**My Salinger Year Video Commentary**

My Salinger Year (also known as My New York Year) is a 2020 drama film written and directed by Philippe Falardeau, based upon the memoir, of the same name, by Joanna Rakoff. It stars Margaret Qualley, Sigourney Weaver, Douglas Booth, Seána Kerslake, Colm Feore and Brían F. O'Byrne.

After attending a recent cinema screening of "My Salinger Year", in 2021, it occurred to me that Excel could help with creating typos just like JD Salinger wanted. This demo app uses a new programming language called Office Scripts to create new three letter words. An online dictionary was added to the solution so that the definitions, if they, exist appear alongside the newly generated words. This demo shows how Office Scripts programmers can add spell checker and dictionary definition functionality to Excel. Synonyms, too, can be obtained using the same methods.

JD Salinger is famous for writing the following literary works:

• The Catcher in the Rye (1951)

• Nine Stories (1953)

• Franny and Zooey (1961)

• Raise High the Roof Beam, Carpenters and Seymour: An Introduction (1963)

• Odds and Ends

In order to operate the Salinger Word Generator we enter a 1 into cell G1 to start the run. It takes around 100 seconds for each run to complete the word generation and the dictionary definition retrieval.

Let’s now see the demo in action. For reasons of brevity I have removed the portions of the video were nothing happens showing only the word generation and the definition input.

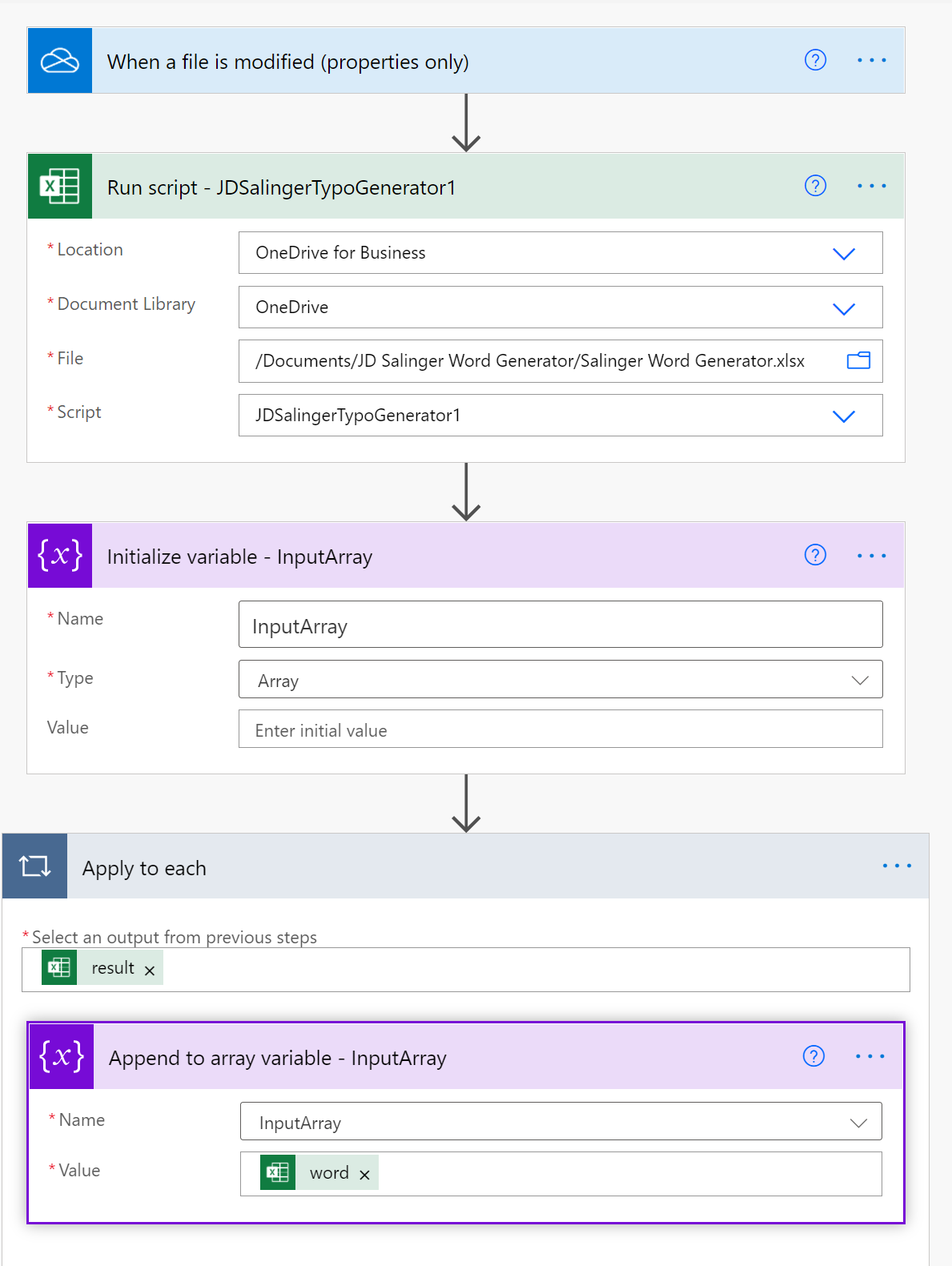
**So, now, moving on to the technical part of this video where I explain how the Salinger words were** generated and how the definitions were obtained.

