AZURE DATA FACTORY-PASSING PARAMETERS

Passing parameter between pipeline and activity as well as between activities

This articles walks you through how to pass parameters between a pipeline and activity as well as between the activities

Written By-

Blesson John (Data Solution Architect-Microsoft)

Issagha BA (Data Solution Architect-Microsoft)

Reviewed By-

Ye Xu (Senior Program Manager-ADF)

Gaurav Malhotra (Principal Program Manager)

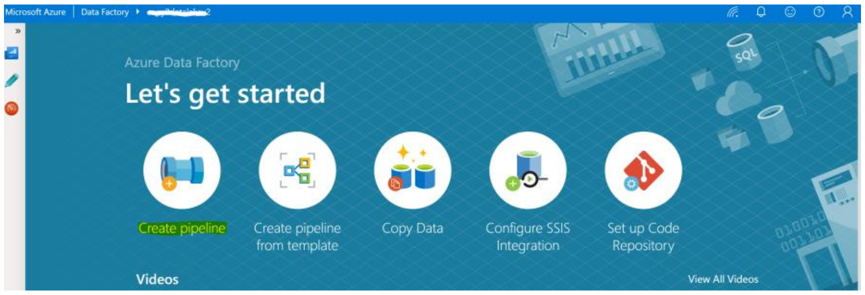
Contents

Azure data Factory –Passing Parameters

Passing parameters to ADF is quite important as it provides the flexibility required to create dynamic pipelines. To reference a parameter, one will have to provide the fully qualified name of the parameter. It is worth noting that parameter names are case sensitive. A parameter could be a user input, which means that the parameter is passed from the pipeline layer or could be an input coming from an activity within the pipeline.

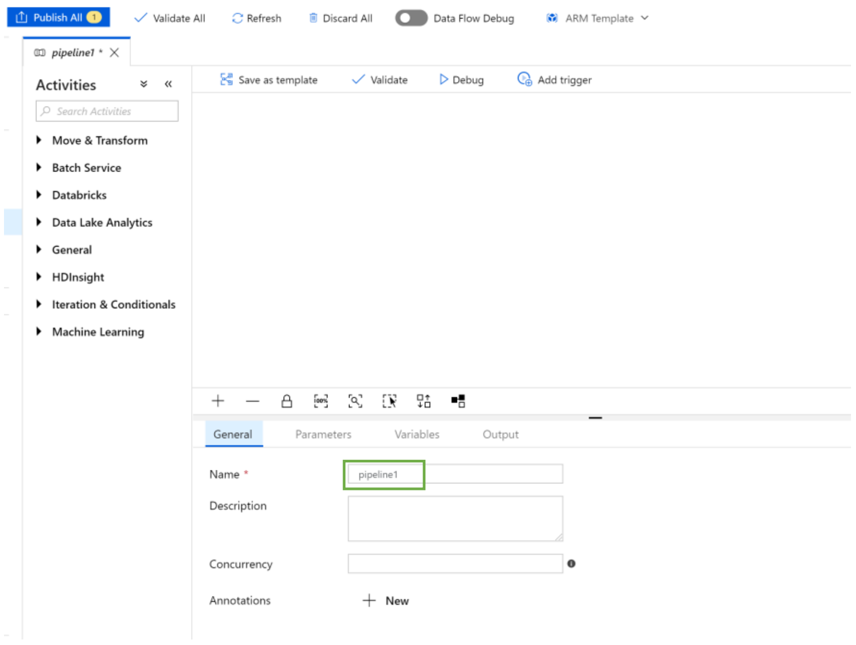
Passing parameter via pipeline

In this example, I am creating a pipeline that will use dynamic REST API URL to extract data in JSON format and move it to a blob store. Click "Author & Monitor" to create a new pipeline. A new tab will appear, and it will look like the one below-



The base URL we will be using is https://conferenceapi.azurewebsites.net/

Select "Create pipeline" icon. Rename the pipeline by replacing "pipeline1" with name of your choice. It is worth following some naming convention.



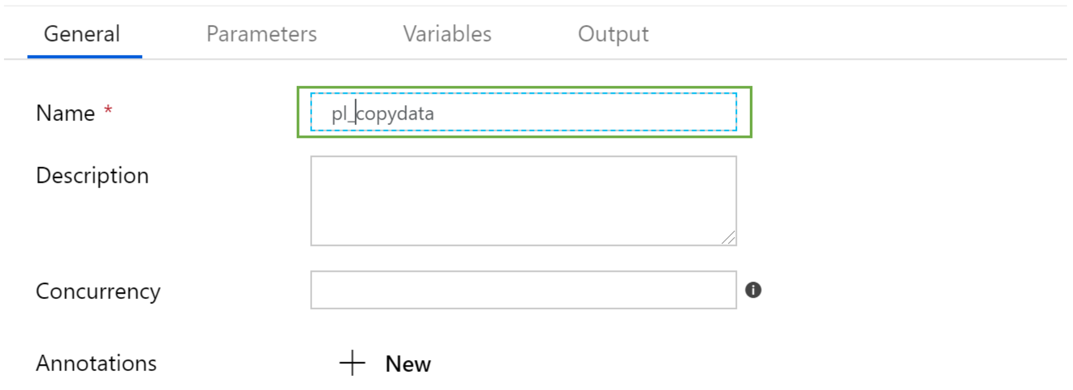
Name * pipeline1

Example of naming convention-

ADF Component	Naming Convention	Example
pipeline	pl_businessfunction_nnnn	pl_financereporting_0001*
dataset	ds_technologyname_nnnn	ds_sql_finance_0001*
activity	ac_techfunction_nnnn	ac_copy_blob_stg_sales_0001*
linked service	ls_connectiontype_nnnn	ls_sql_oreaserver_0001*

*we are using the numbering to get around limits such as maximum number of activities in a pipeline.

After renaming the pipeline, select the parameter tab



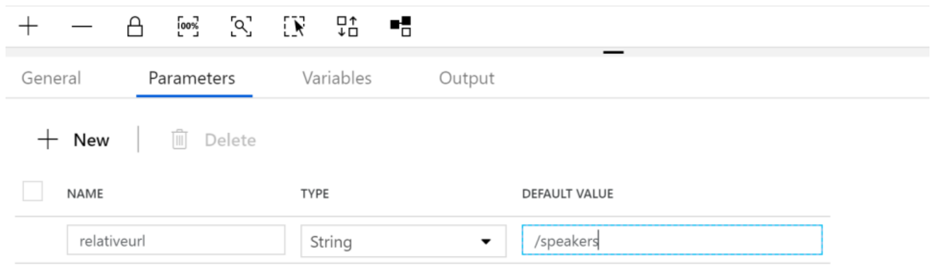
Name *: pl_copydata

Description:

Concurrency:

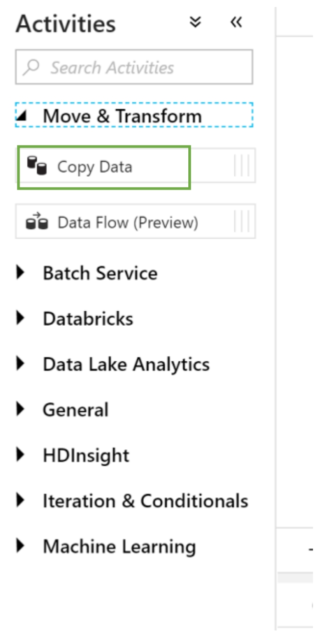
Annotations:

Once parameter tab is clicked, you can hit "+New" icon. Type in the name-**relativeurl**-with type as **string** and default value as **/speakers**.

