WELCOME TO THE TECHNICAL COMMUNITY BLOG OF PAUL ANDREW

Data Platform Principal Consultant & Solution Architect

Azure Data Factory – Web Hook vs Web Activity

Posted on June 18, 2019July 15, 2020 (https://mrpaulandrew.files.wordpress.com/2019/06/web-hook-and-web-activities.png)

As Azure Data Factory continues to evolve as a powerful cloud orchestration service we need to update our knowledge and understanding of everything the service has to offer. Mainly, so we can make the right design decisions when developing complex, dynamic solution pipelines. In this post I want to explore and share the reasons for choosing the new **Web Hook Activity** compared to the older, standard **Web Activity**.

In short, the question you need to ask when developing a pipeline is; do I want my web process call(s) to be synchronous or asynchronous?

Answers to that question:

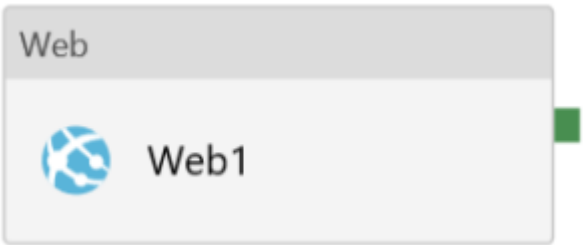
- If asynchronous, or if I doesn't matter, use a Web Activity. Fire and forget.
- If synchronous, use a Web Hook Activity. Fire and wait for completion.

Beyond that question lets go a little deeper and look at what's involved in implementing each activity to achieve this blocking/non-blocking behaviour.

Just before we dive in, I would like to caveat this technical understanding with a previous blog where I used a Web Activity to stop/start the SSIS IR and made the operation synchronous by adding an **Until Activity** that checked and waited for the Web Activity condition to complete. The post referenced here (https://mrpaulandrew.com/2018/06/19/how-to-control-the-ssis-ir-from-your-adfv2-pipelines/). The point being that we can enforce synchronous behaviour from any activity if we want. The new Web Hook activity now just gives us a convenient way to do it without much extra effort and additional operational calls.

Web Activity

(https://mrpaulandrew.files.wordpress.com/2019/06/web-activity.png)

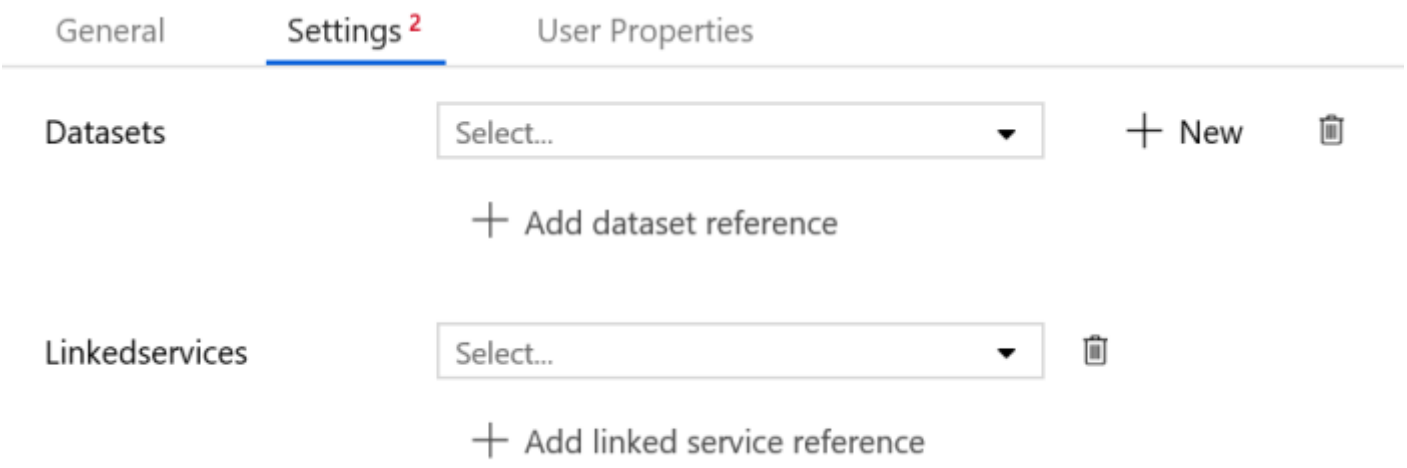


Firstly, let's quickly cover off the Web Activity. As most will know this has been available in Data Factory since the release of version 2. It's a great way to easily hit any API using **PUT**, **POST**, **GET** and **DELETE** methods.