First there is the Power Automate Flow as is shown, here.

As can be seen, by the presence of the two green connectors, the Flow uses two Office Scripts. One to generate the Salinger Words and one to input the generated definitions back into the Excel Workbook.

The flow is triggered by a change in the Workbook. That is, when a 1 is entered into cell G1. The Office Script, then, runs. The Apply to each loop uses the array variable *InputArray* to hold the generated twenty 3-letter words.

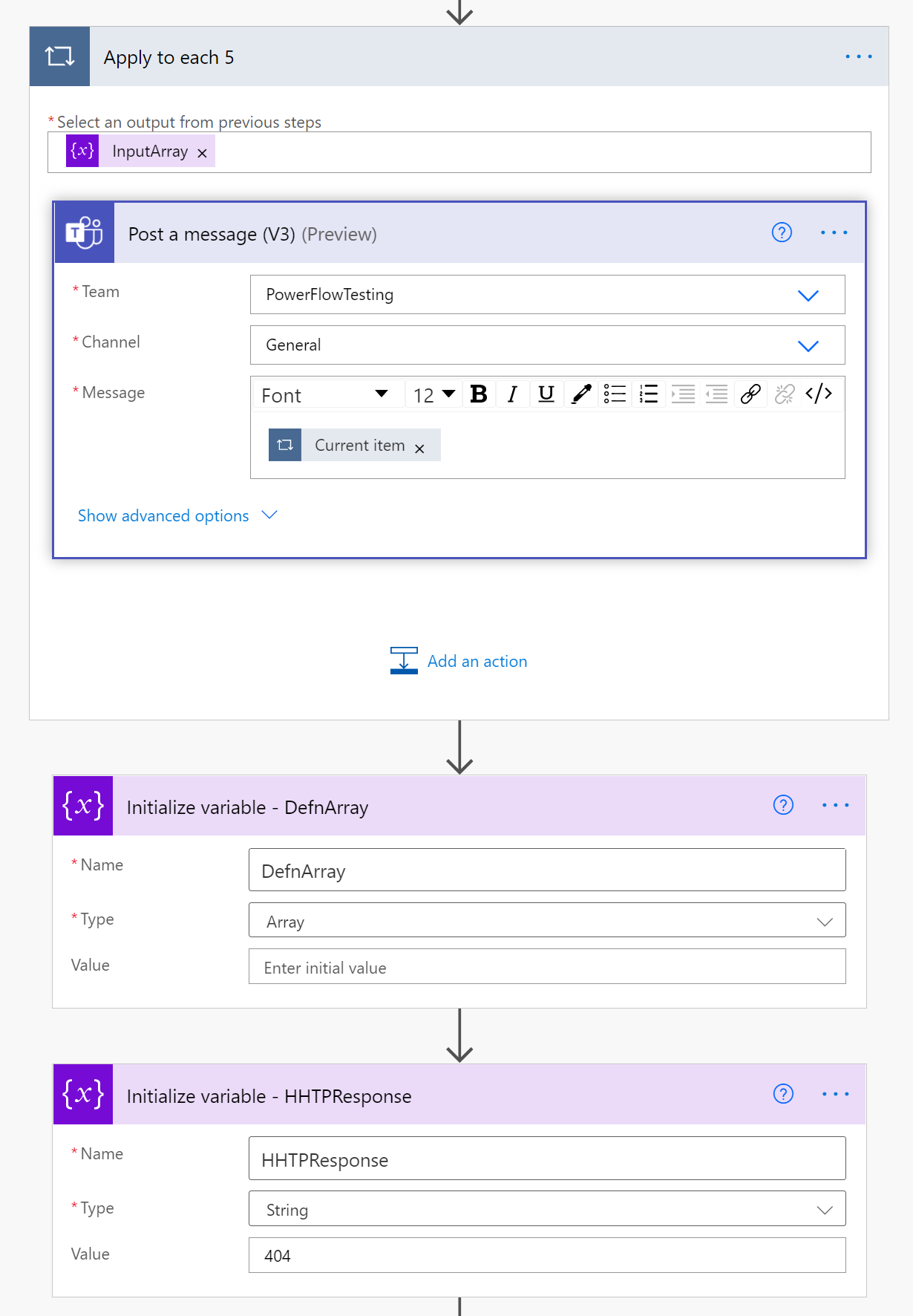
**Fig 4. The InputArray stores the generated twenty, 3-letter, Salinger Words.**



