Transforming JSON data with the help of Azure Data Factory - Part 3

posted by Rayis Imayev on April 06, 2020

(2020-Apr-06) Traditionally I would use data flows in Azure Data Factory (ADF) to flatten (transform) incoming JSON data for further processing. Recently I've found a very simple but very effective way to flatten incoming JSON data stream that may contain a flexible structure of data elements, and this won't require using data flow transformation steps.

Photo by from

Part 1: Transforming JSON to CSV with the help of Azure Data Factory - Mapping Data Flows
Part 2: Transforming JSON to CSV with the help of Azure Data Factory - Wrangling Data Flows

Here is my story :-)

Let's say I have the following JSON file that I want to parse one element (event) at the time:

A simple ADF pipeline can be created to read the content of this file and a stored procedure to call by passing individual JSON data elements as parameters for this procedure.

"ForEach container" with the following expression: *@activity('Get JSON data').output.firstRow.events*

Within my "ForEach" container I have also placed a Stored Procedure task and set 4 data elements from my incoming data stream as values for corresponding parameters.

However this approach will not work for all my incoming JSON events, it actually failed for the last one, since it didn't have both "*stop_time*" and "*last_update*" data elements.

Error message:

An easy way to fix this problem is to add missing data elements with empty values for the last event record, however, when we don't have control over incoming data, we need to adjust our data processing steps.

Solution:
We can check if "*stop_time*" and "*last_update*" data elements exist in the *@item* iteration dataset. If they don't exist, then we can replace them with other default values.

I'm not aware if there are built-in operators in ADF to do this, however, it still can be done by converting *@item* output into a string and then do a simple text search within this converted text line.

"*@if(contains(string(item()),'"stop_time":'),item().stop_time,null)*",

and initial *last_update* expression "*@item().stop_time*" can be replaced with

"*@if(contains(string(item()),'"last_update":'),item().last_update,utcNow())*"

After this quick fix, the whole ADF pipeline ran successfully.

So, using a simple data conversion with the help of String ADF function we can flatten JSON data into a string, and that explains the title of my blog post :-)

Reactions: funny (0) interesting (0) cool (0)

Daniel April 6, 2020 at 5:10 AM

Thanks for this post :)

REPLY

Unknown April 17, 2020 at 12:15 PM

You could probably also write it like this:
@coalesce(item()?.last_update,utcNow())

Notice the question mark for the null check. I've never seen it documented specifically for ADF, but it does work.

Rayis Imayev April 18, 2020 at 12:25 PM

Yes, I've tested the coalesce function as well, it worked in most cases, but there was one where it still failed. And I looked for a more stable solution to extract and search if a particular data element existed in incoming data stream. Thanks for your comment!

REPLY

Data Adventures

to read json data stored in the database?

Rayis Imayev May 18, 2020 at 11:24 AM

Technically you can store JSON value in a table column and then parse it either manually within the control flow or using Flatten transformation in Mapping Data Flows.

REPLY

Popular Posts

Working with Arrays in Azure Data Factory

(2019-June- 06) I remember that I had a professor at my university who would often encourage his students to learn and get more experienced with simple things first. Learn the basics in and out and then move forward to more complicated concepts and practices; that was his motto …

Continuous integration and delivery (CI/CD) in Azure Data Factory using DevOps and GitHub

Data Adventures

solutions within the same data factory workspace and maintain your combined development efforts in a central code repository. C …

Using Azure Functions in Azure Data Factory

(2020-Apr- 19) Creating a data solution with Azure Data Factory (ADF) may look like a straightforward process: you have incoming datasets, business rules of how to connect and change them and a final destination environment to save this transformed data. Very often your d …

Setting Variables in Azure Data Factory Pipelines

(2018-Oct-15) Working with Azure Data Factory you always tend to compare its functionality with well established ETL packages in SSIS. Data flow task have been recreated as Data Copy activities; logical components have found they cloud-based siblings; as well as new kids on t …

Data Adventures

(2020-Mar- 19) Recently, Microsoft introduced a new Flatten task to the existing set of powerful transformations available in the Azure Data Factory (ADF) Mapping Data Flows - https://docs.microsoft.com/en-us/azure/data-factory/data-flow-flatten . What this new task ...

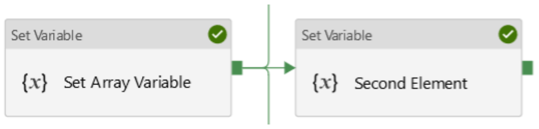
Append Variable activity in Azure Data Factory: Story of combining things together

(2018-Oct-29) There are only a few sentences in the official Microsoft web page that describe newly introduced activity task (Append Variable) to add a value to an existing array variable defined in Azure Data Factory - Append Variable Activity in Azure Data Factory But it signifi ...

Transforming JSON to CSV with the help of Azure Data Factory - Part 2 (Wrangling data flows)

(2020-Mar- 26) There are two ways to create data flows in Azure Data Factory (ADF): regular data flows also known as " Mapping Data Flows " and Power Query based data flows also known as " Wrangling Data Flows ", the latter data flow is still in preview, so do expect more adj ...

Data Adventures



Azure Data Factory: Extracting array first element

(2019-Apr- 28) Full credit goes to Microsoft for the latest efforts updating Azure products with new features and documenting corresponding changes for the end users. Azure Data Factory (ADF) is a great example of this. A user recently asked me a question on my previous blog post (…

Using Azure Data Factory Mapping Data Flows to populate Data Vault

(2019-May- 24) Data Flow as a data transformation engine has been introduced to the Microsoft Azure Data Factory (ADF) last year as a private feature preview . This privacy restriction has been lifted during the last Microsoft Build conference and Data Flow feature has becom …

Continuous integration and delivery (CI/CD) in Azure Data Factory using DevOps and GitHub - Part 2

(2020-Jan- 28) This blog post is a followup to my previous post about DevOps (CI/CD) for Azure Data Factory - Continuous integration and delivery (CI/CD) in Azure Data Factory using DevOps and GitHub - Part 1 where I described a method to design a Data Factory code re …

Data Adventures