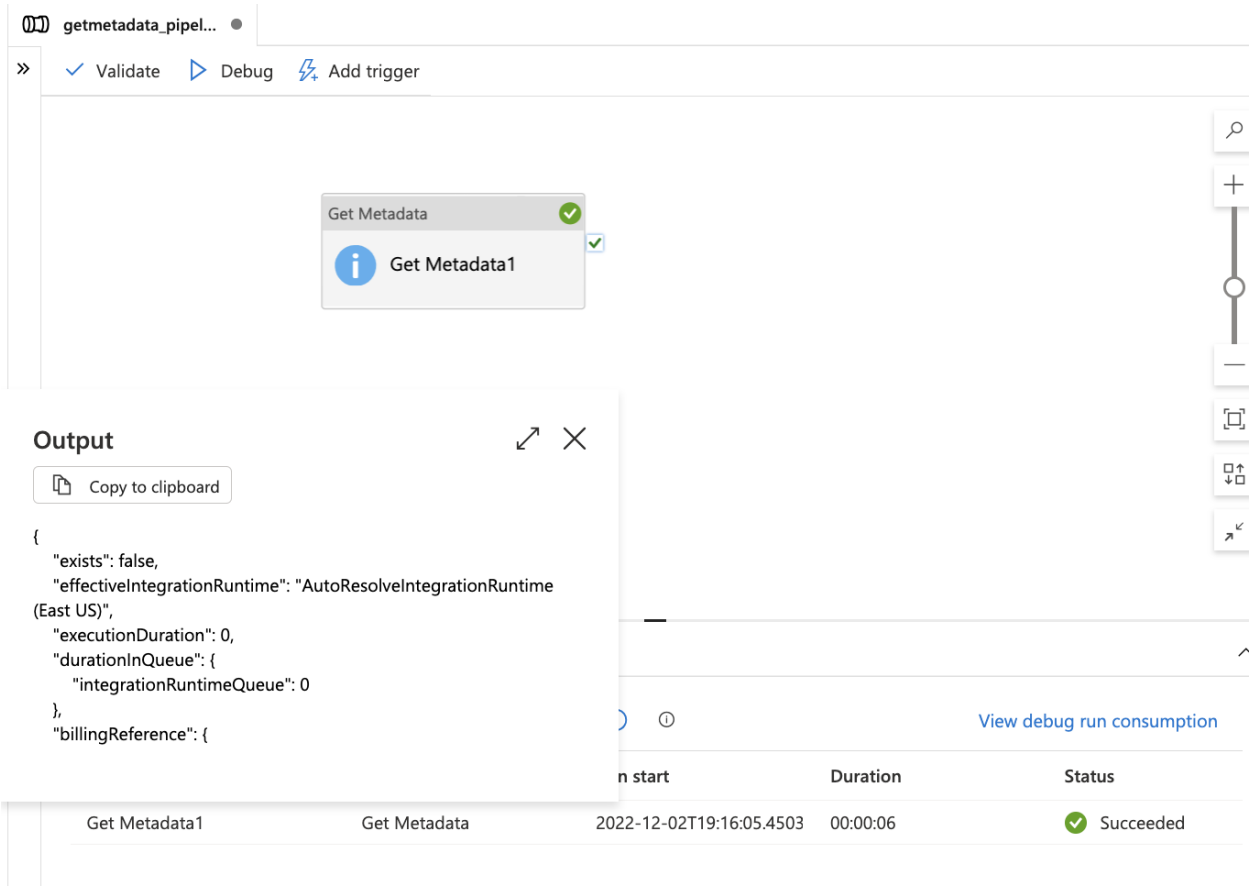
Output

Copy to clipboard


```
{
  "exists": true,
  "structure": [
    {
      "physicalName": "AddressID",
      "type": "Int32",
      "logicalType": "Int32",
      "name": "AddressID",
      "physicalType": "int",
      "precision": 10,
      "scale": 255,
      "DotNetType": "System.Int32, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "AddressLine1",
      "type": "String",
      "logicalType": "String",
      "name": "AddressLine1",
      "physicalType": "nvarchar",
      "precision": 255,
      "scale": 255,
      "DotNetType": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "AddressLine2",
      "type": "String",
      "logicalType": "String",
      "name": "AddressLine2",
      "physicalType": "nvarchar",
      "precision": 255,
      "scale": 255,
      "DotNetType": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "City",
      "type": "String",
      "logicalType": "String",
      "name": "City",
      "physicalType": "nvarchar",
      "precision": 255,
      "scale": 255,
      "DotNetType": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "StateProvince",
      "type": "String",
      "logicalType": "String",
      "name": "StateProvince",
      "physicalType": "nvarchar",
      "precision": 255,
      "scale": 255,
      "DotNetType": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "CountryRegion",
      "type": "String",
      "logicalType": "String",
      "name": "CountryRegion",
      "physicalType": "nvarchar",
      "precision": 255,
      "scale": 255,
      "DotNetType": "System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    },
    {
      "physicalName": "PostalCode",
      "type": "String",
      "logicalType": "String",
      "name": "PostalCode",
      "physicalType": "nvarchar",
```

Get Metadata connected to folder and check whether folder exists or not in ADLS:

