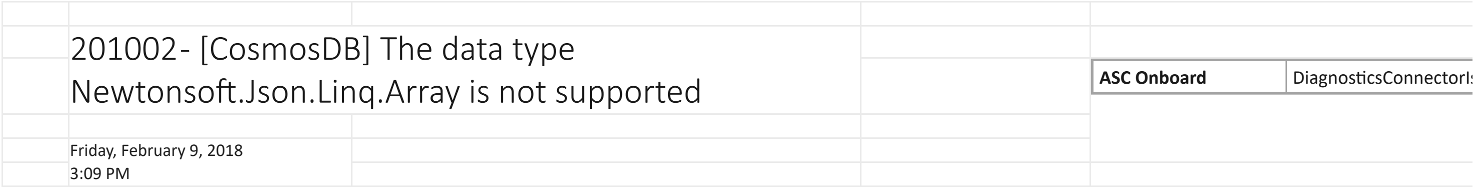
[CosmosDB] The data type Newtonsoft.Json.Linq.Array is not supported

Last updated by | Jackie Huang | Jan 4, 2022 at 12:24 AM PST



201002- [CosmosDB] The data type Newtonsoft.Json.Linq.Array is not supported

Friday, February 9, 2018
3:09 PM

ASC Onboard	DiagnosticsConnectorI

SME

Symptoms

Copy or preview from CosmosDB fail with the following error:

The data type Newtonsoft.Json.Linq.Array is not supported

Cause

1. When copy from CosmosDB to tabular destination, yes JArray is not supported, you need to decide whether to make it as a string or flatten it using CosmosDB query
2. When copy from CosmosDB to Json or CosmosDB, we support copy as is, but if customer specify structure or column mapping, (which means to treat it as tabular), this is not supported

Resolution

1. if customer wants to copy as is from CosmosDB to CosmosDB or Json binary, please suggest customer not to provide structure in both source/sink dataset. (Do not "Import schema" from Madrid, check "Copy as is" box in Copy Wizard)
2. If customer wants to copy to tabular destination as a string, please suggest customer to create a UDF in the collection and call the UDF in the query
3. If customer wants to flatten more than 1 arrays, please suggest customer to use a join query in CosmosDB
4. If customer wants to flatten 1 array in json, please suggest customer to use schema mapping and define collectionReference. https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-schema-and-type-mapping#alternative-schema-mapping
5. Below is an example for option #2 and #3 :

Example:

```
[
 {
  "propertyA": "test 1",
  "propertyB": [ 123, 456],
  "propertyC": [ "x", "y"]
 },
 {
  "propertyA": "test 2",
  "propertyB": [ 321, 654],
  "propertyC": [ "w", "v"]
 }
]
```

There are 2 scenarios:

1. **Customer want to expand his array to multiple record like this (option #3):**

propertyA	propertyB	propertyC
test 1	123	x
test 1	123	y
test 1	456	x
test 1	456	y
test 2	321	w
test 2	321	v
test 2	654	w
test 2	654	v

Solution:
In this case, customer can use DocumentDB query with JOIN to achieve this: SELECT c.propertyA, array1 as propertyB, array2 as propertyC FROM c JOIN array1 in c.propertyB JOIN array2 in c.propertyC

2. **Customer just want to store his array as a string to a single column, like this (option #2):**

propertyA	**propertyB**	**propertyC**
Test 1	[123, 456]	["x", "y"]
Test 2	[321, 654]	["w", "v"]

Solution:
In this case, customer can create a user defined function in their collection, and use this function in their query to achieve this.

User define function:

```
function ArrayToString (ts) {
  return JSON.stringify(ts);
}
```

Query:

SELECT c.propertyA, udf.ArrayToString(c.propertyB) as propertyB, udf.ArrayToString(c.propertyC) as propertyC FROM c

More Information

More sample for JOIN:

```
{
   "id": "00000000-0000-0000-0000-000000000000000",
   "_rid": "Zao3AIMSTwISAAAAAAAAAA==",
   "_self": "dbs/Zao3AA==/colls/Zao3AIMSTwI=/docs/Zao3AIMSTwISAAAAAAAAAA==/",
   "_etag": "\"020070f1-0000-0000-0000-5b316e8e0000\"",
   "Actions": [
   {
    "data": "abc"
   },
   {
    "data": "def"
   }
  ],
   "_attachments": "attachments/",
   "_ts": 1529966222
}
```

SELECT c.id, c._rid, c._self, c._etag, actionArray.data, c._attachments, c._ts FROM c JOIN actionArray in c.Actions

Result:

```
[
  {
     "id": "00000000-0000-0000-0000-0000000000000000",
     "_rid": "vQJSAOp5gAIBAAAAAAAAAA==",
     "_self": "dbs/vQJSAA==/colls/vQJSAOp5gAI=/docs/vQJSAOp5gAIBAAAAAAAAAA==/",
     "_etag": "\"2400ff6c-0000-0000-0000-5b4593510000\"",
     "data": "abc",
     "_attachments": "attachments/",
     "_ts": 1531286353
  },
  {
     "id": "00000000-0000-0000-0000-0000000000000000",
     "_rid": "vQJSAOp5gAIBAAAAAAAAAA==",
     "_self": "dbs/vQJSAA==/colls/vQJSAOp5gAI=/docs/vQJSAOp5gAIBAAAAAAAAAA==/",
     "_etag": "\"2400ff6c-0000-0000-0000-5b4593510000\"",
     "data": "def",
     "_attachments": "attachments/",
     "_ts": 1531286353
  }
]
```

Tags	CosmosDB

Created with Microsoft OneNote 2016.

How good have you found this content?