The twenty Salinger words are output to a Teams channel, as part of the debug checking process. Two variables, *DefnArry* and *HTTPResponse* are created. The DefnArray variable holds the list of retrieved definitions for each of the 20 Salinger words. The HHTPReponse variable contains the HTTP

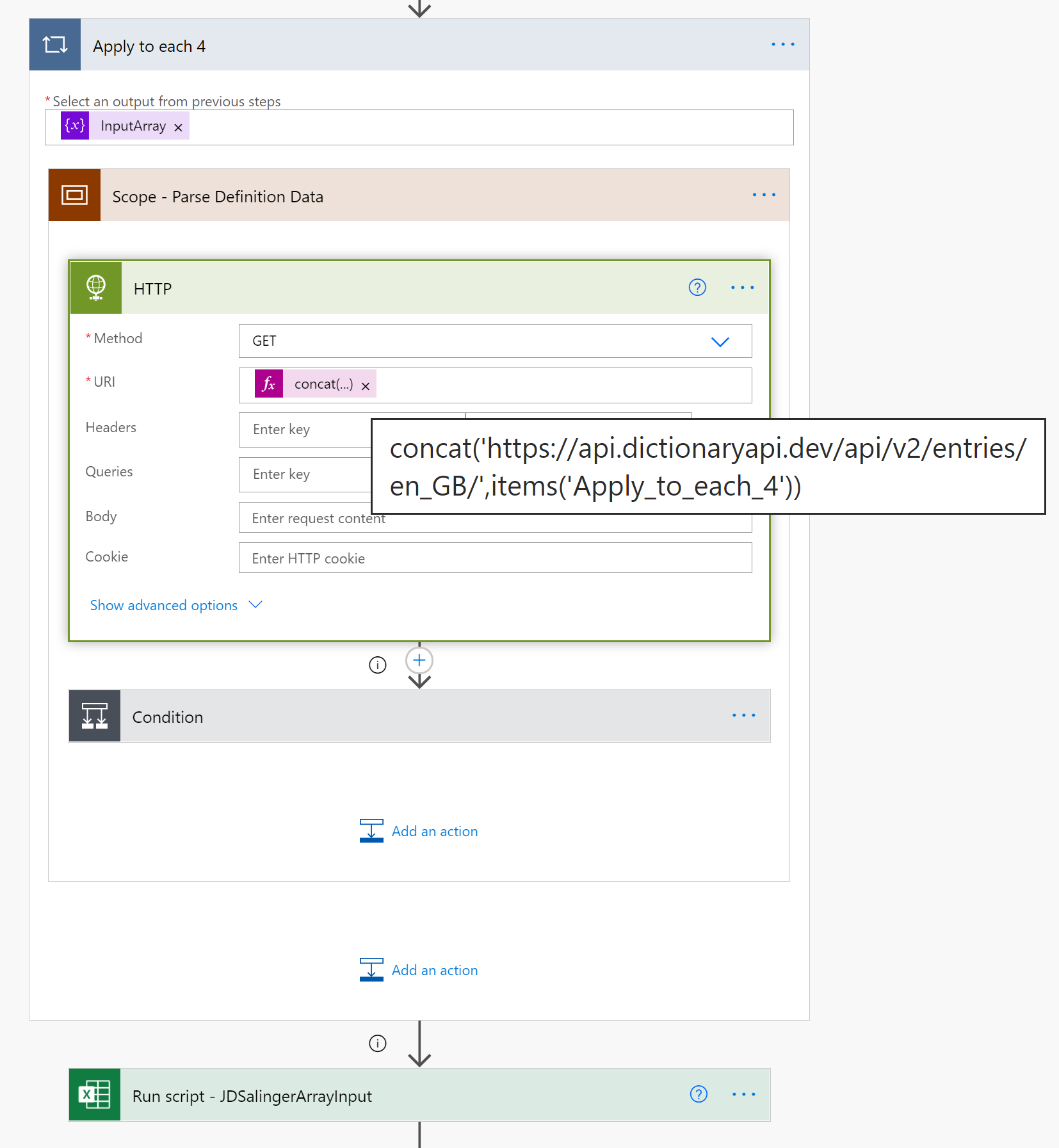
code 404 which “indicates that the browser was able to communicate with a given server, but the server could not find what was requested”. The HTTP connector connects with the free Dictionary REST API from Dicionaryapi.dev : [Free Dictionary API](https://dictionaryapi.dev/).