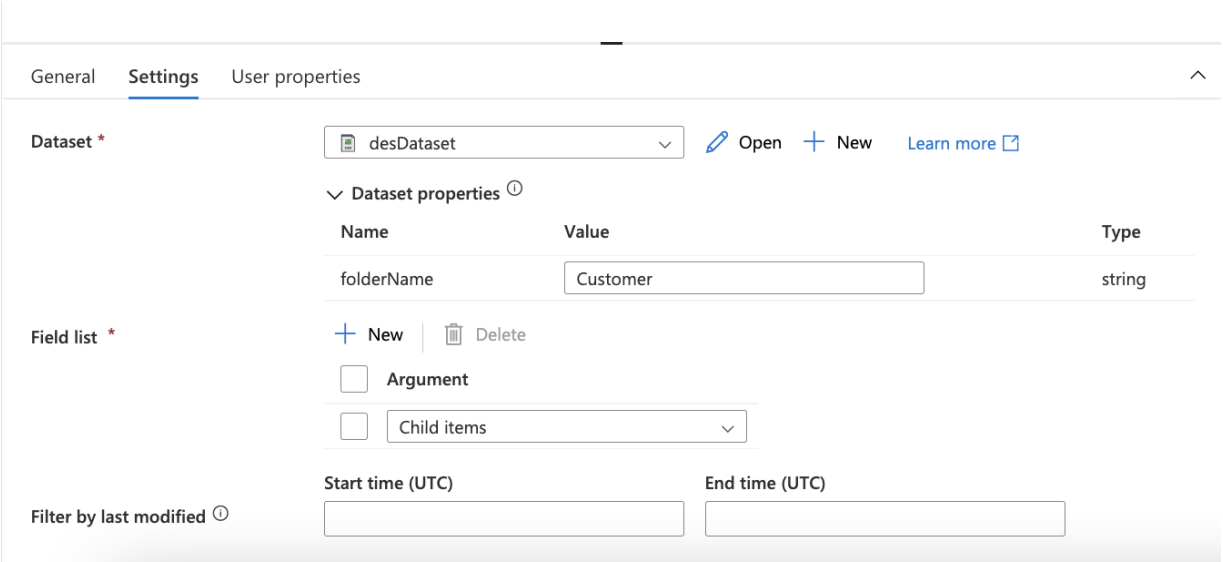




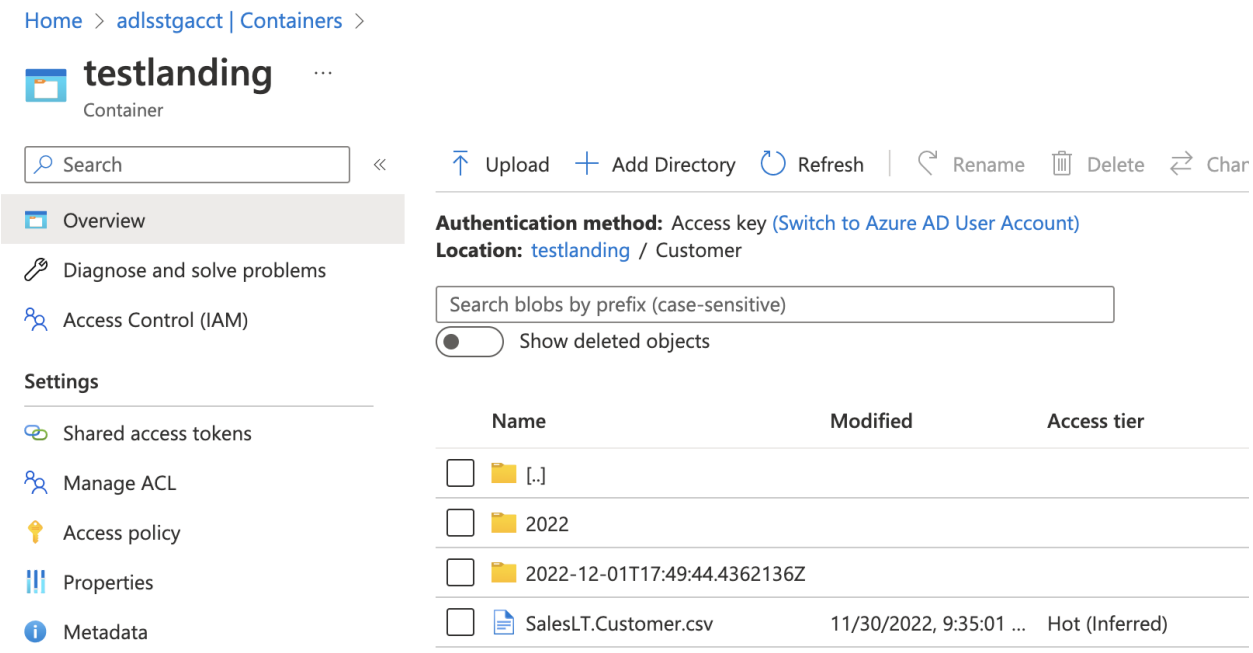
How we implement this argument exists for further process?

If folder exists, then we can use let's say **If Activity** to do some kind of copy activity to copy into other location or any thing else.

“Get child items” shows all the files under particular folder using Metadata activity:



It will show all files and folders in the “customer” folder



Output

Copy to clipboard


```
{
    "childItems": [
        {
            "name": "2022",
            "type": "Folder"
        },
        {
            "name": "2022-12-01T17:49:44.4362136Z",
            "type": "Folder"
        },
        {
            "name": "SalesLT.Customer.csv",
            "type": "File"
        }
    ],
    "effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (East US)",
    "executionDuration": 0,
    "durationInQueue": {
        "integrationRuntimeQueue": 1
    },
    "billingReference": {
        "activityType": "PipelineActivity",
        "billableDuration": [
            {
                "meterType": "ManagedVNetIR",
                "duration": 0.016666666666666666,
                "unit": "Hours"
            }
        ]
    }
}
```