Now the activity also supports Managed Service Identity (MSI) authentication which further undermines my above mentioned blog post, because we can get the bearer token from the Azure Management API on the fly without needing to make an extra call first.

Just to reiterate, this activity will make an asynchronous call to a given API and return a success or failure if no response is received within 1 minute. This timeout isn't configurable.

Beyond the process flow argument the other consideration you might have when choosing the web activity relates to your other Data Factory components. Unlike the web hook activity, the web activity offers the ability to pass in information for your Data Factory Linked Services and Datasets. This can be useful, for example, when uploading information to an endpoint from other parts of your pipeline. Datasets can be passed into the call as an array for the receiving service. Setting screen shot below.

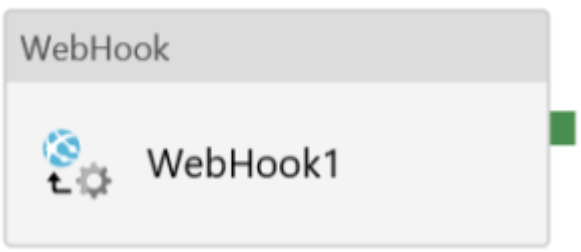


(https://mrpaulandrew.files.wordpress.com/2019/06/web-activity-settings.png)

That covers off the main points for the Web activity. Which I'm assuming people are probably familiar with.

Link to the Microsoft docs if you want to read more: https://docs.microsoft.com/en-us/azure/data-factory/control-flow-web-activity (https://docs.microsoft.com/en-us/azure/data-factory/control-flow-web-activity)

Web Hook Activity

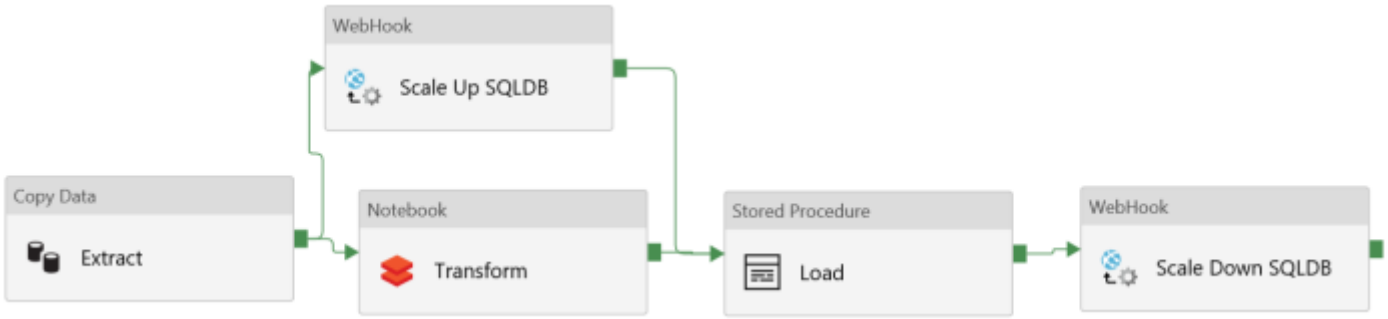


(https://mrpaulandrew.files.wordpress.com/2019/06/web-hook-activity.png)Next up, and the main reason for this blog post, the fairly new Web Hook Activity. This activity offers all the same functionality as its big brother web activity but with two main differences:

1. Other Data Factory Dataset and Linked Service resources can't currently be passed in at runtime. However, I expect this will change in the near future.
2. The activity offers the feature of a **call back URL**. Meaning it will request and wait for a response from the API hit before returning.

To show this important call back feature in action I have a very simple and hopefully common example to share.

Scenario: we have a pipeline doing some data transformation work, or whatever. The output dataset is going to be loaded into an Azure SQLDB table. However, before this happens, for Azure consumption cost efficiencies and loading times, we want to scale up the database tier at runtime. Then once data has been loaded we want to scale down the service. Maybe something like the below pipeline.



(https://mrpaulandrew.files.wordpress.com/2019/06/scale-sqldb-pipeline.png)

(https://mrpaulandrew.files.wordpress.com/2018/06/azureautomationicon1.png)To adjust the service tier of the SQLDB we can use a PowerShell cmdlet, shown below. Which we can wrap up in an Azure Automation Runbook. The Runbook can then have a Webhook added allowing us to hit the PowerShell scripts from a URL. This will be the API will call with our Web Hook Activity.

```
Set-AzureRmSqlDatabase `
    -Edition $targetEdition `
    -RequestedServiceObjectiveName $targetTier `
    -MaxSizeBytes $targetMaxSizeBytes | out-null
```

(https://mrpaulandrew.files.wordpress.com/2019/06/powershell-set-azuresqldb.png)

I won't go into the details on how to create the Runbook itself within Azure Automation and will assume most people are familiar with doing this. There are also plenty of tutorials out there. For this post the important thing to understand is that when the Data Factory pipeline runs the Web Hook activity (calling the Automation Webhook) it passes a supplementary set of values in the Body of the request. These values get appended onto any Body information you add via the activity settings, also, helpfully you can't see this extra information if you Debug the pipeline checking the activity input and outputs!

