This article 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 .............................................................................................................. 2
  Passing parameter via pipeline ............................................................................................................................ 2
  Passing parameter between activities .................................................................................................................. 11
    The architecture ............................................................................................................................................... 11
    The step by step process ................................................................................................................................. 11

© 2019 Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.
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.
Example of naming convention -

<table>
<thead>
<tr>
<th>ADF Component</th>
<th>Naming Convention</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline</td>
<td>pl_businessfunction_nnnn</td>
<td>pl_financereporting_0001*</td>
</tr>
<tr>
<td>dataset</td>
<td>ds_technologynname_nnnn</td>
<td>ds_sql_finance_0001*</td>
</tr>
<tr>
<td>activity</td>
<td>ac_techfunction_nnnn</td>
<td>ac_copy_blob_stg_sales_0001*</td>
</tr>
<tr>
<td>linked service</td>
<td>ls_connectiontype_nnnn</td>
<td>ls_sql_oreaserver_0001*</td>
</tr>
</tbody>
</table>

*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

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

New Dataset

Select a Data Store

Set the name of the dataset in the general tab.

<table>
<thead>
<tr>
<th>General</th>
<th>Connection</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name *</td>
<td>ds_restapi</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annotations</td>
<td>+ New</td>
<td></td>
</tr>
</tbody>
</table>
Select the Connection tab, and create the linked service by clicking the “+New” icon

<table>
<thead>
<tr>
<th>General</th>
<th>Connection</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linked service *</td>
<td>Select...</td>
<td><img src="image" alt="New" /></td>
</tr>
<tr>
<td>Relative Url</td>
<td></td>
<td><img src="image" alt="Preview data" /></td>
</tr>
<tr>
<td>Request Method</td>
<td>GET</td>
<td><img src="image" alt="Preview data" /></td>
</tr>
</tbody>
</table>

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

**New Linked Service (REST)**

Name *

Description

Base URL: https://restapi.com/restfulapi

Authentication Type *

Anonymous

Server Certificate Validation

Enable

Annotations

Now let us parameterize the Relative Url

<table>
<thead>
<tr>
<th>General</th>
<th>Connection</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linked service *</td>
<td><img src="image" alt="Is_restapi" /></td>
<td><img src="image" alt="Test connection" /> <img src="image" alt="Edit" /> <img src="image" alt="New" /></td>
</tr>
<tr>
<td>Relative Url</td>
<td><img src="image" alt="Preview data" /></td>
<td><img src="image" alt="Preview data" /></td>
</tr>
<tr>
<td>Request Method</td>
<td>GET</td>
<td><img src="image" alt="Preview data" /></td>
</tr>
</tbody>
</table>

Click finish and select the parameters tab

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>relativeurl</td>
<td>String</td>
<td>[default value]</td>
</tr>
</tbody>
</table>
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

Clear Contents

Use expressions, functions or refer to system variables.

- Functions
  - Expand All
  - Collection Functions
  - Conversion Functions
  - Date Functions
  - Logical Functions
  - Math Functions
  - String Functions

- Parameters
  - 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.

<table>
<thead>
<tr>
<th>General</th>
<th>Source</th>
<th>Sink</th>
<th>Mapping</th>
<th>Settings</th>
<th>User Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source dataset</td>
<td><img src="edit" alt="Resource1" /></td>
<td></td>
<td></td>
<td></td>
<td>Preview data</td>
</tr>
<tr>
<td>Dataset properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>VALUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relativeurl</td>
<td>@pipeline().parameters.relativeurl</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Defining Sink**

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

<table>
<thead>
<tr>
<th>General</th>
<th>Source</th>
<th>Sink</th>
<th>Mapping</th>
<th>Settings</th>
<th>User Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sink dataset</td>
<td>Select...</td>
<td></td>
<td></td>
<td></td>
<td>New</td>
</tr>
</tbody>
</table>
When the list of new datasets appears, search for “Azure data lake storage gen2” and select “Azure data lake storage gen2”.

Select the format to be Json

In the general tab, fill the details using the naming convention that is used by your team.
Click on the Connection tab and setup the linked service. Use managed identity where possible.

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

The step by step process

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

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

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

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

```sql
/*===================================================
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,
```
Create Stored Procedure Template for Azure SQL Database

-- 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
(
    @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
Select the stored procedure that was created “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.

Click on the Value text box and press “add dynamic content”
Select the Activity output or you can manually type the value
The first selection will look something like the one below.

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

Clear Contents

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

```
select * from TB_FILE_METADATA
```

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.