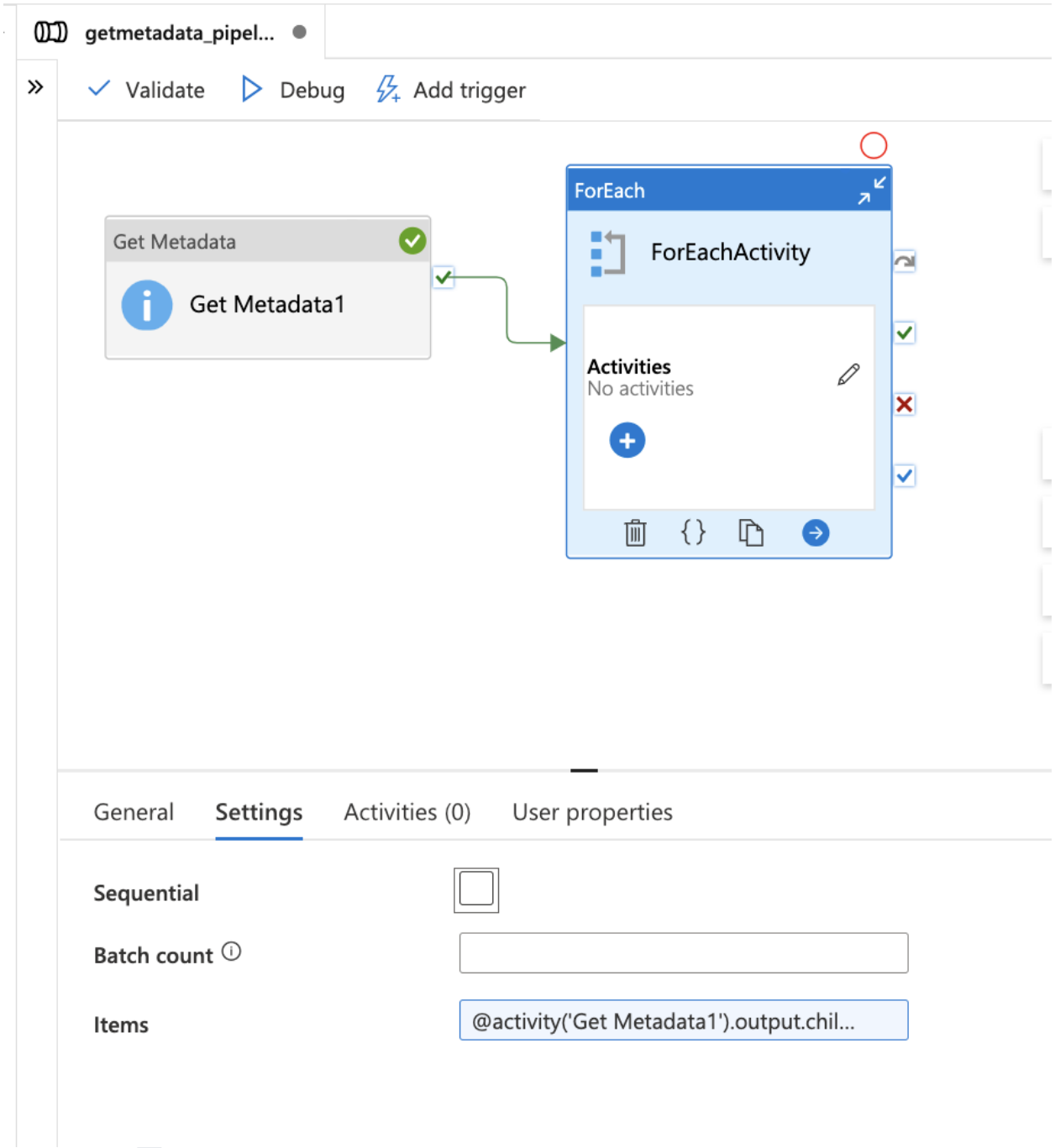
How we implement this “Child Items” for further process.