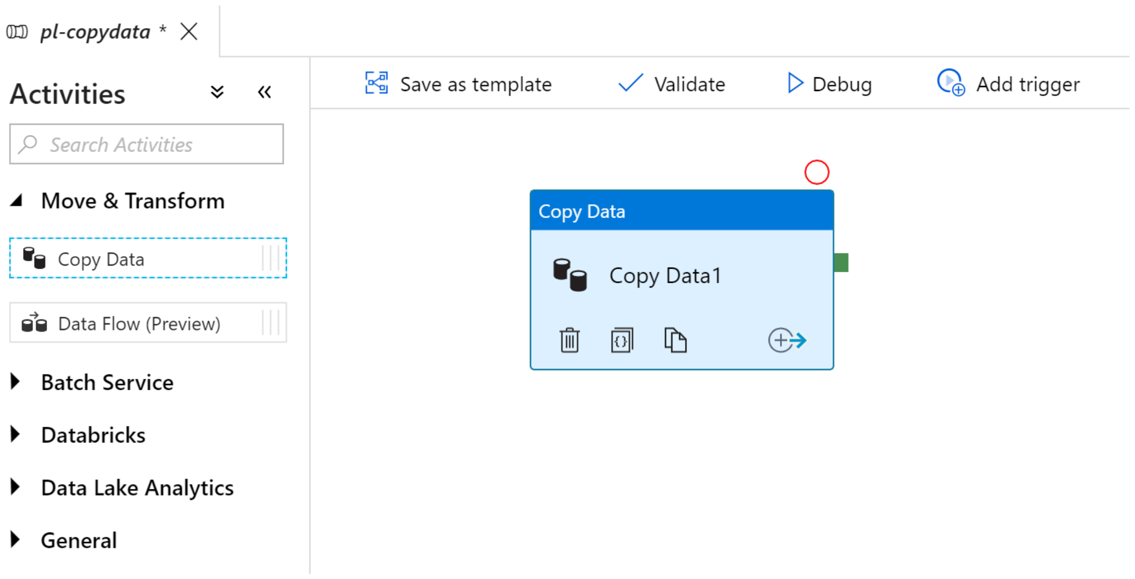


NAME	TYPE	DEFAULT VALUE
relativeurl	String	/speakers

Either use the search bar to find copy data activity or expand "Move & Transform" node to find copy data activity. Drag and drop this activity to the white canvas on your right



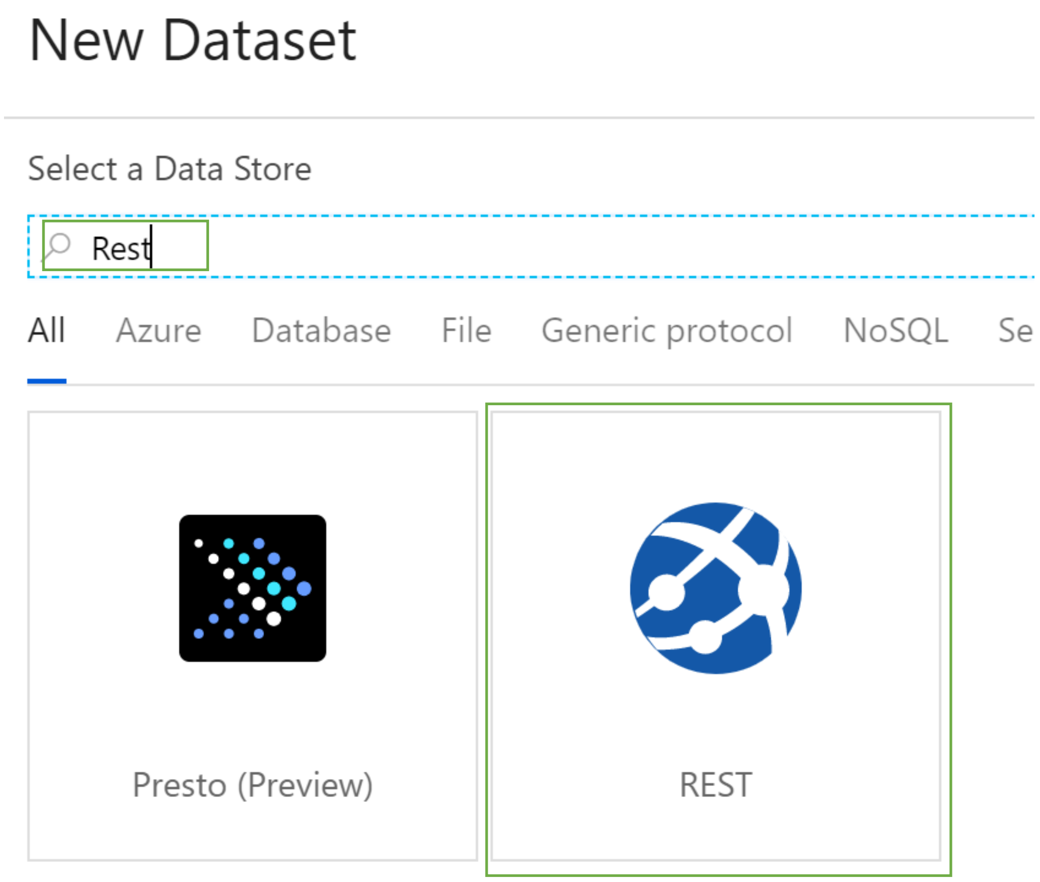
Once copied to the right pane, your screen will appear like the one below-



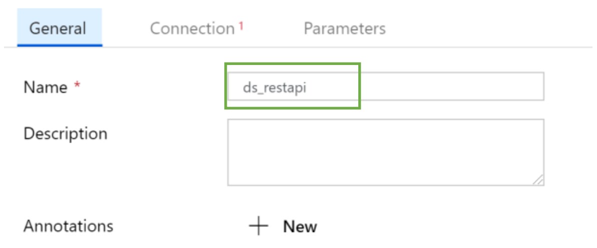
Within the copy activity, we need to define the source and sink. In this case, the source will be the REST API and the sink will be a blob store.

Defining Source

Click the **Source** tab and click the icon “**+New**”. When the list of new datasets appears, search for “Rest” and select “Rest”.