**Fig 5. The Salinger Words are posted to Teams**



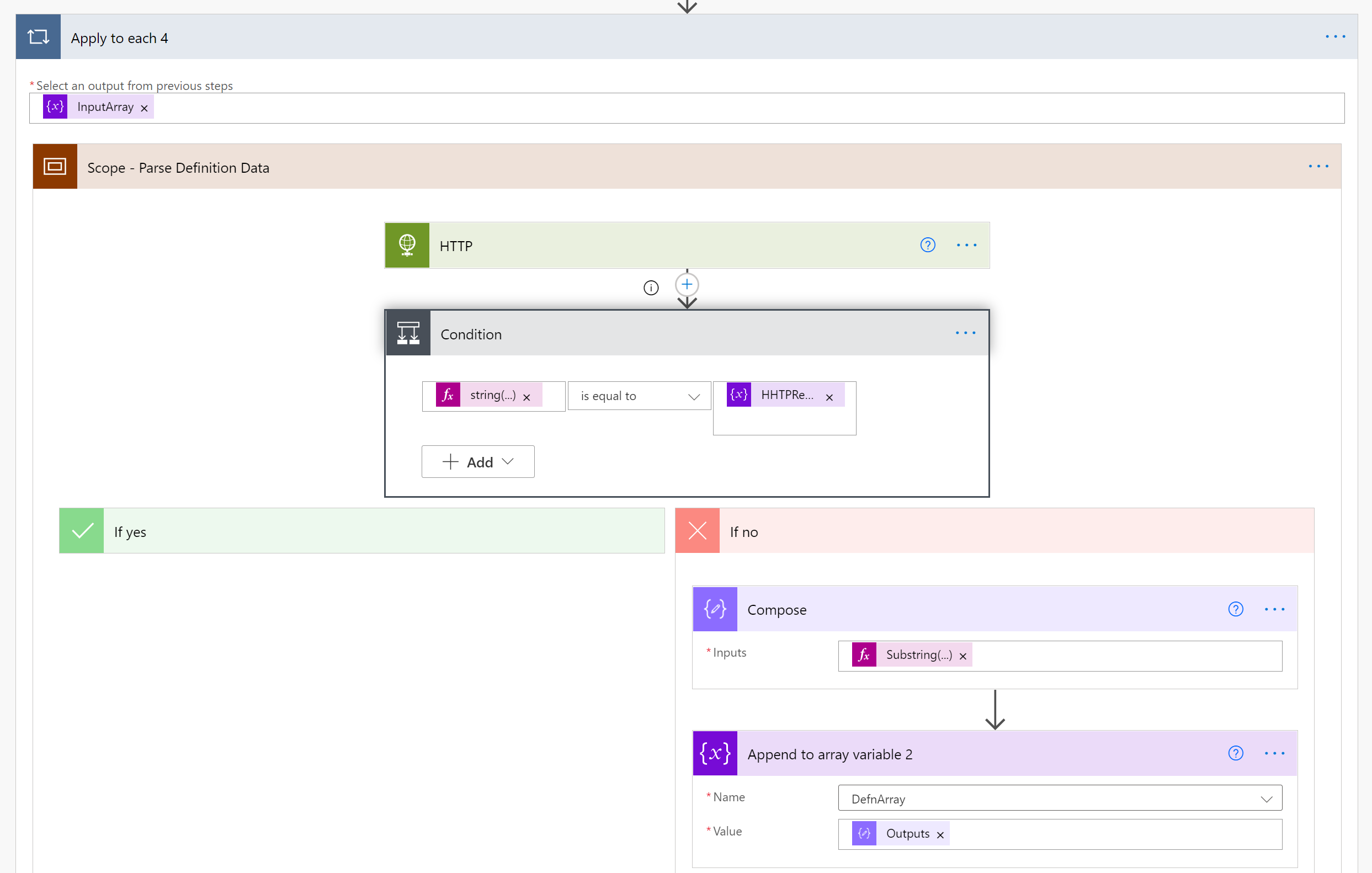
In the “Apply to each 4” loop, the definitions for each of the Salinger words are retrieved using the HTTP connector and added to the DefnArray.