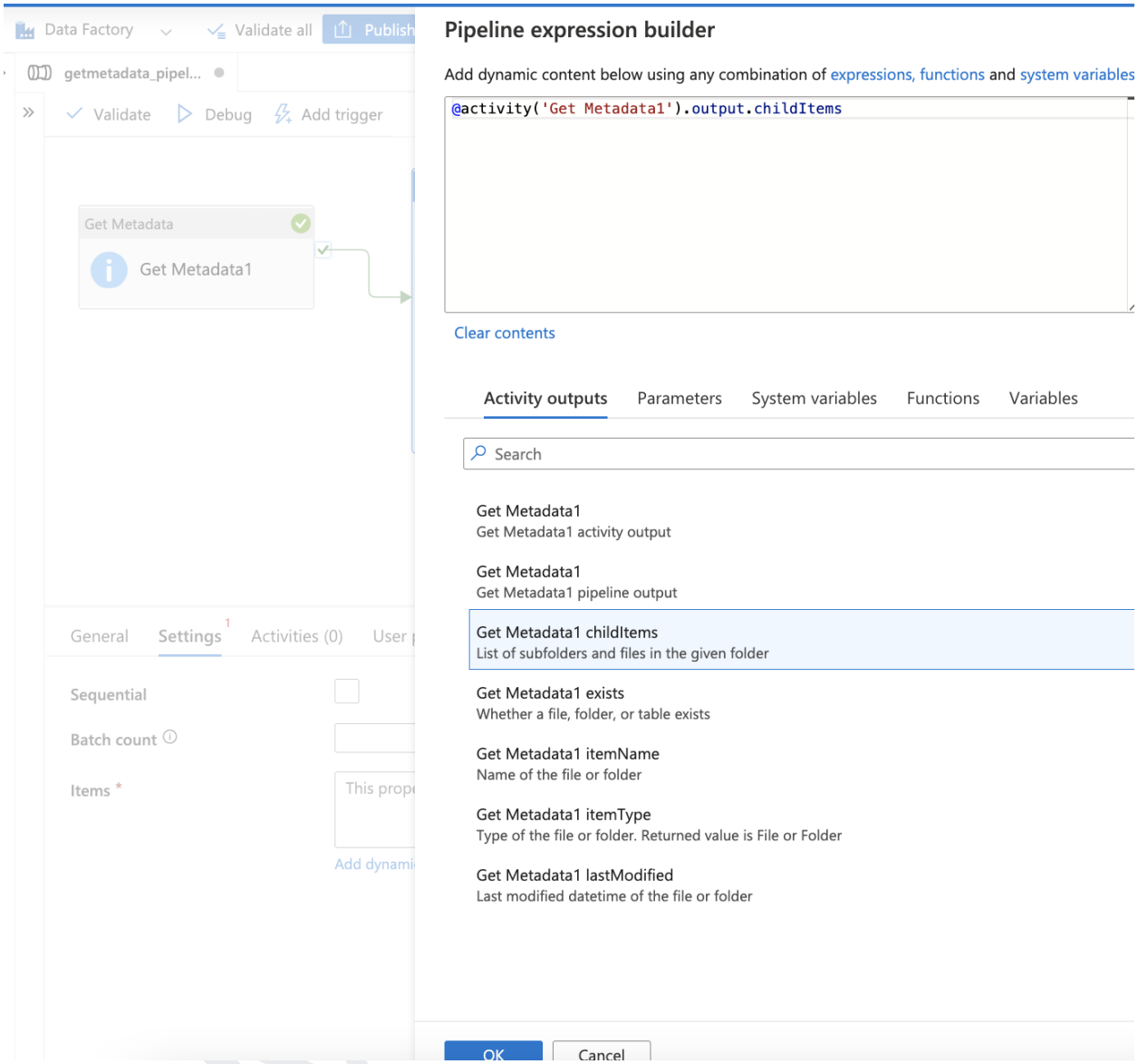
We can take this child item and do “foreach activity “ to loop over all folders or files.

Example:

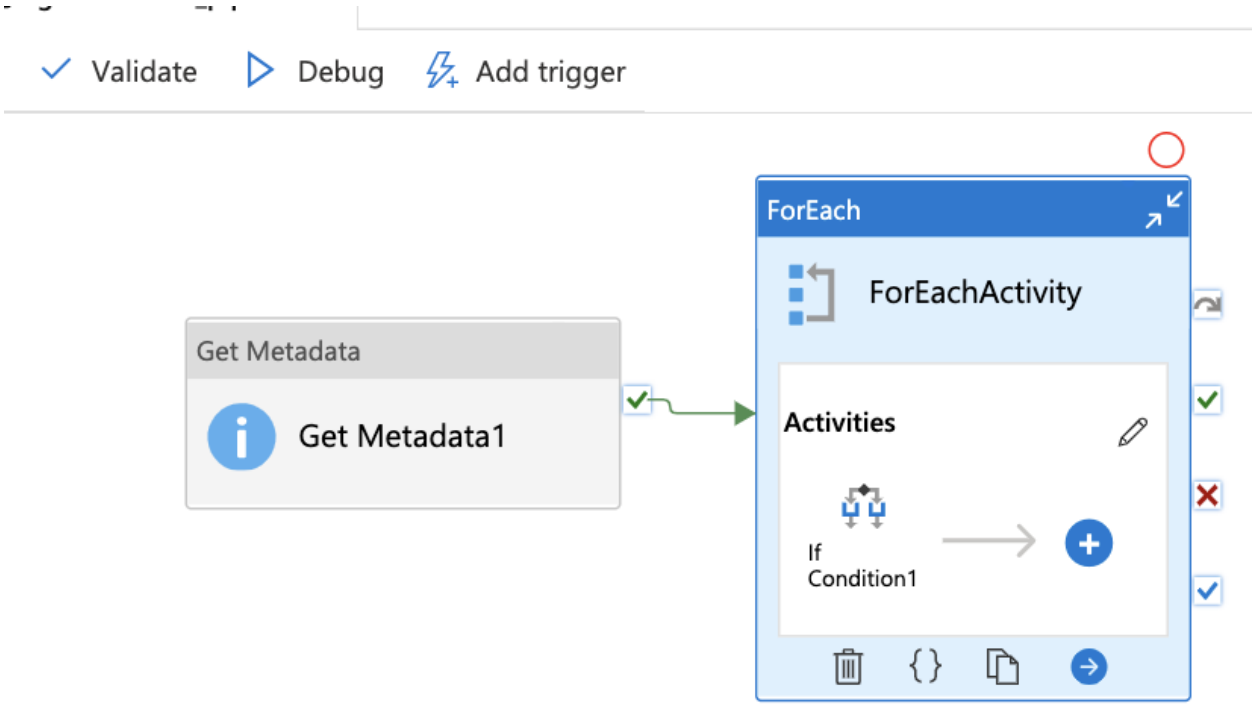
Assume we have a folder path in ADLS and this contain many folders and paths, out of this copy only copy one file into another location.

Get list of files-> for-each activity to loop over all files->use “If Activity”


getmetadata_pipel...
Validate
Debug
Add trigger
Get Metadata
Get Metadata1
ForEach
ForEachActivity
Activities
No activities
General
Settings
Activities (0)
User properties
Sequential
Batch count
Items
@activity('Get Metadata1').output.chil...