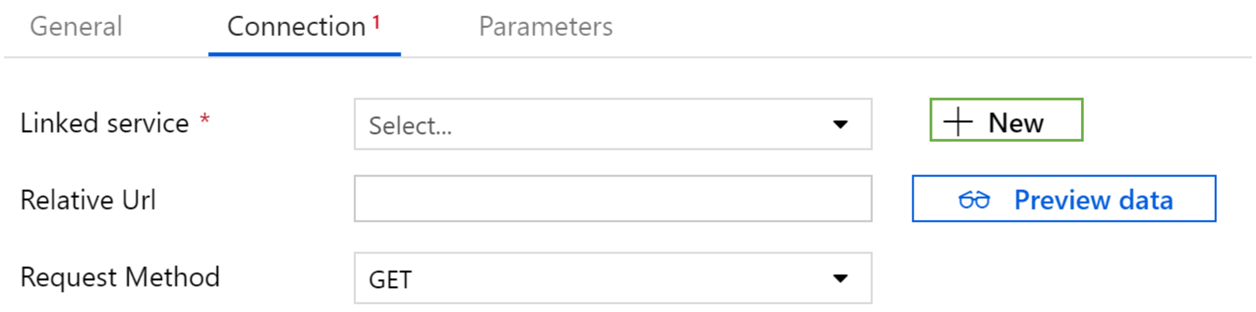
Set the name of the dataset in the general tab.



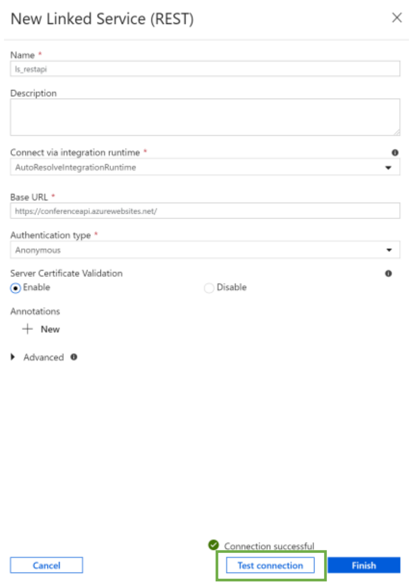
Name * ds_restapi

Description

Select the Connection tab, and create the linked service by clicking the “+New” icon



Fill the information as given below and hit the test connection button-


New Linked Service (REST)

Name *
ls_restapi

Description

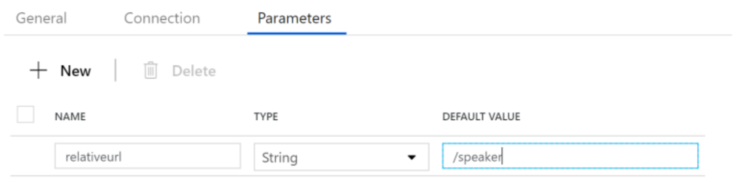
Connect via integration runtime *
AutoResolveIntegrationRuntime

Base URL *
https://conferenceapi.azurewebsites.net/

Authentication type *
Anonymous

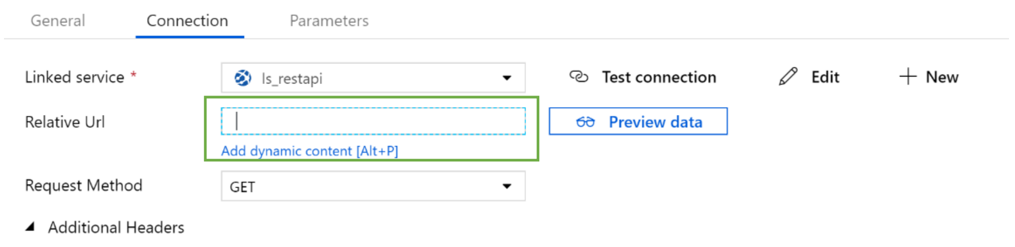
Server Certificate Validation
[✓ Enable] Disable

Click finish and select the parameters tab

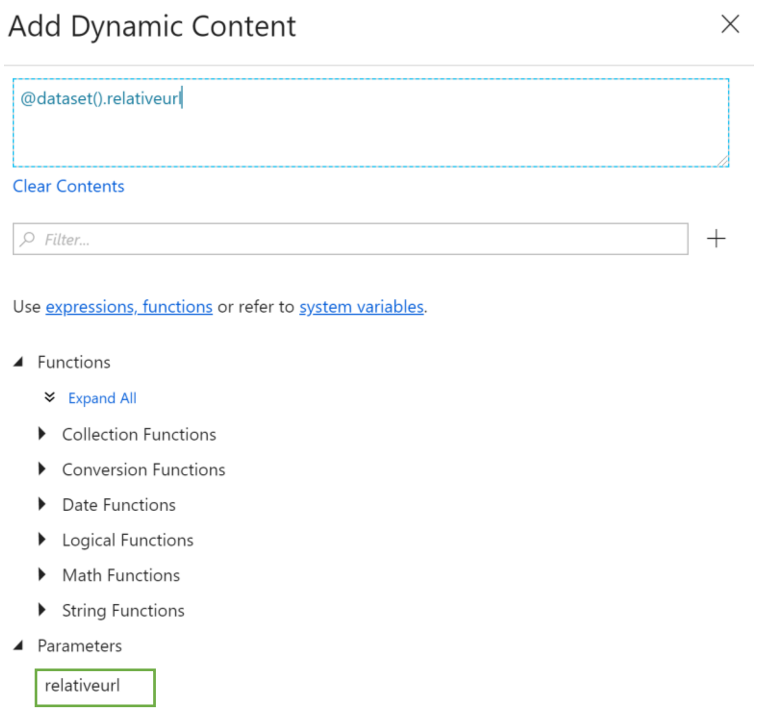


NAME	TYPE	DEFAULT VALUE
relativeurl	String	/speaker

Now let us parameterize the Relative Url

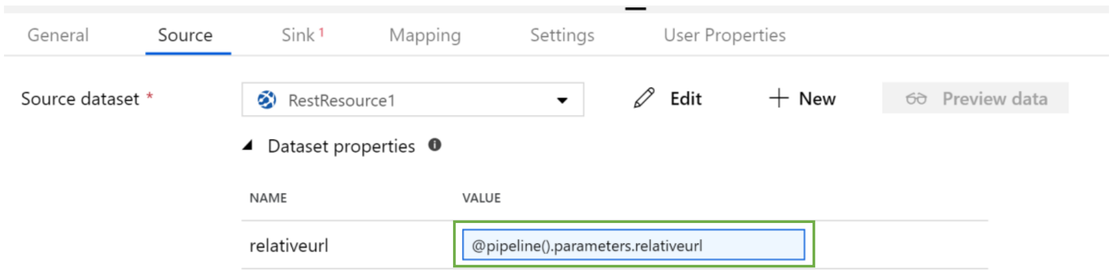


Any field that comes up with the “Add dynamic content” allows parameterization. We need to set the Relative Url to parameter that is coming from pipeline. Earlier, I stated that we need to use the fully qualified name. Double click “relativeurl” and click finish


Add Dynamic Content

```
@dataset().relativeurl
```

At the source page, set the dataset level parameter to the pipeline level parameter. **You cannot see the pipeline level parameter at the dataset level.** This is a very important thing to note.

