Convert Table to JSON

Derek Pascarella
Updated 1 month ago

Unfollow

Applies To: Ayehu NG

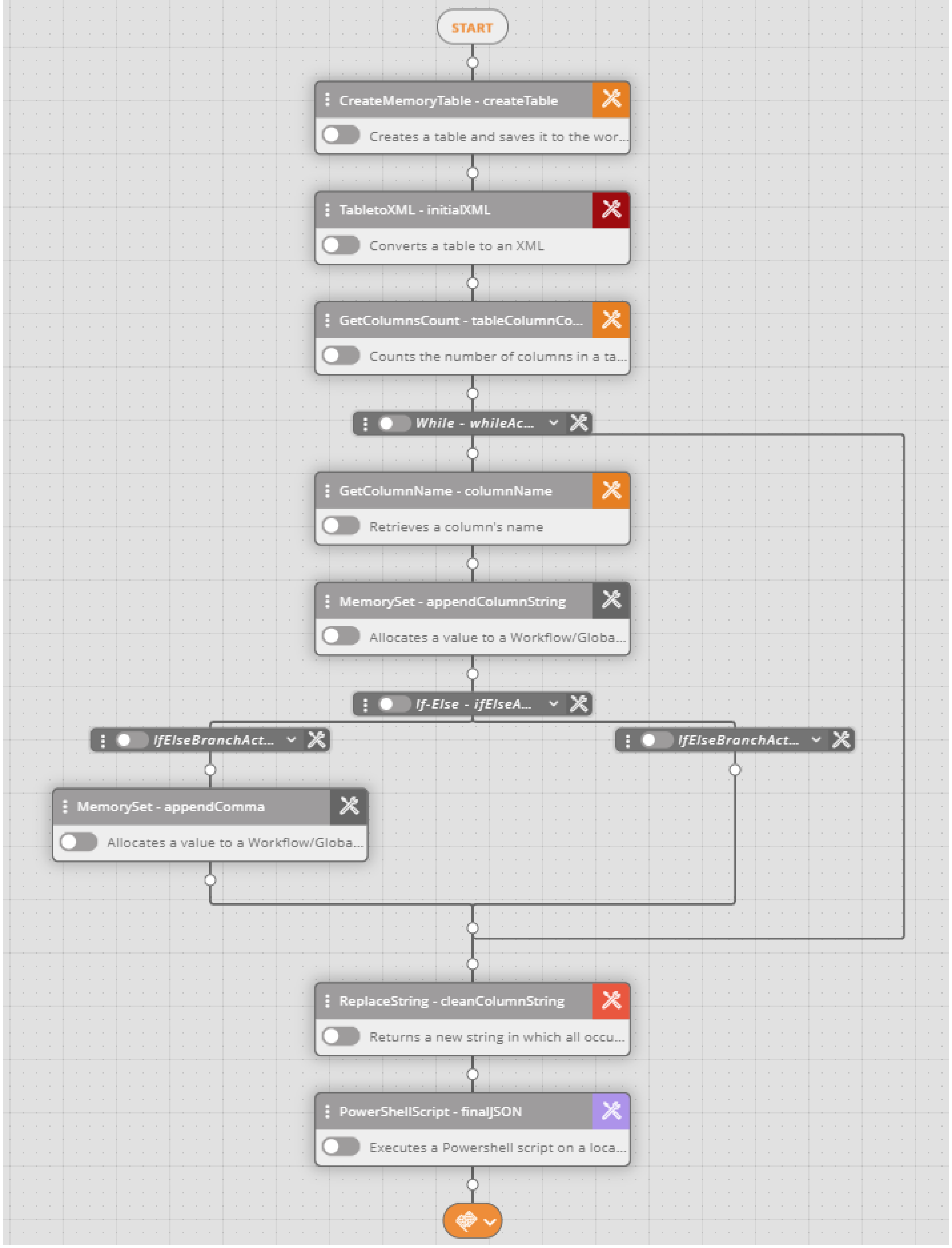
Description

Ayehu NG utilizes tables as a useful and easy-to-read/manipulate format for both storing and retrieving data. Other formats are also available on the platform, like JSON and XML. Although there exists an easy and convenient **Table to XML** activity for converting XML to **Ayehu NG**'s native ResultSet table format, no such equivalent is available for converting a table to JSON.

The following tutorial outlines the simple steps needed to convert a table to JSON in an **Ayehu NG** workflow. Please note that you can also use this workflow template as a child workflow, which can be called from a parent workflow, so that it's reusable all throughout any of your workflows. For more information on parent and child workflows, see the following article: https://support.ayehu.com/hc/en-us/articles/360008089093

Workflow Overview

Below is a screenshot of an **Ayehu NG** workflow that creates a sample table and then converts it to JSON. You can also download an export of this workflow attached to this article or on the Ayehu GitHub at https://github.com/Ayehu/custom-workflows/tree/master/Table%20to%20JSON.


START
CreateMemoryTable - createTable
Creates a table and saves it to the wor...
TabletoXML - initialXML
Converts a table to an XML
GetColumnsCount - tableColumnCo...
Counts the number of columns in a ta...
While - whileAc...
GetColumnName - columnName
Retrieves a column's name
MemorySet - appendColumnString
Allocates a value to a Workflow/Globa...
If-Else - ifElseA...
IfElseBranchAct...
IfElseBranchAct...
MemorySet - appendComma
Allocates a value to a Workflow/Globa...
ReplaceString - cleanColumnString
Returns a new string in which all occu...
PowerShellScript - finalJSON
Executes a Powershell script on a loca...

Click image to view full-sized version.

The workflow achieves this by taking the following steps:

1. Converts the table to XML (**TableToXML** activity).

2. Stores the number of columns from the table (**GetColumnsCount** activity).
3. Loops through each column and appends its name to a variable, with each column name separated by commas (e.g. Col1,Col2,Col3).
4. Executes a Powershell script to convert the XML to JSON (**PowerShellScript** activity).

Activity Configuration

Now, let's look at each activity in this workflow, step-by-step. By doing so, you'll be able to follow along and implement these activities into your own workflows wherever you need to convert a table to JSON.

Our first step is to create a new table named **myTable** using the **CreateMemoryTable** activity. In your workflow, you'll already have a table available. Here is the example we will work with in this tutorial.