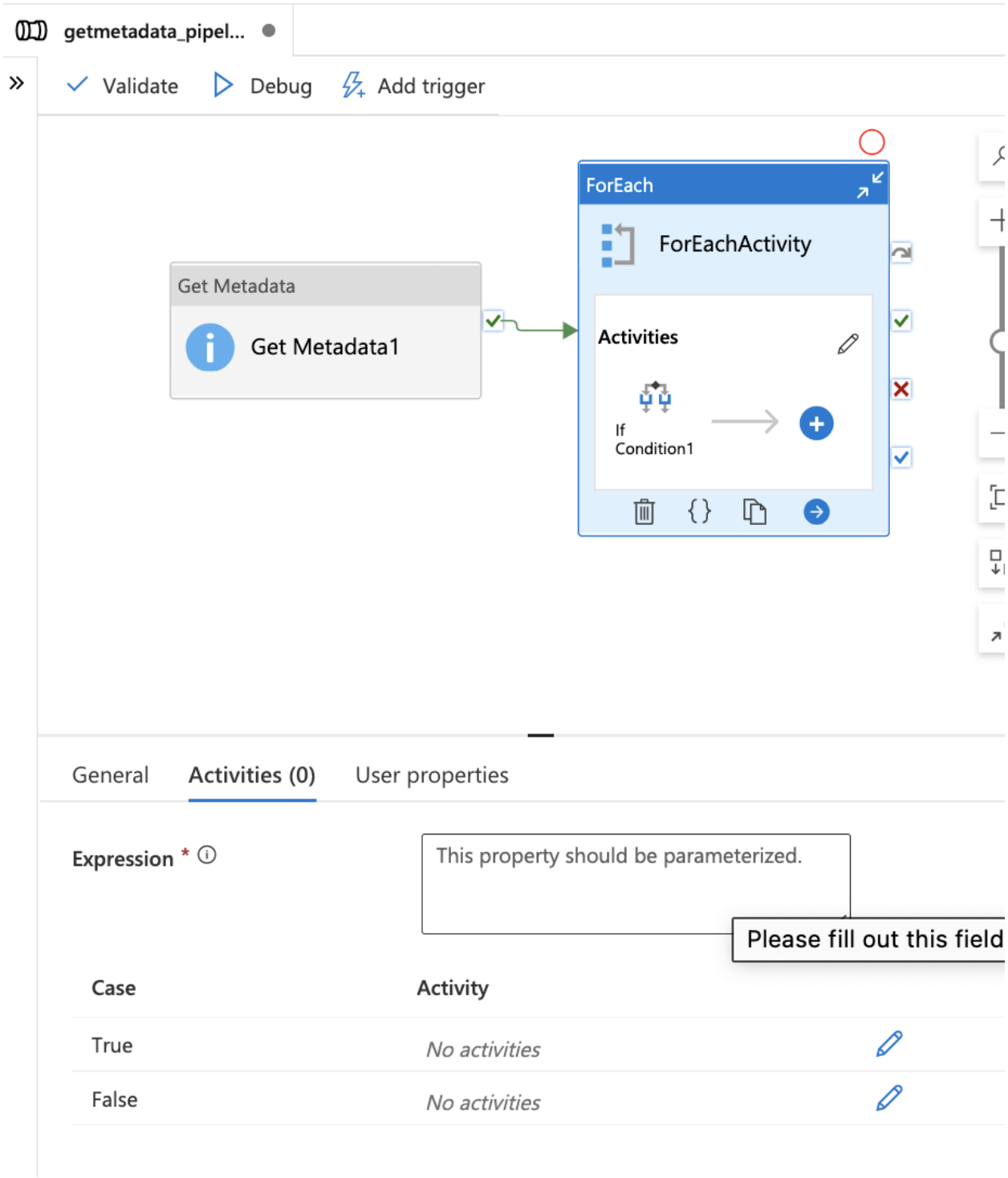

Data Factory
Validate all
Publish
getmetadata_pipel...
Validate
Debug
Add trigger
Get Metadata
Get Metadata1
General
Settings
Activities (0)
Sequential
Batch count
Items *
Pipeline expression builder
Add dynamic content below using any combination of expressions, functions and system variables
@activity('Get Metadata1').output.childItems
Clear contents
Activity outputs
Parameters
System variables
Functions
Variables
Search
Get Metadata1
Get Metadata1 activity output
Get Metadata1
Get Metadata1 pipeline output
Get Metadata1 childItems
List of subfolders and files in the given folder
Get Metadata1 exists
Whether a file, folder, or table exists
Get Metadata1 itemName
Name of the file or folder
Get Metadata1 itemType
Type of the file or folder. Returned value is File or Folder
Get Metadata1 lastModified
Last modified datetime of the file or folder
OK
Cancel