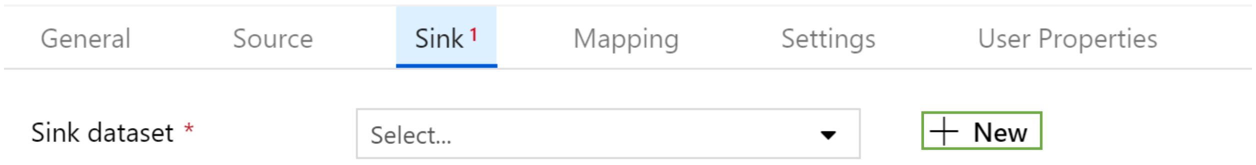


Source dataset: RestResource1

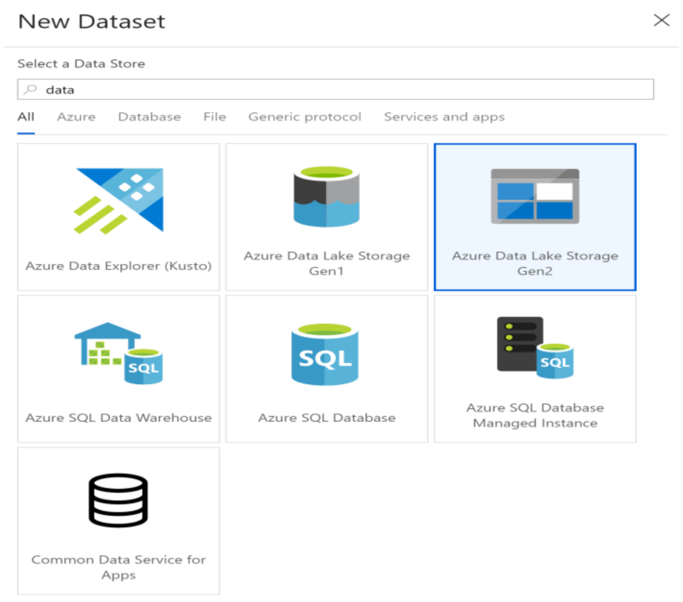
NAME	VALUE
relativeurl	@pipeline().parameters.relativeurl

Defining Sink

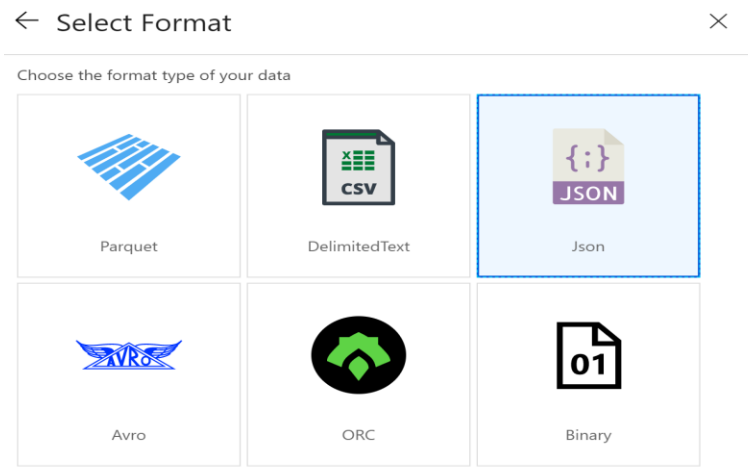
Click the **Sink** tab and click the icon “**+New”.**



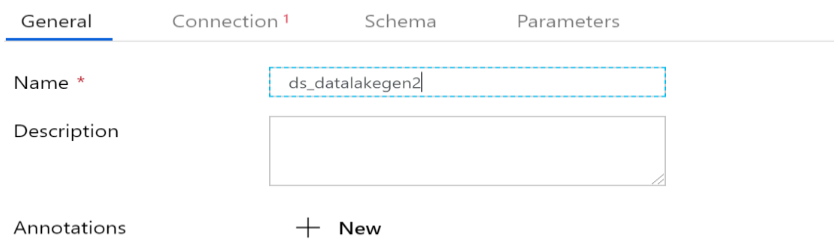
When the list of new datasets appears, search for “Azure data lake storage gen2” and select “Azure data lake storage gen2”.


New Dataset
Select a Data Store
data
All
Azure
Database
File
Generic protocol
Services and apps
Azure Data Explorer (Kusto)
Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2
Azure SQL Data Warehouse
Azure SQL Database
Azure SQL Database Managed Instance
Common Data Service for Apps

Select the format to be Json



In the general tab, fill the details using the naming convention that is used by your team

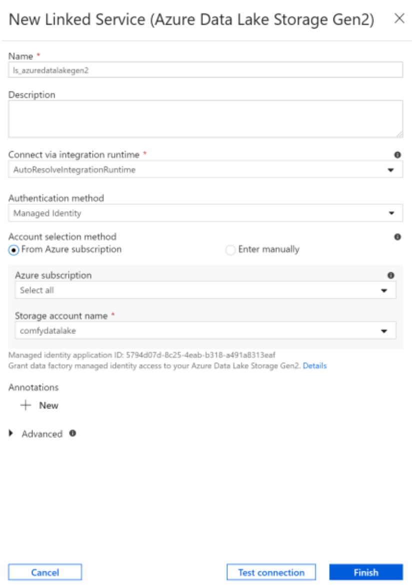

General | Connection 1 | Schema | Parameters

Name *: ds_datalakegen2

Description:

Annotations: + New

Click on the Connection tab and setup the linked service. Use managed identity where possible.


New Linked Service (Azure Data Lake Storage Gen2)

Name *: ls_azuredatalakegen2

Description:

Connect via integration runtime *: AutoResolveIntegrationRuntime

Authentication method: Managed Identity

Account selection method: [✓ From Azure subscription] Enter manually

Azure subscription: Select all

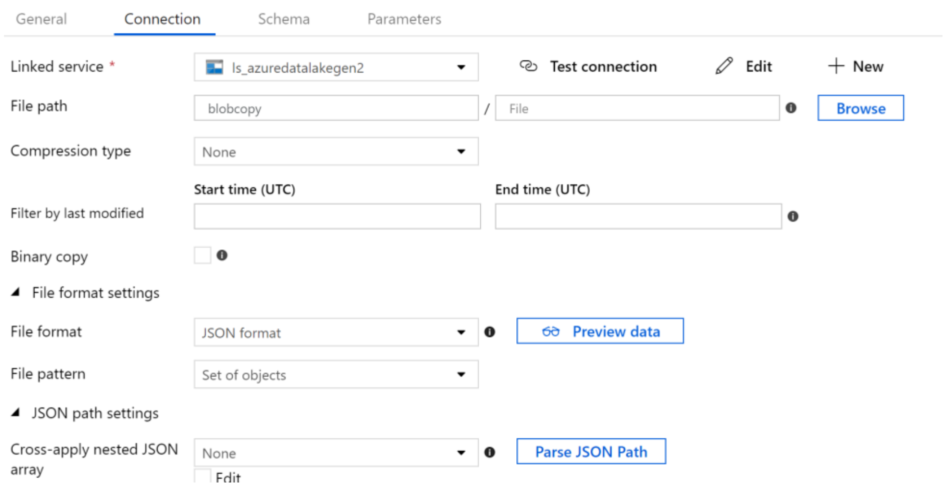
Storage account name *: comfydatalake

Managed identity application ID: 5794d07d-8c25-4eab-b318-a491a8313eaf
Grant data factory managed identity access to your Azure Data Lake Storage Gen2. Details

Annotations: + New

Advanced

Once the destination has been setup, the connection page will look like the one below-