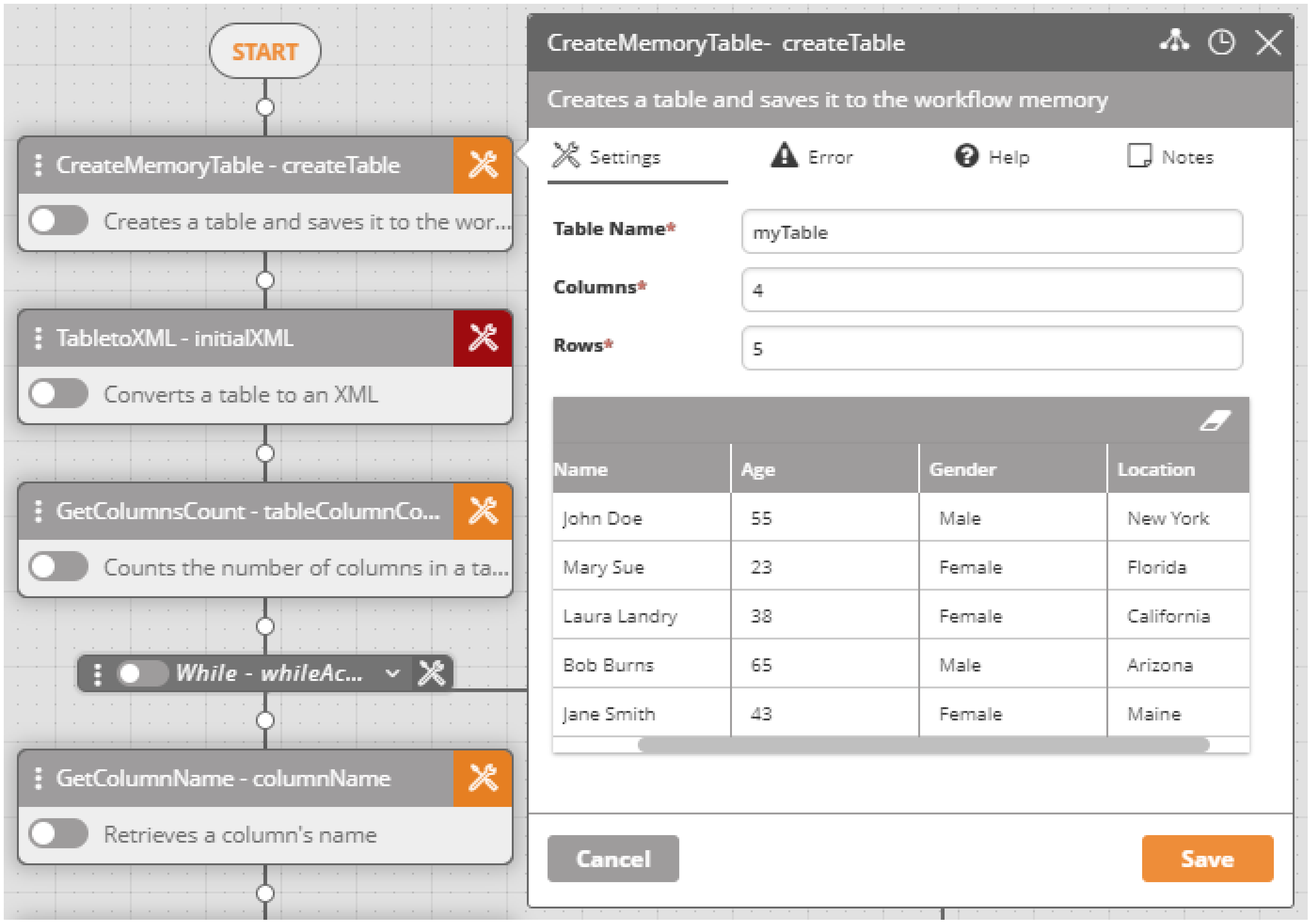

START
CreateMemoryTable - createTable
Creates a table and saves it to the wor...
TabletoXML - initialXML
Converts a table to an XML
GetColumnsCount - tableColumnCo...
Counts the number of columns in a ta...
While - whileAc...
GetColumnName - columnName
Retrieves a column's name
CreateMemoryTable- createTable
Creates a table and saves it to the workflow memory
Settings
Error
Help
Notes
Cancel
Save

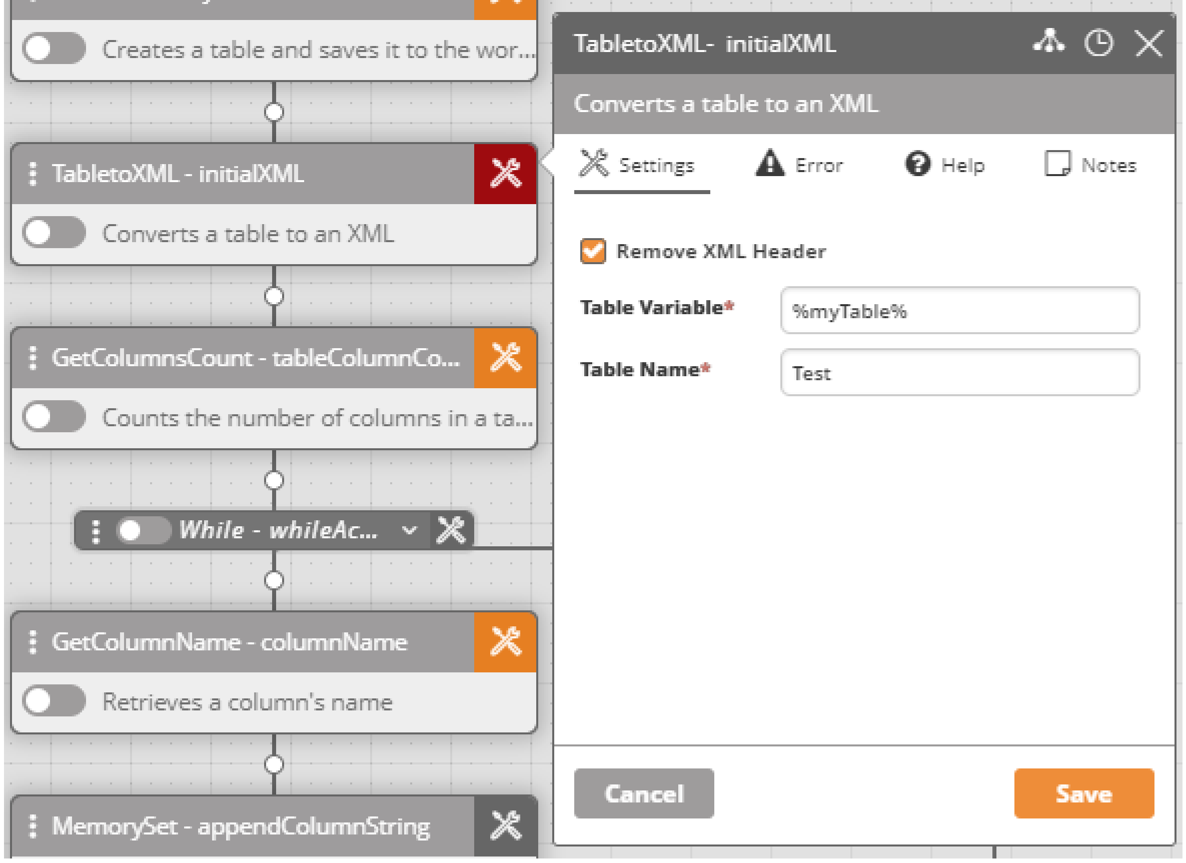
Table Name* myTable

Columns* 4

Rows* 5

Name	Age	Gender	Location
John Doe	55	Male	New York
Mary Sue	23	Female	Florida
Laura Landry	38	Female	California
Bob Burns	65	Male	Arizona
Jane Smith	43	Female	Maine

Our next step is to use the **TableToXML** activity to convert **myTable** to XML format. Be sure to check the **Remove XML Header** checkbox.