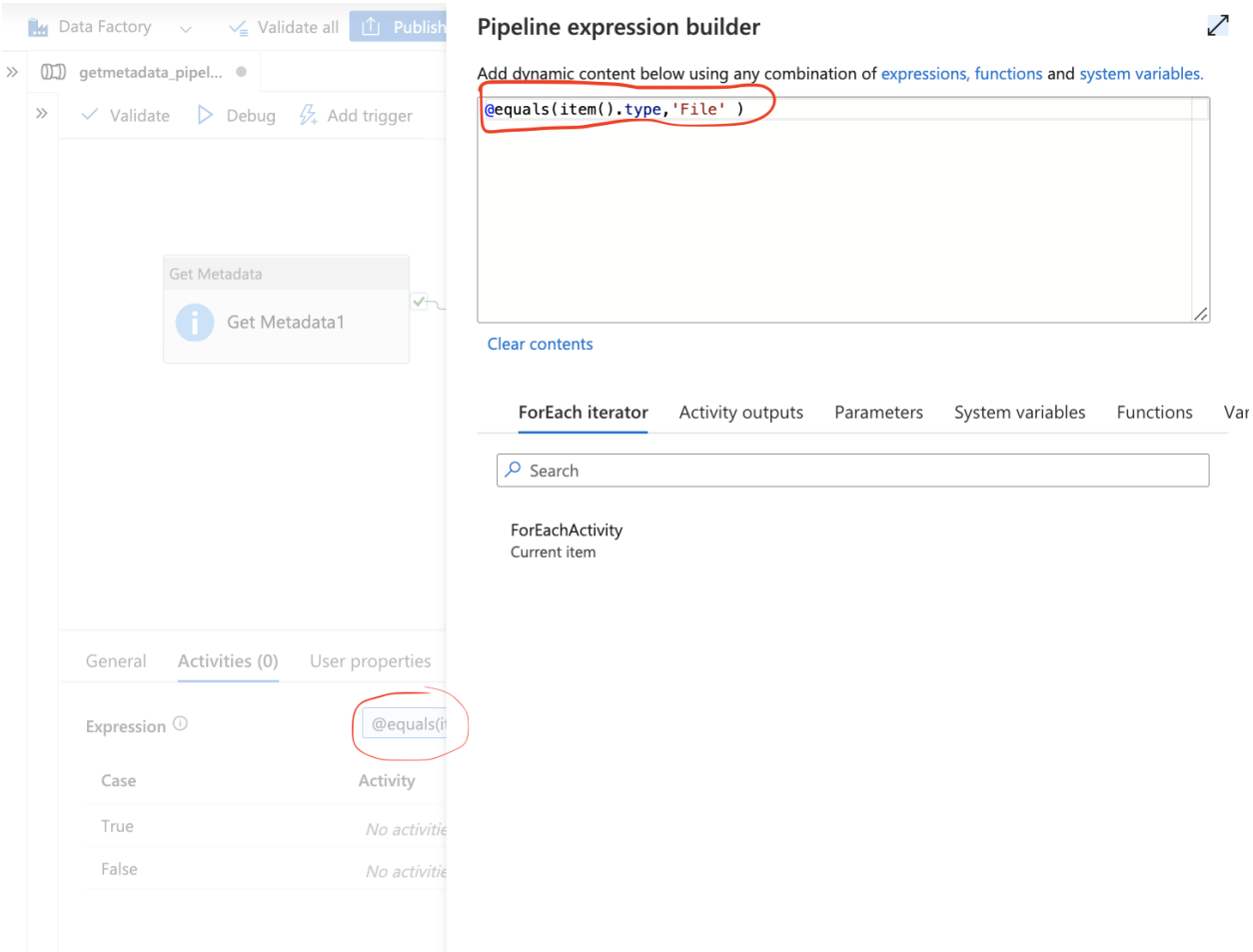
If activity inside for-each activity:



Check if condition is type of file or not:

Go to expression under IF Activity:

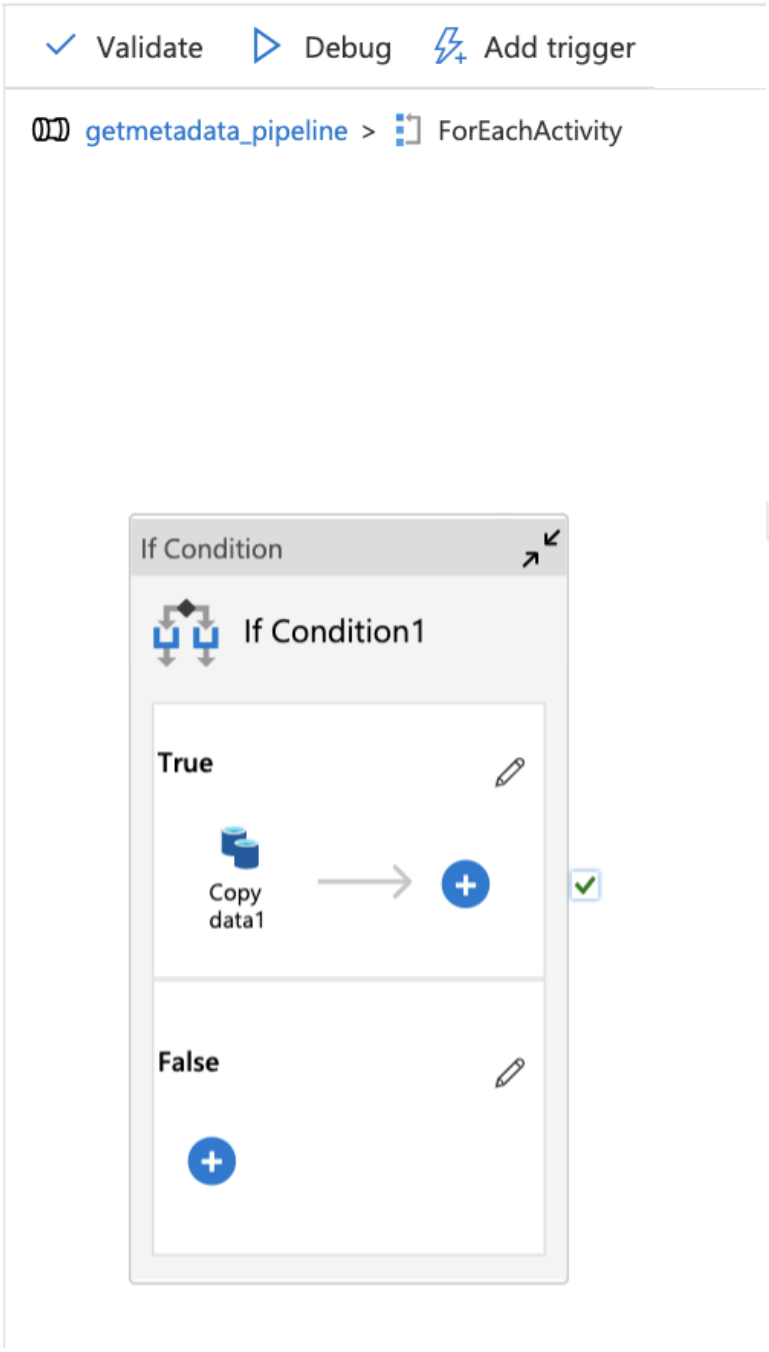


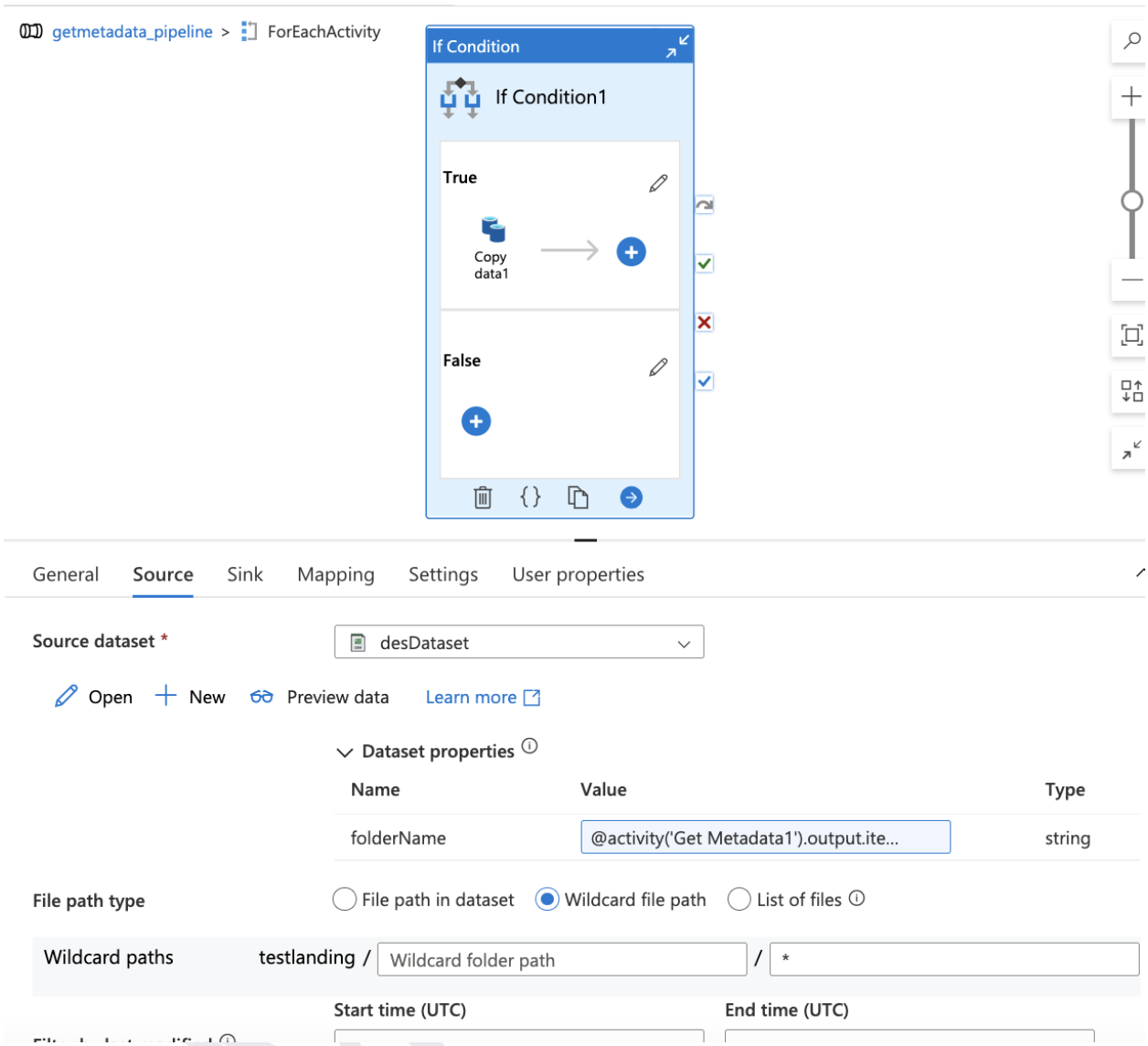


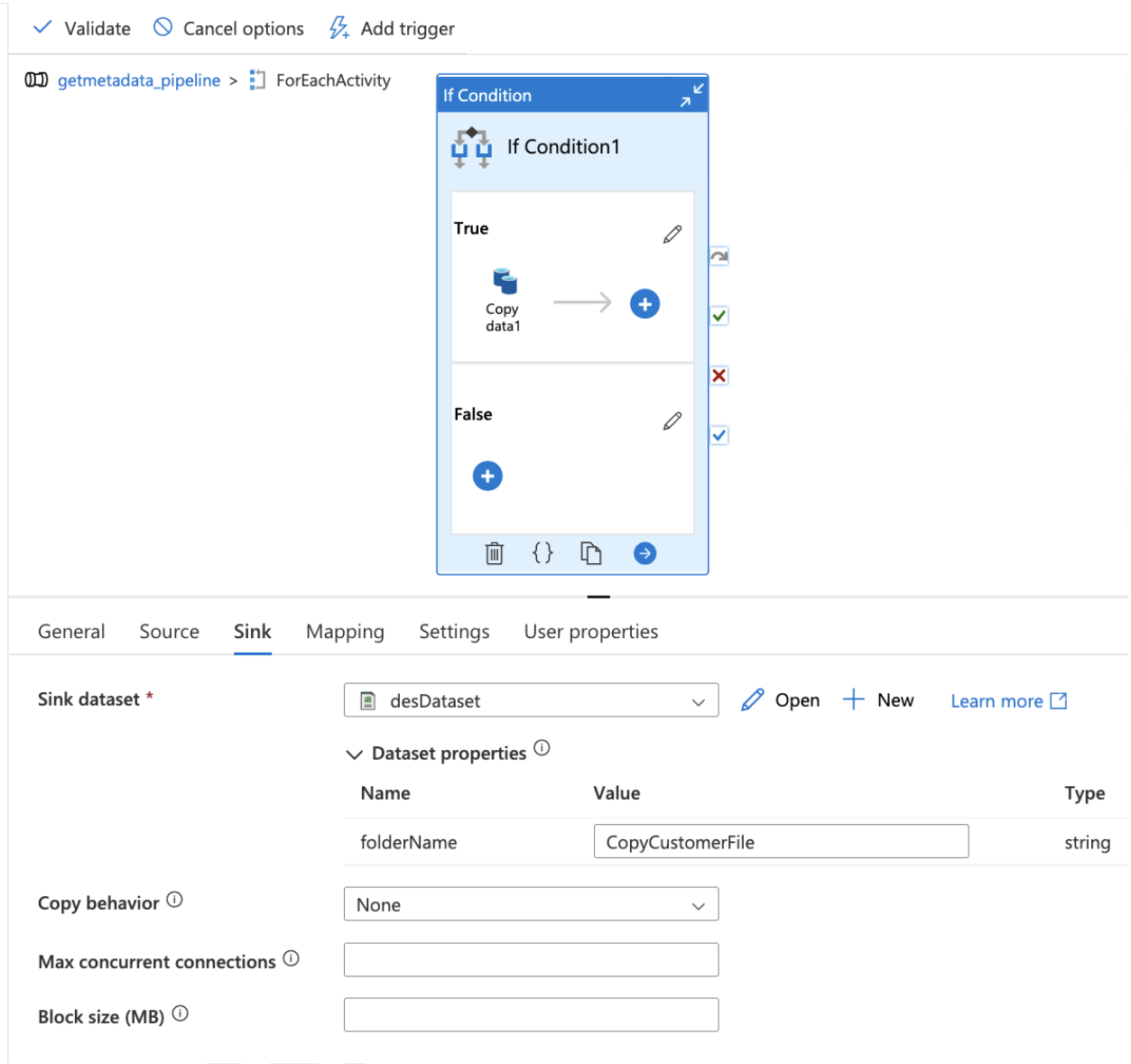
This is the expression to verify if it is a file or not.

```
{
    "childItems": [
        {
            "name": "2022",
            "type": "Folder"
        },
        {
            "name": "2022-12-01T17:49:44.4362136Z",
            "type": "Folder"
        },
        {
            "name": "SalesLT.Customer.csv",
            "type": "File"
        }
```

If “If condition” is true , we do copy activity.




getmetadata_pipeline > ForEachActivity
If Condition
If Condition1
True
Copy data1
False
General
Source
Sink
Mapping
Settings
User properties
Source dataset *
desDataset
Open
New
Preview data
Learn more
Dataset properties
Name
Value
Type
folderName
@activity('Get Metadata1').output.ite...
string
File path type
File path in dataset
Wildcard file path
List of files
Wildcard paths
testlanding /
Wildcard folder path
/
*
Start time (UTC)
End time (UTC)



This is an example to explain in general how to use “metadata activity” for other activities in pipeline for further process.

Limitations of the Get Metadata Activity

- "Wildcard Filter" on the folders, or, files, is not supported for the "Get Metadata" Activity.
- The maximum size of returned Metadata is 4MB.
- For "Azure Blob Storage", the "field List" Property "lastModified" applies to the Container and the Blob, but, not to the Virtual Folder.
- The "field List" Properties "structure" and "columnCount" are not supported when getting the Metadata from "Binary", "JSON", or, "XML" Files.
- When the Filters "modifiedDatetimeStart" and "modifiedDatetimeEnd" are set on the File Store Connectors of the "Get Metadata" Activity, the Metadata information can only be retrieved from a folder, and, not from a file, in the specified path.
- When the Filters "modifiedDatetimeStart" and "modifiedDatetimeEnd" are set on the File Store Connectors of the "Get Metadata" Activity, the "field list" Property "childItems", in the Output of the "Get Metadata" Activity, includes only the files that are "Modified" within the specified time range in the specified path, but, not the items inside the sub-folders.
- To apply the Filters "modifiedDatetimeStart" and "modifiedDatetimeEnd" on the File Store Connectors of the "Get Metadata" Activity, it will enumerate all the files in the specified folder, and, check the "Modified Time". Hence, it is best to avoid pointing to a folder with a large number of files, even if the expected qualified file count is small.
- For the "field list" Property "structure" to provide the actual Data Structure for the "Delimited Text" and "Excel" format Datasets, the Property "First row as header" must be enabled, which is supported for only these two Data Sources.