**Fig 6. The “Apply to each 4” loop.**



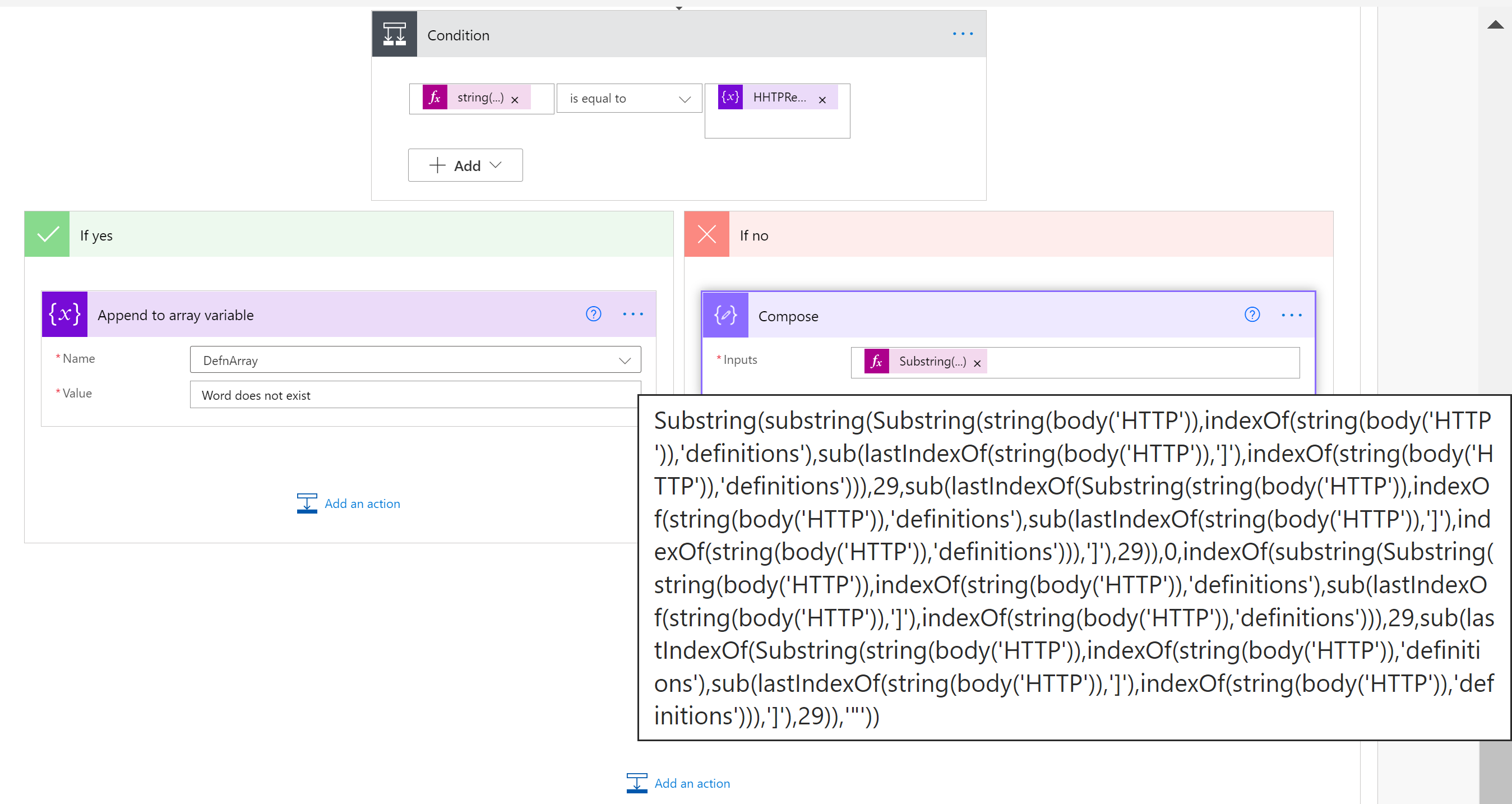
In the condition step, the formula “string(outputs('HTTP')['statusCode'])” is compared to the value 404. If true then the “No Word Exists” text is added as an array item to the DefnArray variable. If the condition is false then a definition has been found and it is extracted from the HTTP Body response value.