Creates a table and saves it to the wor...
TabletoXML - initialXML
Converts a table to an XML
GetColumnsCount - tableColumnCo...
Counts the number of columns in a ta...
While - whileAc...
GetColumnName - columnName
Retrieves a column's name
MemorySet - appendColumnString
TabletoXML- initialXML
Converts a table to an XML
Settings
Error
Help
Notes
Cancel
Save

Remove XML Header [✓]

Table Variable* %myTable%

Table Name* Test

Next, we use a **GetColumnsCount** activity to store the number of columns from **myTable**.

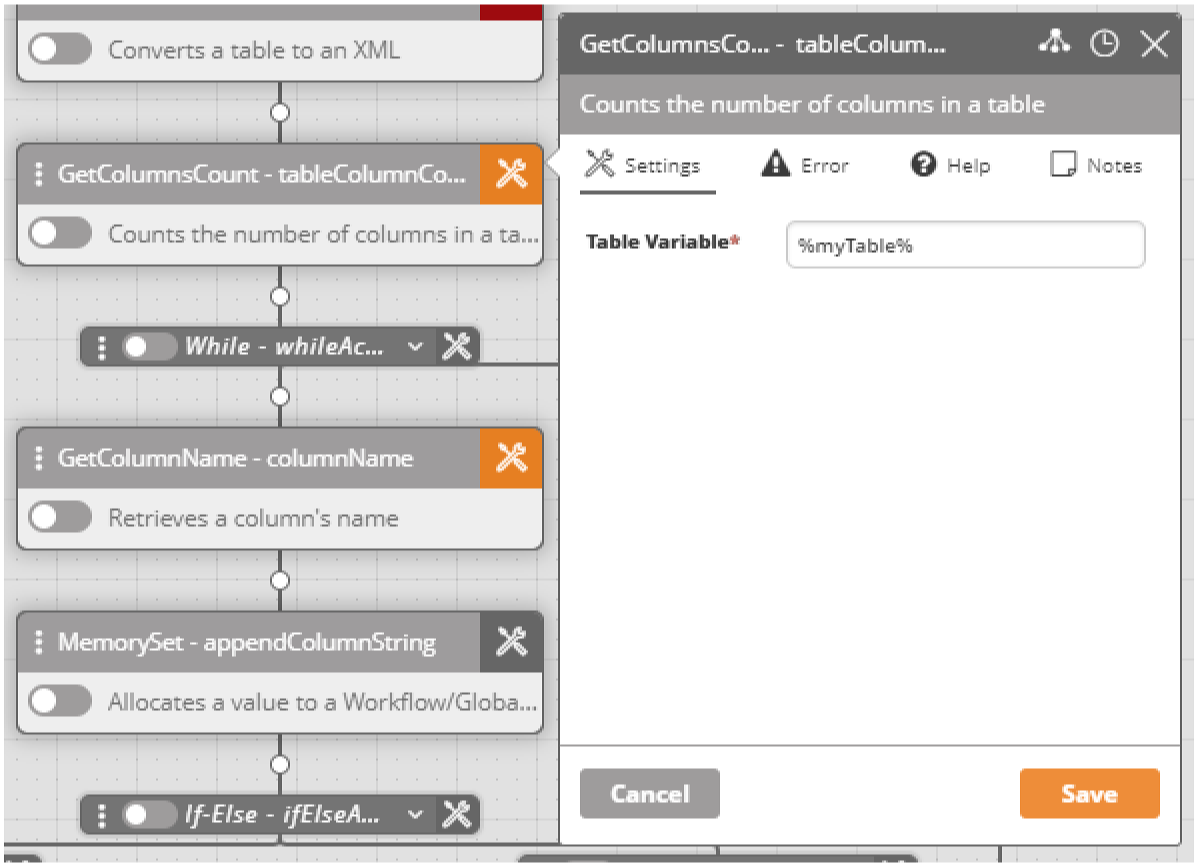
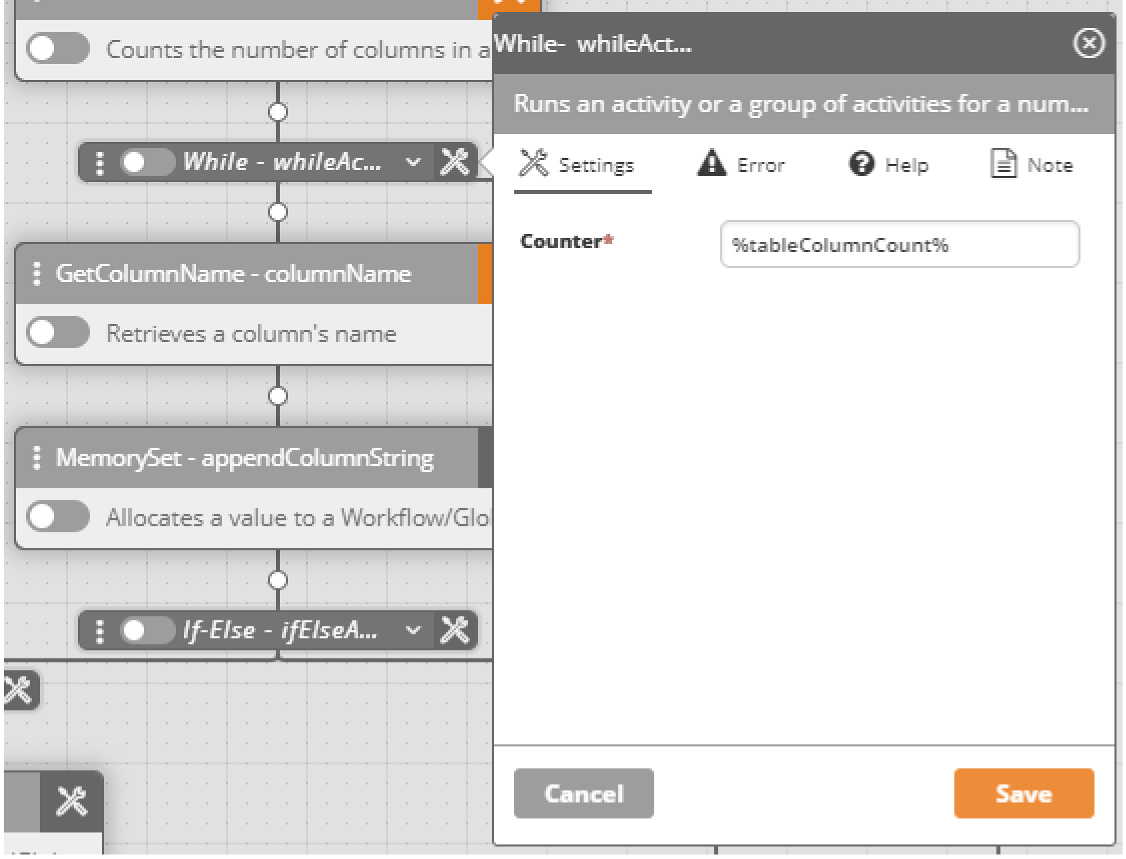