General | Connection | Schema | Parameters

Linked service *: ls_azuredatalakegen2

File path: blobcopy / File

Compression type: None

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

Binary copy: ☐

File format settings

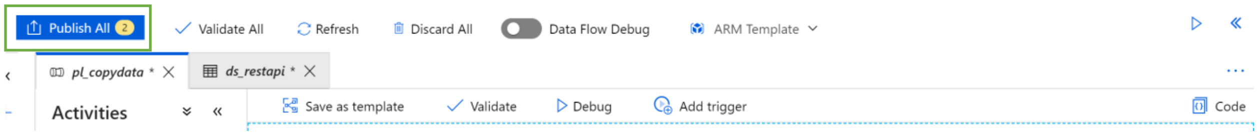
File format: JSON format

File pattern: Set of objects

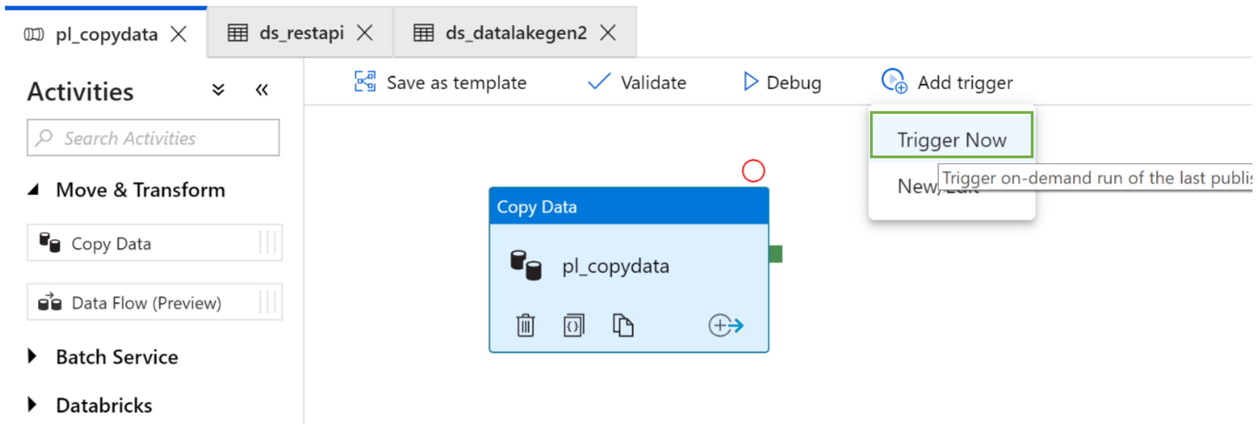
JSON path settings

Cross-apply nested JSON array: None

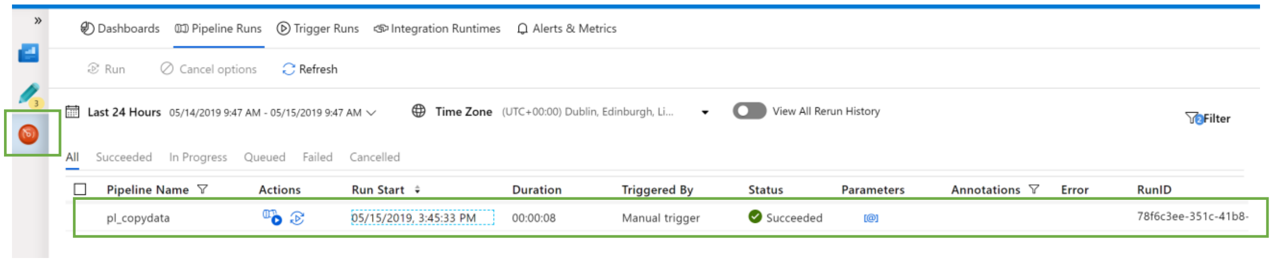
To save this, we need to click the publish all icon at the top left-



Once the pipeline has been published successfully, you can trigger the pipeline to test it. Select trigger now to trigger the pipeline.



Click on the monitor icon on the left and check whether the pipeline has completed successfully



Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Annotations	Error	RunID
pl_copydata		05/15/2019, 3:45:33 PM	00:00:08	Manual trigger	✓ Succeeded	[@]			78f6c3ee-351c-41b8-

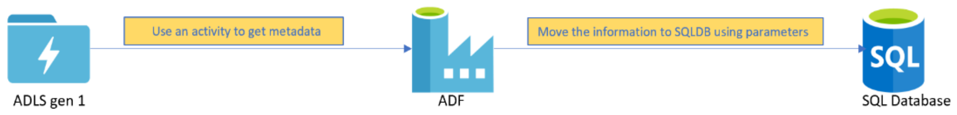
Passing parameter between activities

Before we even kick off this session, it is important to understand what an activity is. Activity is where a particular action is performed. The action could be to cleanse data or update a control table within SQL Database. An activity is within a pipeline, where the pipeline is a logical container having one or more activities.

We will be creating a pipeline that will copy the meta data information of a file in blob store and then loads this information into an Azure SQL database table. All the information will be moving between activities via parameters-the output parameters.

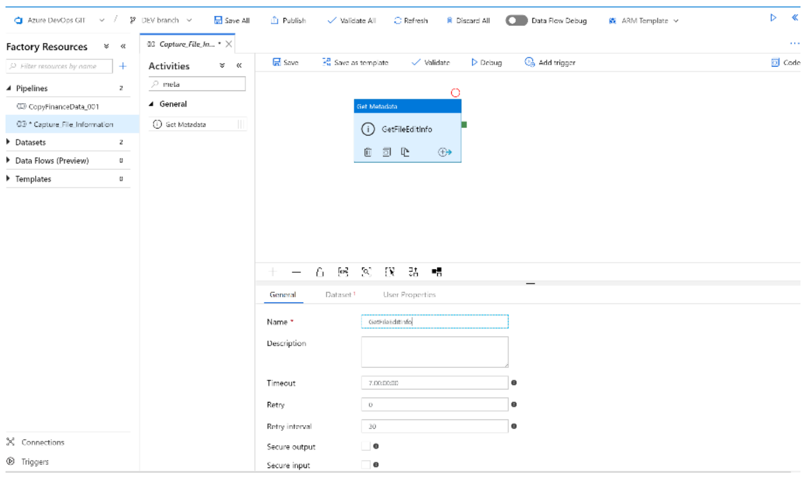
The architecture

We will follow a three-step process to show how to pass parameters between activities. The flat file currently exists on ADLS gen 1 storage. We use ADF activity to get the meta data about a file stored in ADLS gen 1. Once this information is available, we use parameters to pass the details to the next activity, which is a stored procedure within Azure SQL database.


Use an activity to get metadata
Move the information to SQLDB using parameters
ADLS gen 1
ADF
SQL
SQL Database

The step by step process

Go to a data factory and create a new pipeline and drag the activity “Get Metadata”.