Example JSON of the full Body request as received via the Automation service:


```
{
    "WebhookName":"SQLDBScaler",
    "RequestBody":{
        "count": 1,
        "value": [
            {
                "EnvironmentName": "AzureCloud",
                "ResourceGroupName": "ResourceGroup01",
                "ServerName": "SQLInstance01",
                "DatabaseName": "Database01",
                "TargetEdition": "Standard",
                "TargetTier": "S3"
            }
        ],
    "effectiveIntegrationRuntime": "PaulsFunFactoryIR01 (North Europe)",
    "callBackUri": "https://PMNortheurope.svc.datafactory.azure.com/workflow/callback/##RUNID##?callbackUrl=##TOKEN##",
    "RequestHeader":
        {
            "Connection":"Keep-Alive",
            "Expect":"100-continue",
            "Host":"s9events.azure-automation.net",
            "x-ms-request-id":""
        }
}
```

(https://mrpaulandrew.files.wordpress.com/2019/06/callback-full-body-1.png)

The additional Body information, as you can see, includes the call back URI created by Data Factory during execution along with a bearer token to authenticate against the Data Factory API.

All that is required within your PowerShell Runbook is to capture this URI from the passed in Body and then Invoke a Web Request POST against it once all your other cmdlets have completed.

Depending on what other parameters you want to pass in and what other exception handling you want to put into the PowerShell the entire Runbook could be as simple as the below script:

```
#Get web hook call body information:
if ($WebhookData) {
    $parameters = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)
    if ($parameters.callBackUri)  {$callBackUri = $parameters.callBackUri}
}

$SizeInBytes = 250 * 1024 * 1024 * 1024

#Set SQLDB tier:
Set-AzureRmSqlDatabase `
    -Edition "Standard" `
    -RequestedServiceObjectiveName "S9" `
    -MaxSizeBytes $SizeInBytes | out-null

#Call back to ADF:
if ($callBackUri) {
    Invoke-WebRequest -Uri $callBackUri -Method POST
}
```

(https://mrpaulandrew.files.wordpress.com/2019/06/powershell-set-azuresqldb-with-callback.png)

Ultimately this behaviour means Data Factory will wait for the activity to complete until it receives the POST request to the call back URI. Making the pipeline activity synchronous.

As long as the API you hit can handle this behaviour and call back to Data Factory once complete the Web Hook activity does the 'rest' for you, pun intended 🙂

Link to the Microsoft docs if you want to read more: https://docs.microsoft.com/en-us/azure/data-factory/control-flow-webhook-activity (https://docs.microsoft.com/en-us/azure/data-factory/control-flow-webhook-activity)

I hope you found the above guide helpful for working with the Web Hook Activity.

Many thanks for reading.

Tagged Azure Data Factory, Dynamic ADF Pipelines, Web Activity, Web Hook Activity

Published by mrpaulandrew

Principal consultant and architect specialising in big data solutions on the Microsoft Azure cloud platform. Data engineering competencies include Azure Data Factory, Data Lake, Databricks, Stream Analytics, Event Hub, IoT Hub, Functions, Automation, Logic Apps and of course the complete SQL Server business intelligence stack. Many years' experience working within healthcare, retail and gaming verticals delivering analytics using industry leading methods and technical design patterns. STEM ambassador and very active member of the data platform community delivering training and technical sessions at conferences both nationally and internationally. Father, husband, swimmer, cyclist, runner, blood donor, geek, Lego and Star Wars fan!
View all posts by mrpaulandrew

17 thoughts on "Azure Data Factory – Web Hook vs Web Activity"

1. Pingback: Last Week Reading (2019-06-23) | SQLPlayer
2. **chris** says:
   July 5, 2019 at 4:49 am
   One difference I've found is that at time of writing, webhook does not seem to work with dynamic content in the body. I'm guessing a bug as the same dynamic content used in web activity is fine.

   Reply
   mrpaulandrew says:
   July 11, 2019 at 9:53 am
   Hey Chris, yes, it certainly sounds like it. I would recommend reaching out to Microsoft with an example to get this fixed. Thanks for the comment.

3. Reply
   Ankur Jain says:
   August 12, 2019 at 1:01 pm
   How do you handle callback URL in case of async call. If I have to call ASYNC REST api and it will return me result after 20 min (callback). How do I handle it?