Converts a table to an XML
GetColumnsCount - tableColumnCo...
Counts the number of columns in a ta...
While - whileAc...
GetColumnName - columnName
Retrieves a column's name
MemorySet - appendColumnString
Allocates a value to a Workflow/Globa...
If-Else - ifElseA...
GetColumnsCo... - tableColum...
Counts the number of columns in a table
Settings
Error
Help
Notes
Cancel
Save

Table Variable* %myTable%

We then create a while-loop that iterates through **myTable** based on the number of columns found.


Counts the number of columns in a
While - whileAc...
GetColumnName - columnName
Retrieves a column's name
MemorySet - appendColumnString
Allocates a value to a Workflow/Glo
If-Else - ifElseA...
While- whileAct...
Runs an activity or a group of activities for a num...
Settings
Error
Help
Note
Cancel
Save

Counter* %tableColumnCount%

The first activity in the loop is **GetColumnName** which stores the label for the current column number from our while-loop.

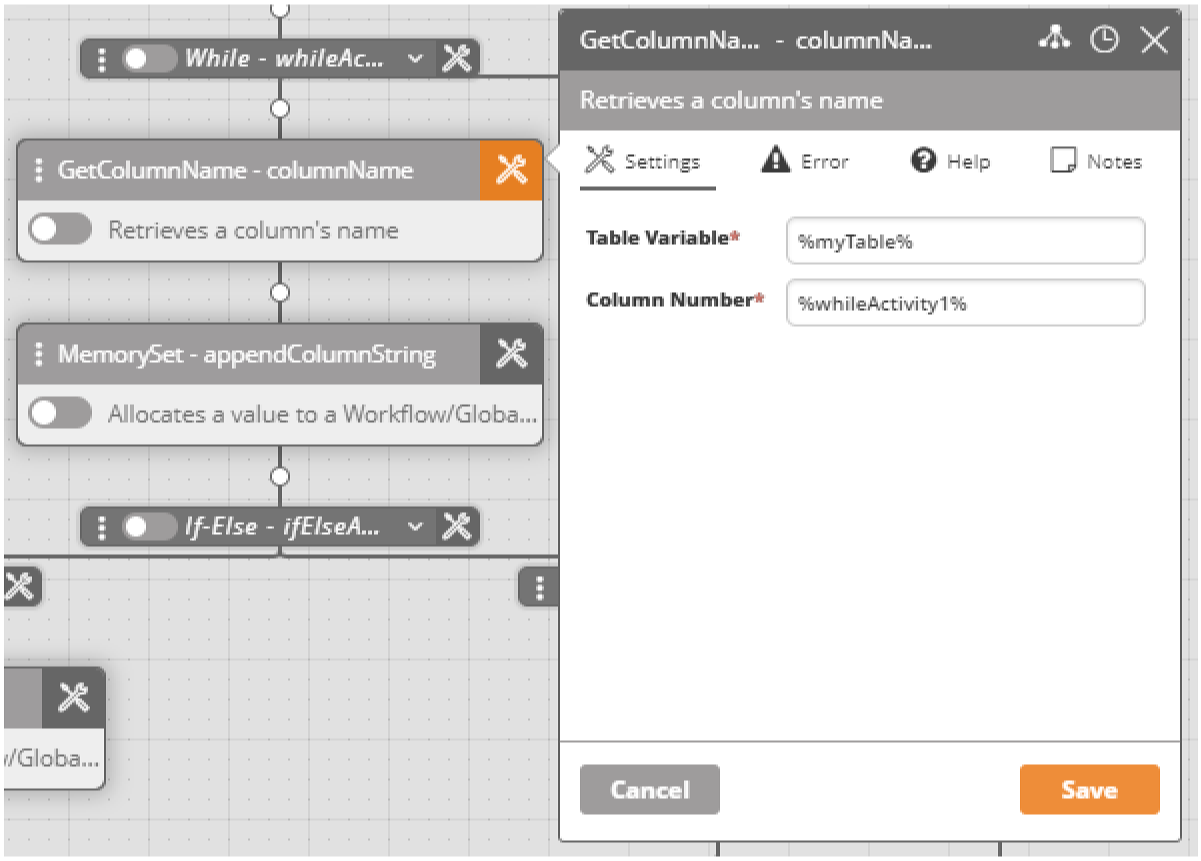
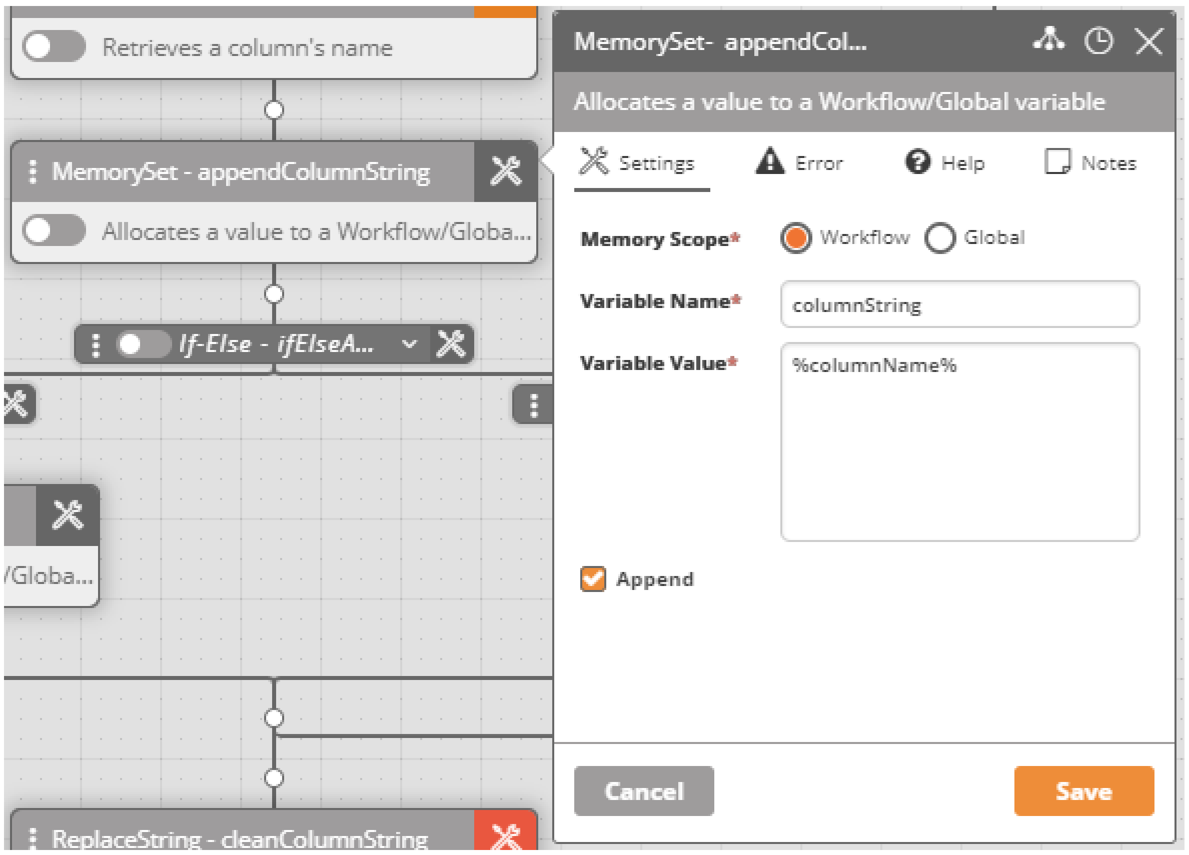