Name * GetFileEditInfo

Description

Timeout 7.00:00:00

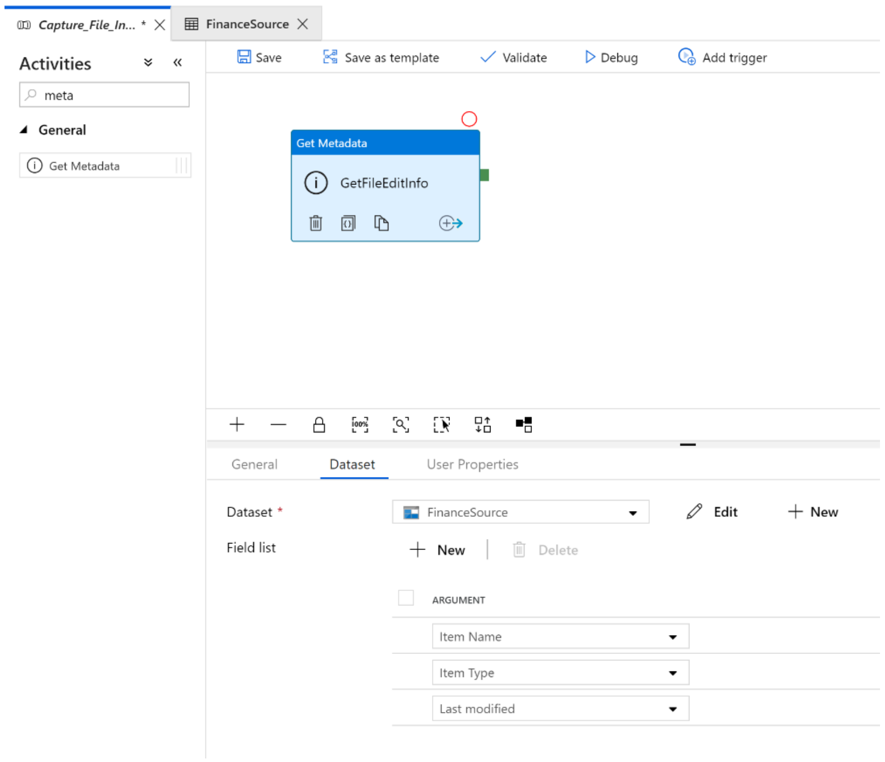
Retry 0

Retry interval 30

Secure output

Secure input

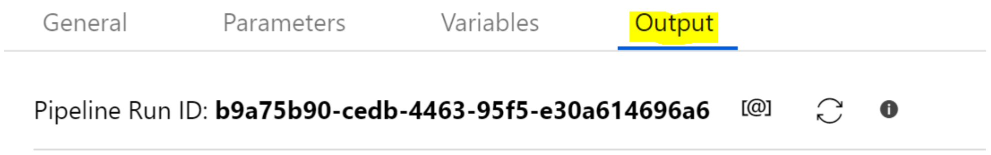
We plan to find the last modified date of a file in ADLS storage. Set the dataset to be the file stored in the ADLS storage. I am planning to capture three arguments- Item Name, Item Type and Last Modified.


Dataset * FinanceSource

Field list

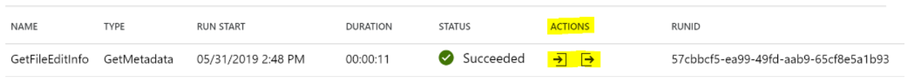
ARGUMENT
Item Name
Item Type
Last modified

To pass these arguments to the next activity, we need to first debug the pipeline. You will be able to see the outcome in the output tab.


General Parameters Variables Output

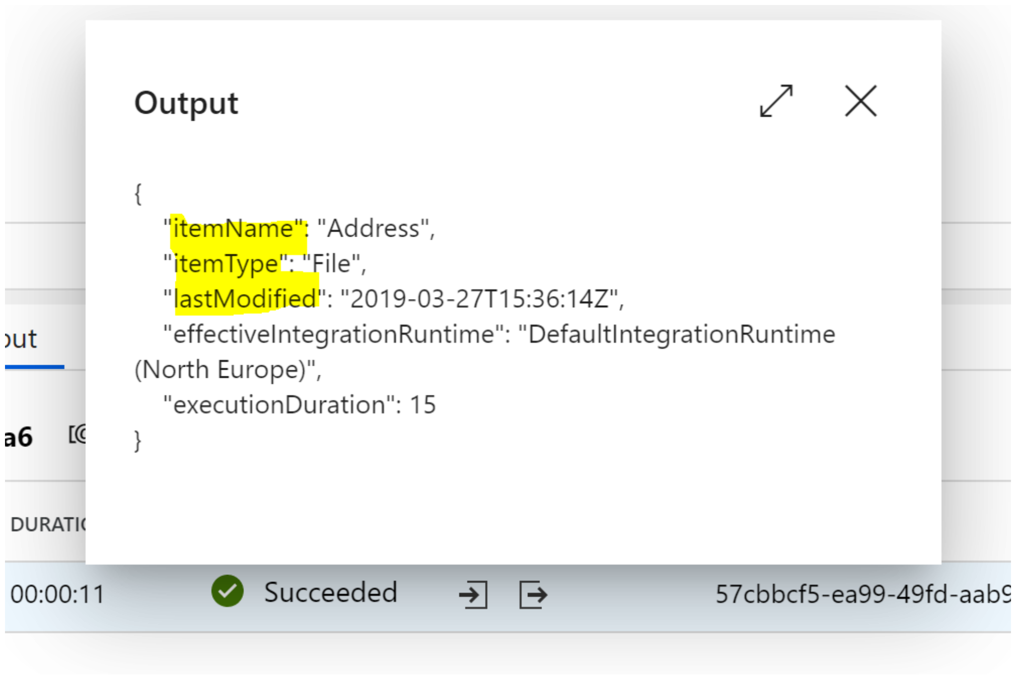
Pipeline Run ID: **b9a75b90-cedb-4463-95f5-e30a614696a6**

Once the execution completes, you have two option within action. One is the input and the other is output.



NAME	TYPE	RUN START	DURATION	STATUS	ACTIONS	RUNID
GetFileEditInfo	GetMetadata	05/31/2019 2:48 PM	00:00:11	Succeeded		57cbbcf5-ea99-49fd-aab9-65cf8e5a1b93

In this case, click on the ⇨ symbol and view whether the three arguments we selected are displayed.


Output

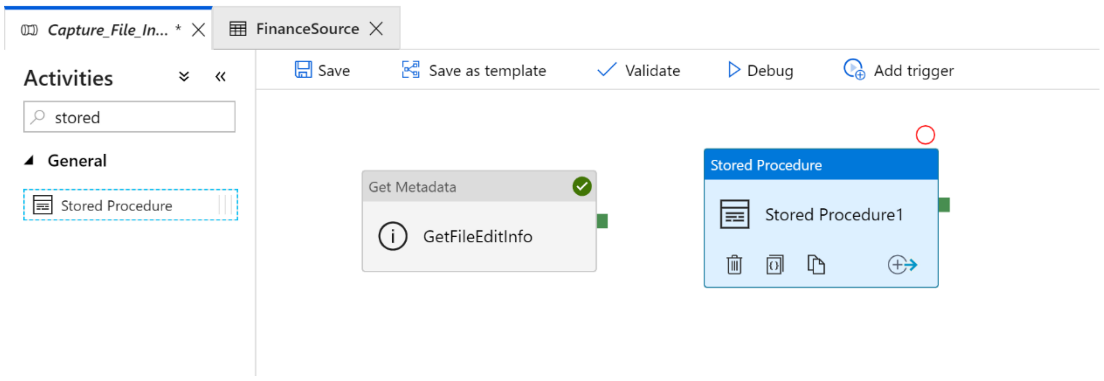
```
{
    "itemName": "Address",
    "itemType": "File",
    "lastModified": "2019-03-27T15:36:14Z",
    "effectiveIntegrationRuntime": "DefaultIntegrationRuntime
(North Europe)",
    "executionDuration": 15
}
```

Create a table and stored procedure within the Azure database that was created. The code is provided below.