4. Reply
Scott says:
August 26, 2019 at 7:08 pm
Any examples of using the callBackUri in a console webjob. I have used it successfully in an Azure Runbook to scale up my app service plan. But I am unable to find an example of a console application. I am calling my webjob from Azure Data Factory and I need to respond back after a long running console job with the callBackUri to notify the pipeline the webjob has completed before continuing processing of the rest of the pipeline.
Thanks

5. Reply
kbaig109 says:
September 12, 2019 at 9:25 pm
Now the activity also supports Managed Service Identity (MSI) authentication which further undermines my above mentioned blog post, because we can get the bearer token from the Azure Management API on the fly without needing to make an extra call first. ** do you have an example that shows this capability or can you quickly put something together for the community **
Appreciate that alot

6. Reply
Andrew Holowaty says:
September 30, 2019 at 4:21 pm
Mr. Paul Andrew
I am struggling to come up with a ADFv2 webhook to azure automation to refresh a cube. The Azure Automation works but I do not know what to use for the body and callback URI for my scenario. Your post is the only one I see that comes even close to what I need. Do you know where I can find examples and documentation on how to make this work.

Thanks!!
Andrew Holowaty

Reply
mrpaulandrew says:
September 30, 2019 at 4:50 pm
Sadly there isn't much, hence writing the post.
The callback URI is passed into the body automatically by ADF. It extends the JSON body from your activity. You should be able to see it in the activity output if you run it in debug mode. Once you have it just hit it from your Automation Runbook.

7. Reply
Andrew Holowaty says:
September 30, 2019 at 7:28 pm
So, if I understand correctly from above, the following line:
"callBackUri":
"https://PMNortheurope.svc.datafactory.azure.com/workflow/callback/##RUNID##?callbackUrl=##TOKEN##"

##RUNID## and ##TOKEN## will be obtained automatically from ADF. How did you derive the PMNortheurope?

Reply
mrpaulandrew says:

October 1, 2019 at 5:58 am
I didn't derive any of it. ADF generates it all and just appends it to the body of the request.

Reply

Md Nuruddin Pathan says:
October 29, 2019 at 1:23 pm
Hey Paul,

Can you please help me by giving an example of fetching that callBackUri in .Net console app and response back from there. I have one webjob written in C#, so, I want to implement the callBackUri mechanism on that webjob since I have a long-running workflow.

Waiting for your response….

Regards,
Nuruddin

8. **Md Nuruddin Pathan** says:
October 17, 2019 at 3:13 am
Hi Paul,

Would mind giving an example of how to handle the callBackUri from a .Net WebJob? without interacting with the runbook.

Thank & Regards,
Nuruddin

Reply

Md Nuruddin Pathan says:
November 1, 2019 at 6:15 am
Hi Paul,

It would be really helpful for me if you can give the solution, I am searching for solution for more than 2 weeks. Please help me!.

Regards,
Nuruddin

9. Reply
Amy says:
December 11, 2019 at 11:27 am
"Datasets can be passed into the call as an array for the receiving service"

I assume this means you can pass information from a dataset into the request to the web activity? If this is the cause, have you successful done this?

Reply

10. Pingback: Best Practices for Implementing Azure Data Factory – Welcome to the Technical
11. Community Blog of Paul Andrew
ansoneesan says:
May 26, 2020 at 1:46 am
Interesting article…thank you!

I'm trying to do something very basic in Data Factory, and for the life of me, I can't find the solution.

Literally, all I'm trying to do is process a couple of SQL queries, export the results to Data LAke storage, and then email the files. I have everything working EXCEPT the mail portion. I've createdd a logic app, and if I trigger the pipeline, it returns an error, but somehow the emails still get generated.

The error is:
"errorCode": "2108",
"message": "{\"error\":{\"code\":\"MissingApiVersionParameter\",\"message\":\"The api-version query parameter (?api-version=) is required for all requests.\"}}",

I'm assuming this is due to the METHOD I've chosen for the logic app activity. All I want to do is when the Copy Data task completes, send an email. This was so much easier in SSIS.

I know this question is off topic somewhat, but if you could provide some insight, that would be great!

Thank you

12. Reply
Ganesh Karri says:
July 1, 2020 at 10:38 pm
callbackuri doesn't work if the response is not received in 1 min. I am unable to understand how to send a response back to data factory without using callbackuri. I have a long running process which do not finish in 1 min. Azure documentation says, i need to send a accepted (202) response along with status url and retry after attributes but I am lost as to how to send a response back to data factory. If I use callbackuri, the pipeline was successful but I want the pipeline to wait until my process is finished.

Reply