While - whileAc...
GetColumnName - columnName
Retrieves a column's name
MemorySet - appendColumnString
Allocates a value to a Workflow/Globa...
If-Else - ifElseA...
/Globa...
GetColumnNa... - columnNa...
Retrieves a column's name
Settings
Error
Help
Notes
Cancel
Save

Table Variable* %myTable%

Column Number* %whileActivity1%

Next, we use a **MemorySet** activity to store that column name into **columnString**. Be sure to check the **Append** checkbox so that each column name is stored as the loop makes each of its runs.


MemorySet- appendCol...

Allocates a value to a Workflow/Global variable

Settings | Error | Help | Notes

Memory Scope* [✓ Workflow] Global

Variable Name* columnString

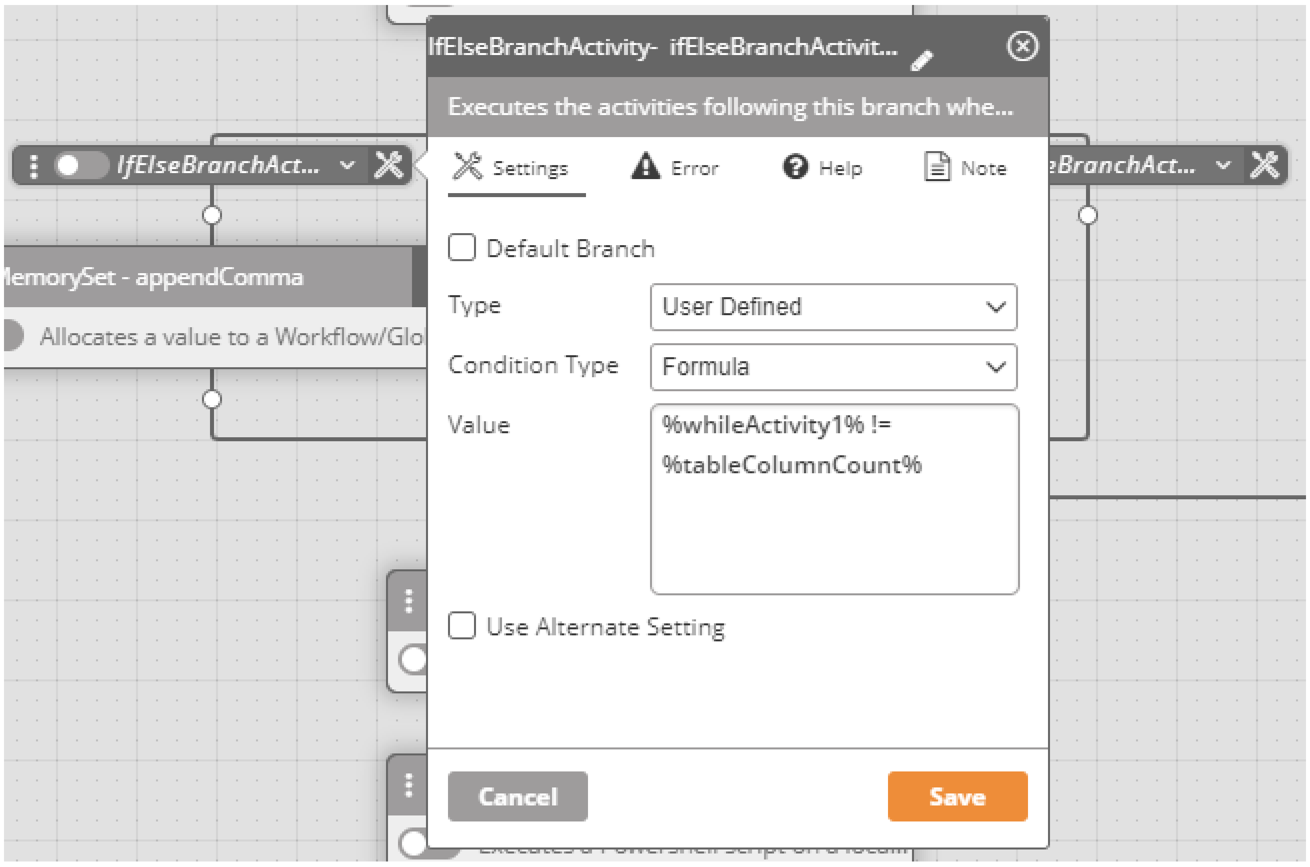
Variable Value* %columnName%

[✓] Append

Cancel | Save

Retrieves a column's name
MemorySet - appendColumnString
Allocates a value to a Workflow/Globa...
If-Else - ifElseA...
/Globa...
ReplaceString - cleanColumnString

We then create an if-else branch that checks to see if whether or not we're on the last iteration of our loop. If not, we proceed with an additional **MemorySet** activity to append a comma (,) to **columnString**.


IfElseBranchActivity- ifElseBranchActivit...

Executes the activities following this branch whe...