**Fig 7. Condition evaluates to TRUE or FALSE adding the HHTP connector response to the DefnArray**



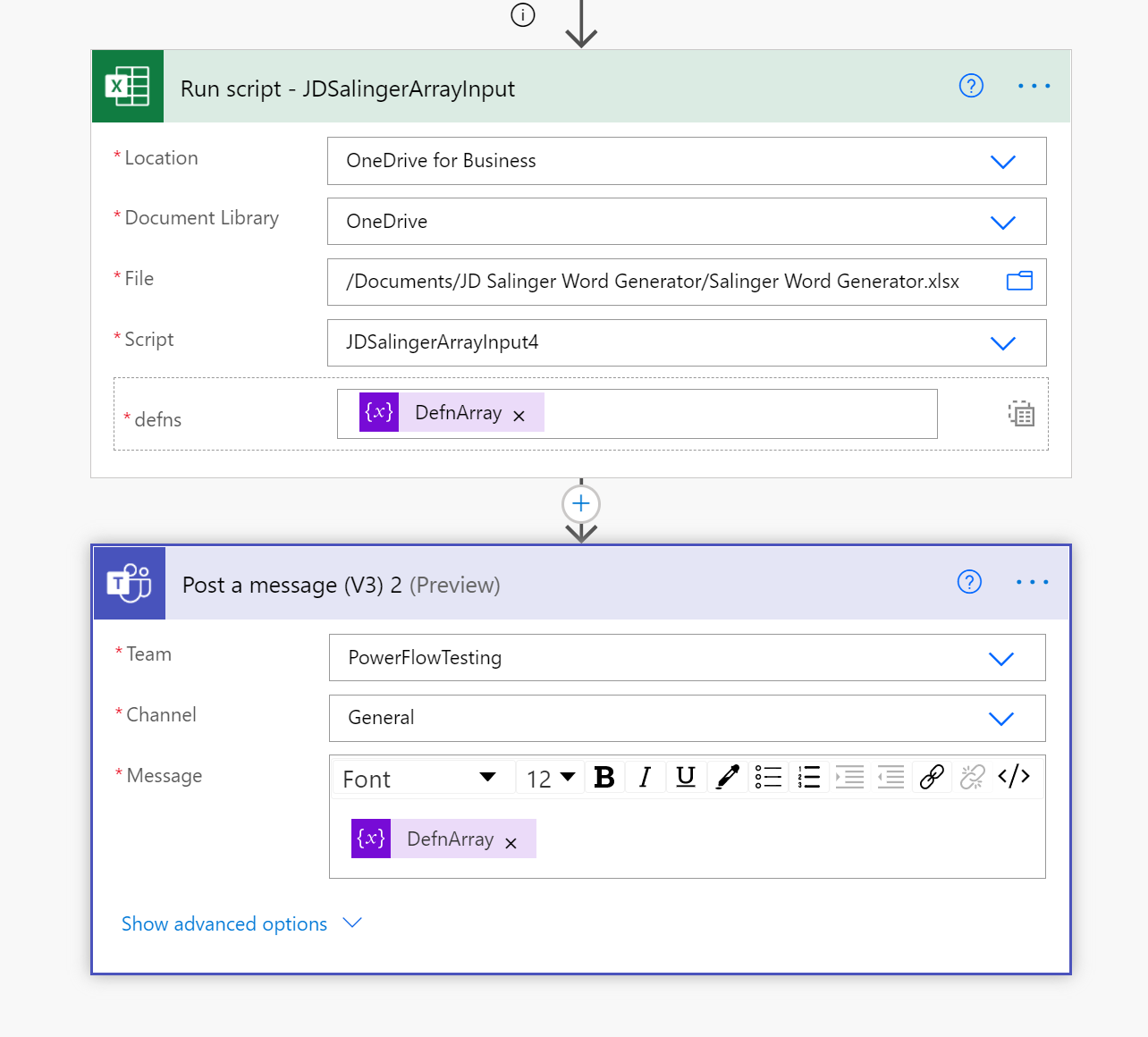
Even though the HTTP Response was in JSON format and Parse JSON connectors do operate on this response, they could not be used in this case for practical reasons. This was because the JSON syntax changes, dynamically, depending on the number of *partOfSpeech* items in the JSON response. That is, the JSON changes when a definition contains of adverb, verb, noun, adjective and so on. Each word has a variable amount of these and the JSON for the two examples, as provided in Appendix 1 and 2, in this document, ranges from 1 page to 7 pages. So, there is a huge difference in the size and number of layers for each JSON response for each word definition retrieved. Creating a set of PARSE Json operations as well as the associated loops and variables to manage this would be onerous to create. An alternative much easier method was possible. As can be seen in Fig 8, the string extraction formula was used. It extracts the first definition it encounters ignoring all subsequent definitions with that same response body. Each response can contain up to 7 or, even, more definitions. Only the first one was extracted.

**Fig 8. The definition string extraction formula**



The final Office Script inputs the definitions, found, into the Excel workbook.

**Fig 9.** **The final Office Script input the definitions, found, into Excel.**



Moving on, now, to the Office Scripts,

The first, **JDSalingerTypoGenerator,** generates a string of 3 characters long using the Math.Random TypeScript function. A loop function repeats this twenty times with each word being added to an array. The array forms an input back into the Flow.

Text

Description automatically generated

In the second script, **JDSalingerArrayInput4**, one of its input parameters, is an array previously created by the Flow to hold the dictionary retrieved definitions. The array is converted into a string and the commas, contained, within some of the individual definitions, are removed. The commas left behind are the separator characters. The plit function is used to split the string, by that comma character, into an array. The array is iterated and output into the workbook column C.

Text

Description automatically generated