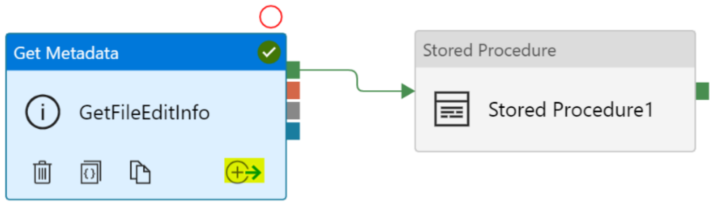
```
/*=====================================================
Create a control table that will store information
=====================================================*/
CREATE TABLE TB_FILE_METADATA
(
 I_ID INT IDENTITY(1,1),
 V_ITEM_NAME VARCHAR(255) NOT NULL,
 V_ITEM_TYPE VARCHAR(255) NOT NULL,
 D_LAST_MODIFIED DATETIME NOT NULL
)
```

```
-- =======================================================
-- Create Stored Procedure Template for Azure SQL Database
-- =======================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--
=========================================================================================
==
-- Author:      John Doe
-- Create Date: 31/05/2019
-- Description: This stored procedure will populate the data in the control table
--
=========================================================================================
===
CREATE PROCEDURE usp_populate_control_table
(
    -- Add the parameters for the stored procedure here
    @V_Item_Name varchar(255),
       @V_Item_Type varchar(255),
       @D_Last_Modified datetime
)
AS
BEGIN
       SET NOCOUNT ON

       BEGIN TRY

              INSERT INTO TB_FILE_METADATA
              (
                V_ITEM_NAME,
                V_ITEM_TYPE,
                D_LAST_MODIFIED
              )
              VALUES(@V_Item_Name,
                        @V_Item_Type,
                        @D_Last_Modified
              )
       END TRY
       BEGIN CATCH
              SELECT
                     ERROR_NUMBER() AS ErrorNumber
                  ,ERROR_MESSAGE() AS ErrorMessage;
       END CATCH
END
GO
```

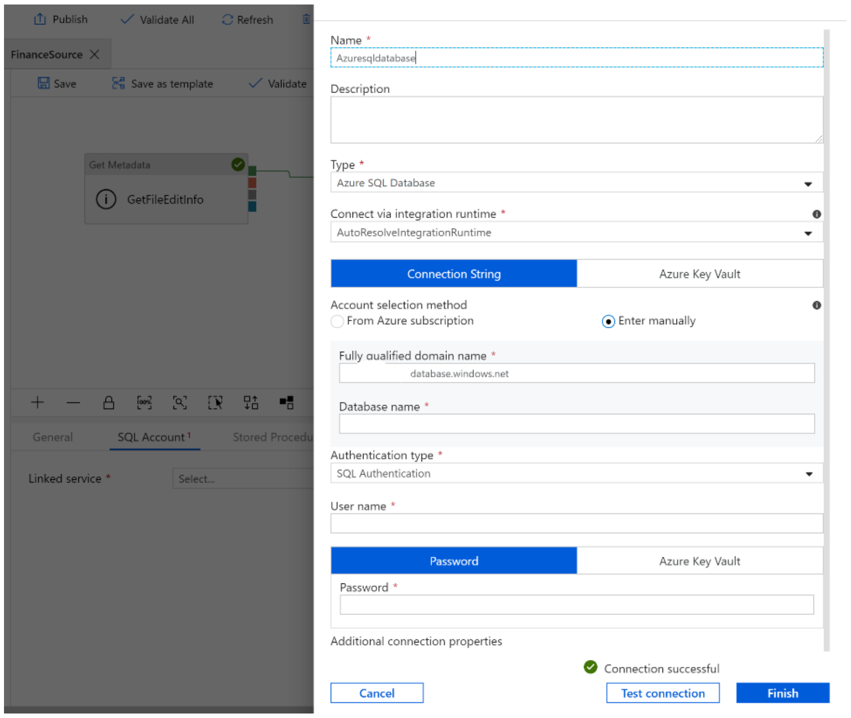
Drag the stored procedure activity into the ADF canvas.



Now connect the activities such that when "get metadata" activity completes successfully, you will execute the stored procedure. Just as in SSIS, you have option such as on failure, completion, success etc.



Setup the SQL account as shown below



Linked service * Select...

Name * Azuresqldatabase

Description

Type * Azure SQL Database

Connect via integration runtime * AutoResolveIntegrationRuntime

Connection String | Azure Key Vault

Account selection method
From Azure subscription [✓ Enter manually]

Fully qualified domain name * database.windows.net

Database name *

Authentication type * SQL Authentication

User name *

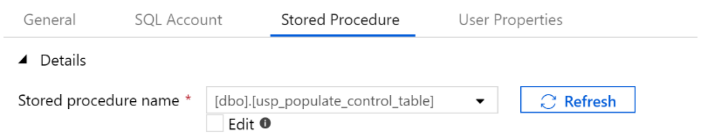
Password | Azure Key Vault

Password *

Additional connection properties

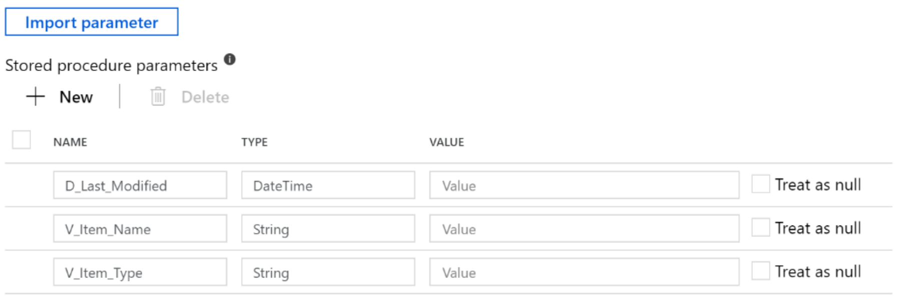
Connection successful

Select the stored procedure that was created “usp_populate_control_table”



Stored procedure name * [dbo].[usp_populate_control_table]

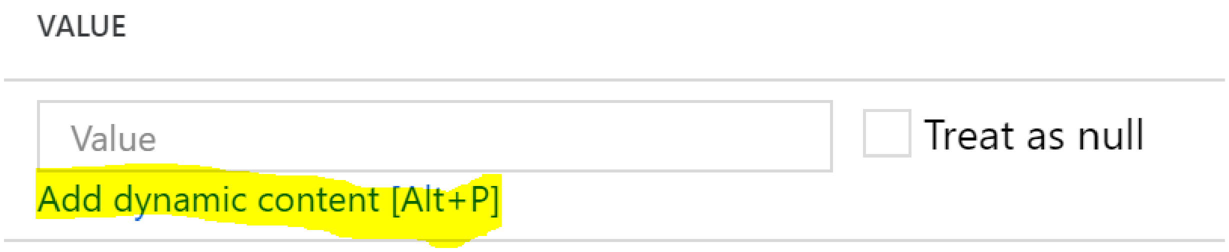
You can either import the parameters or manually enter the parameter. When you use import parameters, the input parameters for the stored procedure are identified automatically.