Settings | Error | Help | Note

[ ] Default Branch

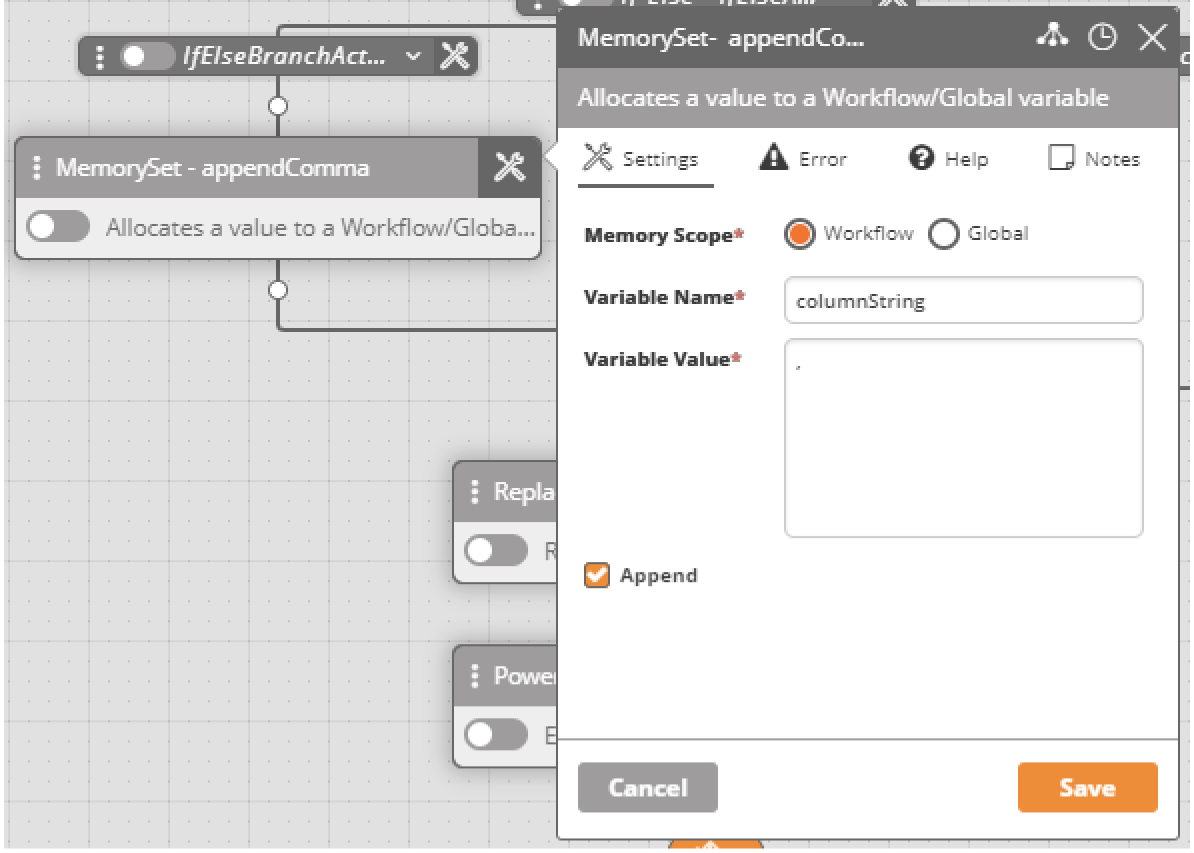
Type: User Defined

Condition Type: Formula

Value: %whileActivity1% != %tableColumnCount%

[ ] Use Alternate Setting

Cancel | Save



MemorySet- appendCo...

Allocates a value to a Workflow/Global variable

Memory Scope* [✓ Workflow] Global

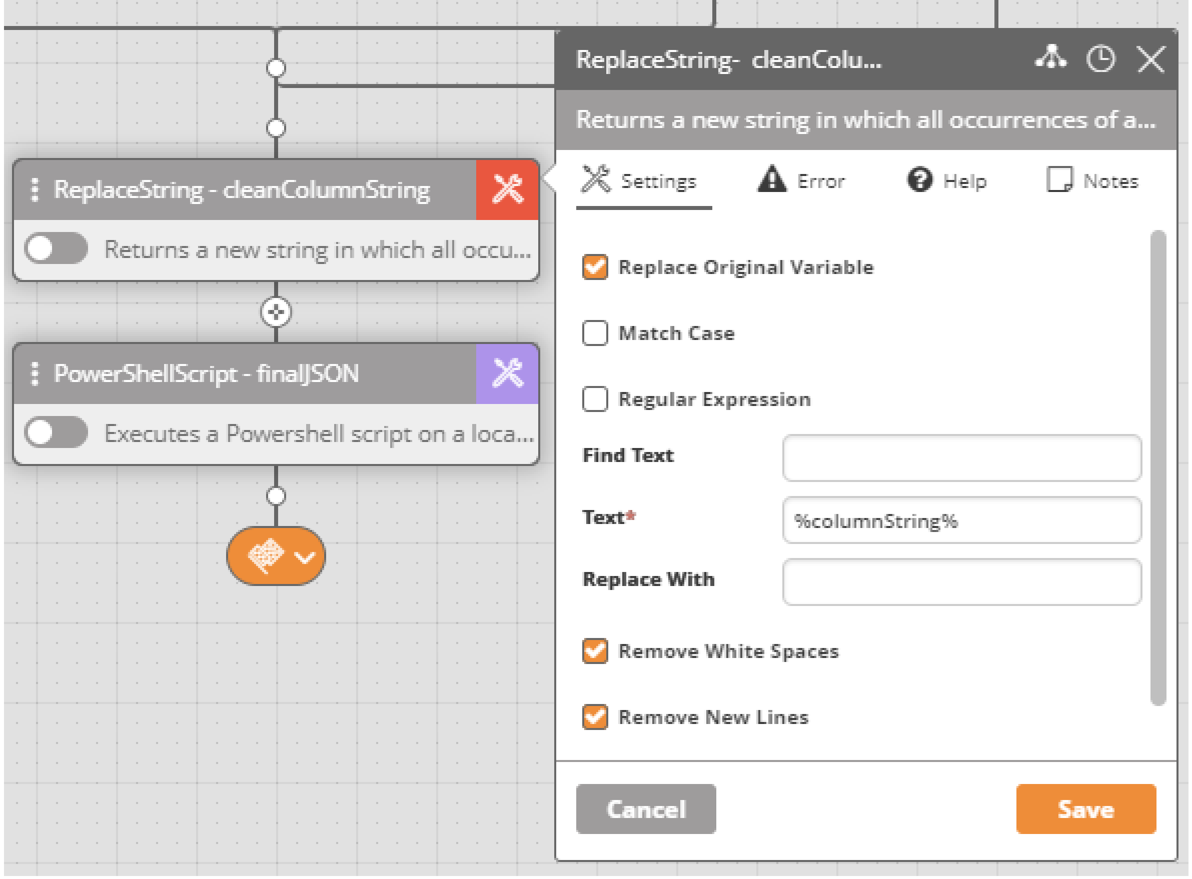
Variable Name* columnString

Variable Value* ,

[✓] Append

After the while-loop ends, a **ReplaceString** activity is used to remove all whitespace from **columnString**. Ensure that the following checkboxes are checked for this activity:

- Replace Original Variable
- Remove White Spaces
- Remove New Lines
- Remove Tabs


ReplaceString - cleanColumnString
Returns a new string in which all occu...
PowerShellScript - finalJSON
Executes a Powershell script on a loca...
ReplaceString- cleanColu...
Returns a new string in which all occurrences of a...
Settings
Error
Help
Notes
Cancel
Save

[✓] Replace Original Variable

[ ] Match Case

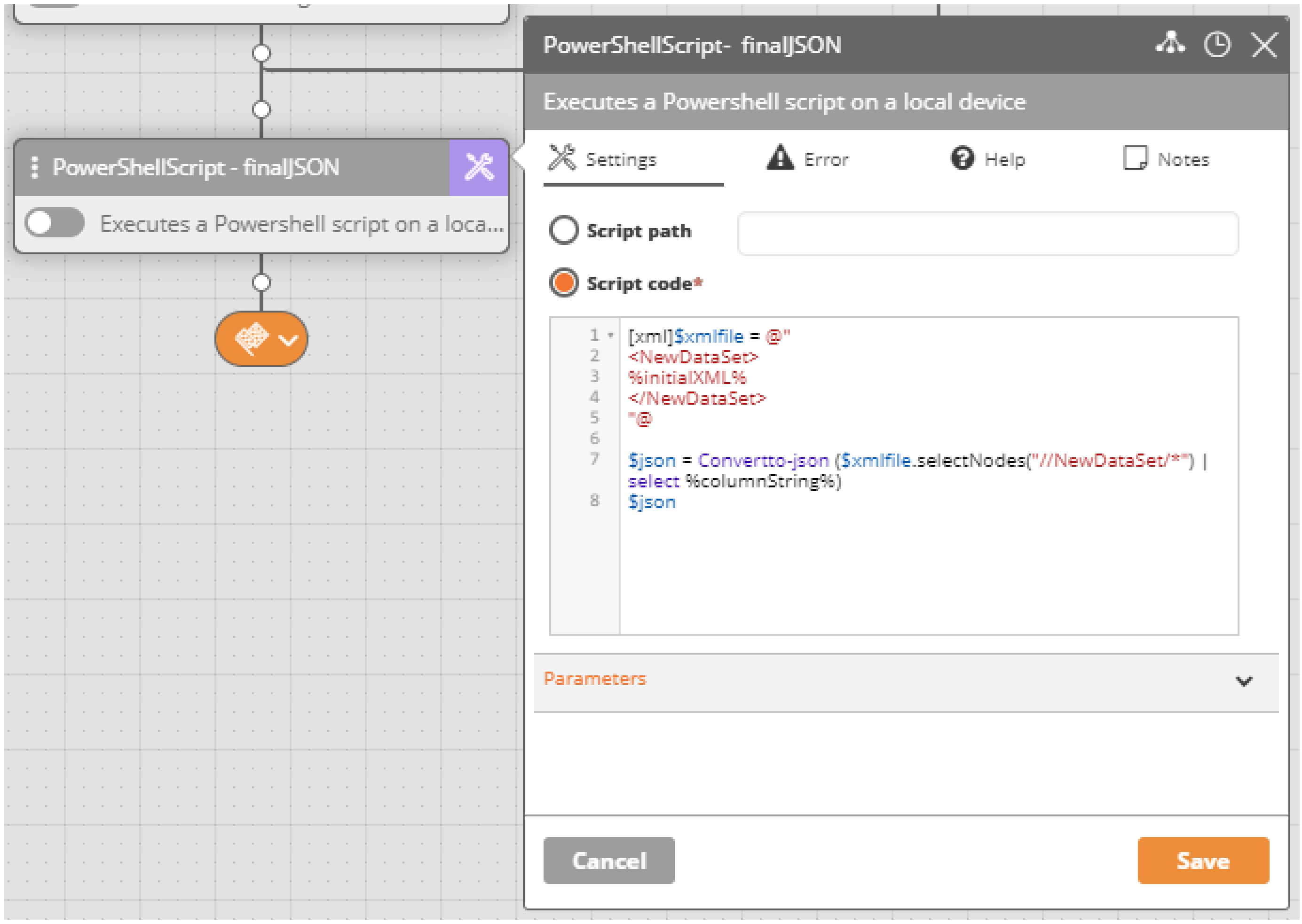
[ ] Regular Expression

Find Text	
Text*	%columnString%
Replace With	

[✓] Remove White Spaces

[✓] Remove New Lines

Our final step is a **PowerShellScript** activity that converts the XML we generated earlier in the workflow into JSON.


PowerShellScript - finalJSON
Executes a Powershell script on a loca...
PowerShellScript- finalJSON
Executes a Powershell script on a local device
Settings
Error
Help
Notes
Parameters
Cancel
Save

( ) Script path

(●) Script code*

```
[xml]$xmlfile = @"
<NewDataSet>
%initialXML%
</NewDataSet>
"@

$json = Convertto-json ($xmlfile.selectNodes("//NewDataSet/*") | select %columnString%)
$json
```