Stored procedure parameters

NAME	TYPE	VALUE	
D_Last_Modified	DateTime	Value	Treat as null
V_Item_Name	String	Value	Treat as null
V_Item_Type	String	Value	Treat as null

Click on the Value text box and press “add dynamic content”

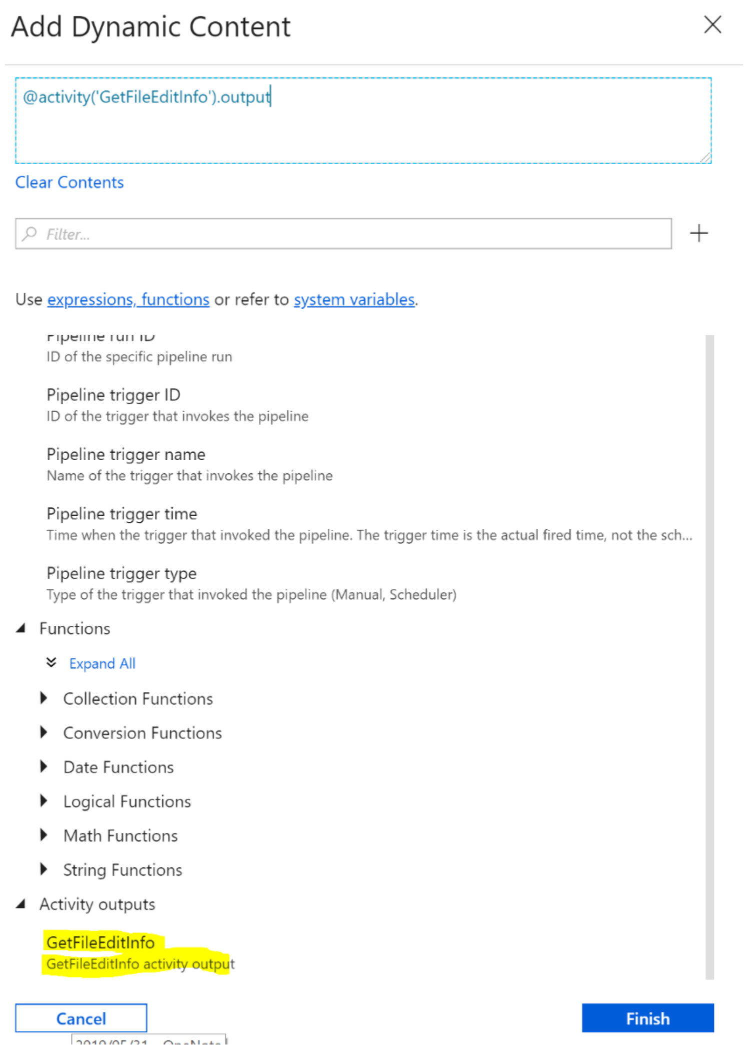

VALUE

Value

Add dynamic content [Alt+P]

Treat as null

Select the Activity output or you can manually type the value


Add Dynamic Content

```
@activity('GetFileEditInfo').output
```

Clear Contents

Use expressions, functions or refer to system variables.

Pipeline run ID
ID of the specific pipeline run

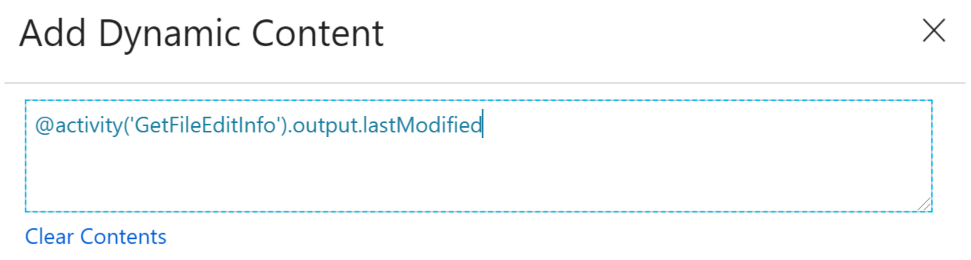
Pipeline trigger ID
ID of the trigger that invokes the pipeline

Pipeline trigger name
Name of the trigger that invokes the pipeline

Pipeline trigger time
Time when the trigger that invoked the pipeline. The trigger time is the actual fired time, not the sch...

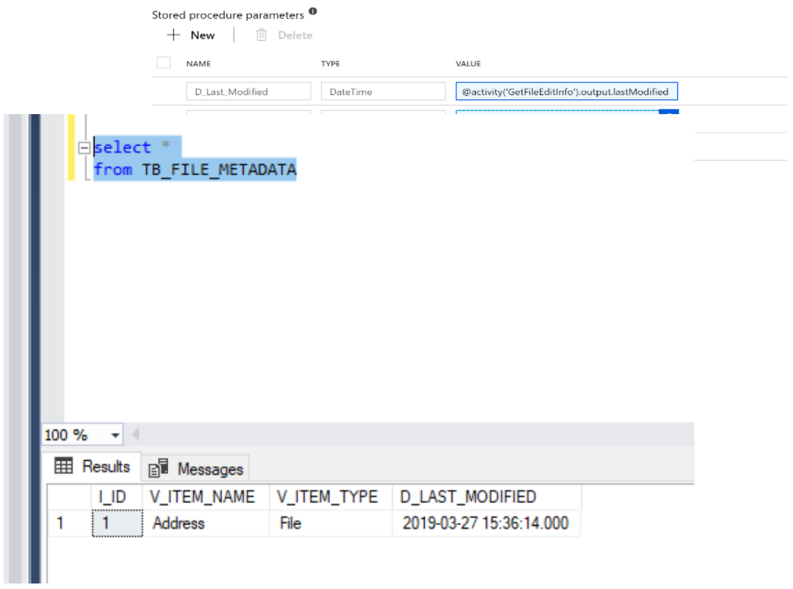
Pipeline trigger type
Type of the trigger that invoked the pipeline (Manual, Scheduler)

The first selection will look something like the one below.


Add Dynamic Content

```
@activity('GetFileEditInfo').output.lastModified
```

The names are case sensitive and once completed the screen will show up as follows



NAME	TYPE	VALUE
D_Last_Modified	DateTime	@activity('GetFileEditInfo').output.lastModified

```
select *
from TB_FILE_METADATA
```

	I_ID	V_ITEM_NAME	V_ITEM_TYPE	D_LAST_MODIFIED
1	1	Address	File	2019-03-27 15:36:14.000

You could now debug the pipeline and if everything looks good you will see that the table in azure SQL database is populated.

This indicates that the file was last modified in March 2019.