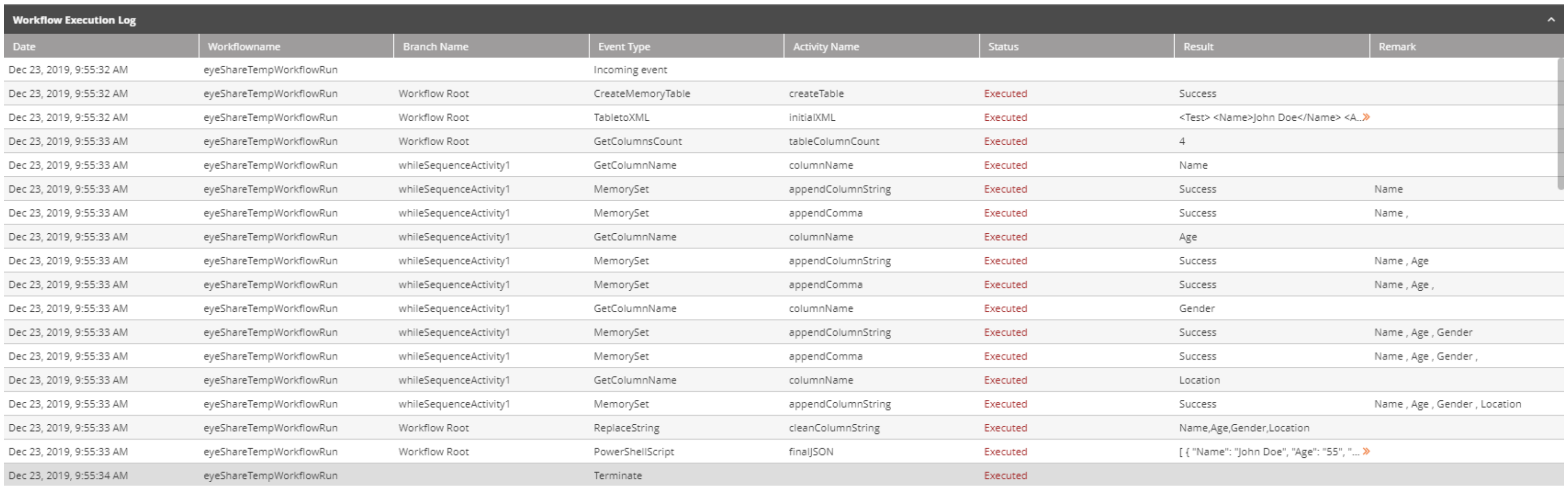
POWERSHELL CODE

```
[xml]$xmlfile = @"
<NewDataSet>
%initialXML%
</NewDataSet>
"@

$json = Convertto-json ($xmlfile.selectNodes("//NewDataSet/*") | select %columnString%)
$json
```

Workflow Execution

Below is a screenshot of the **Workflow Execution Log** from the sample workflow used in this article.

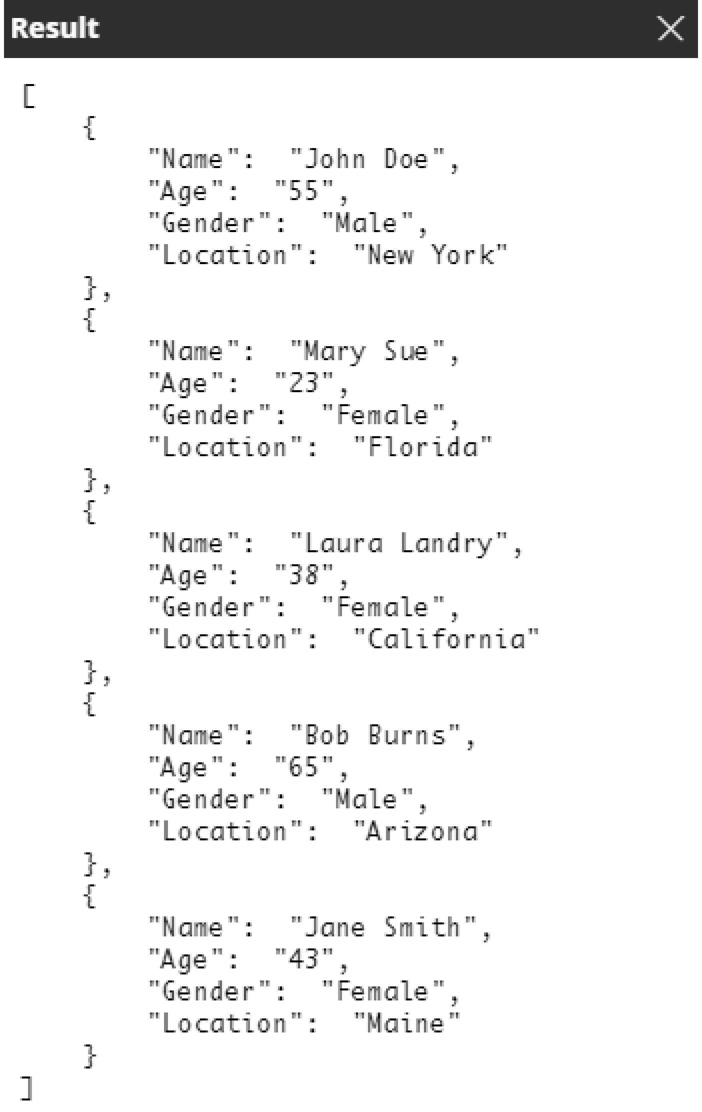


Workflow Execution Log

Date	Workflowname	Branch Name	Event Type	Activity Name	Status	Result	Remark
Dec 23, 2019, 9:55:32 AM	eyeShareTempWorkflowRun		Incoming event				
Dec 23, 2019, 9:55:32 AM	eyeShareTempWorkflowRun	Workflow Root	CreateMemoryTable	createTable	Executed	Success	
Dec 23, 2019, 9:55:32 AM	eyeShareTempWorkflowRun	Workflow Root	TabletoXML	initialXML	Executed	<Test> <Name>John Doe</Name> <A..»	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	Workflow Root	GetColumnsCount	tableColumnCount	Executed	4	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	GetColumnName	columnName	Executed	Name	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendColumnString	Executed	Success	Name
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendComma	Executed	Success	Name ,
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	GetColumnName	columnName	Executed	Age	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendColumnString	Executed	Success	Name , Age
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendComma	Executed	Success	Name , Age ,
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	GetColumnName	columnName	Executed	Gender	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendColumnString	Executed	Success	Name , Age , Gender
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendComma	Executed	Success	Name , Age , Gender ,
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	GetColumnName	columnName	Executed	Location	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	whileSequenceActivity1	MemorySet	appendColumnString	Executed	Success	Name , Age , Gender , Location
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	Workflow Root	ReplaceString	cleanColumnString	Executed	Name,Age,Gender,Location	
Dec 23, 2019, 9:55:33 AM	eyeShareTempWorkflowRun	Workflow Root	PowerShellScript	finalJSON	Executed	[{ "Name": "John Doe", "Age": "55", "... »	
Dec 23, 2019, 9:55:34 AM	eyeShareTempWorkflowRun		Terminate		Executed		

Click image to view full-sized version.

The final result from the **PowerShellScript** activity is a JSON-formatted version of the table created at the beginning of the workflow.



```
[
    {
        "Name":  "John Doe",
        "Age":  "55",
        "Gender":  "Male",
        "Location":  "New York"
    },
    {
        "Name":  "Mary Sue",
        "Age":  "23",
        "Gender":  "Female",
        "Location":  "Florida"
    },
    {
        "Name":  "Laura Landry",
        "Age":  "38",
        "Gender":  "Female",
        "Location":  "California"
    },
    {
        "Name":  "Bob Burns",
        "Age":  "65",
        "Gender":  "Male",
        "Location":  "Arizona"
    },
    {
        "Name":  "Jane Smith",
        "Age":  "43",
        "Gender":  "Female",
        "Location":  "Maine"
    }
]
```