**Office Scripts Scrabble ® Solver**

**Contents**

* Introduction
* Operating the Office Scripts Scrabble Solver
* The Disclaimer
* The Power Automate Flow
* Further Notes
* GitHub Repo location
* YouTube Video Location
* The Office Scripts
* ScrabbleWordGenerator7
* ScrabbleSolverArrayInput
* References
* All Scrabble-accepted Two letter words
* Scrabble Words With No Vowels and No "Y"
* Scrabble Words With No Vowels That Use "Y"

**Introduction**

The Office Scripts Scrabble ® Solver uses the following Microsoft tools : Office Scripts, Power Automate, OneDrive for Business and Excel Online. Just like in Scrabble the player has a set of tiles on their rack which they must find a word from. The number of combinations possible from 7 tiles (characters) is 5,568. For 6 characters it is ***1,584***. Given medium strong players already know all the 2 letter words, this script, does not supply all the possible two letter words. The script ends at the three letter word level.

**Operating the Office Scripts Scrabble Solver**

In order to operate the Office Scripts Scrabble Solver enter a 1 in cell D5 to start a run. For 4 character length strings it takes around 96 seconds for each run to complete the word generation and the dictionary definition retrieval. The vast majority of that time is for the round tripping to the dictionary API and back.

In Fig 1. we can see the result of the input word “micro” into a run. Only those words that have a definition are visible. All others have been filtered using the Excel filter functionality.

**Fig 1. The Office Scripts Scrabble Solver after a run with the word “microso”**

Graphical user interface, application, table

Description automatically generated

The scripts used in this demo are supplied separately as .ts files, in the associated GitHiub repo, and are included at the end of this Word document.

**The Disclaimer**

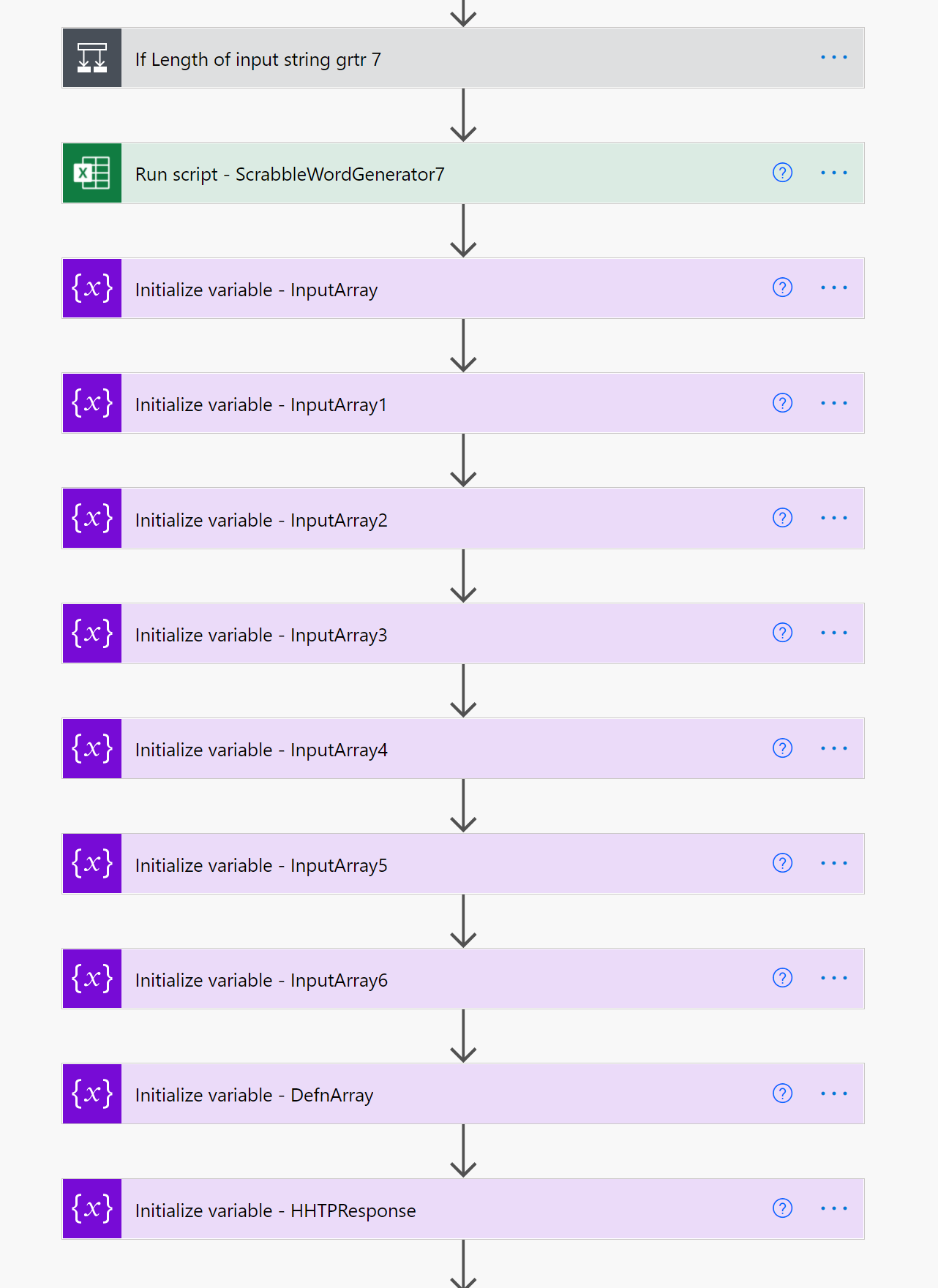
SCRABBLE® is a registered trademark. All intellectual property rights in and to the game are owned in the U.S.A and Canada by Hasbro Inc., and throughout the rest of the world by J.W. Spear & Sons Limited of Maidenhead, Berkshire, England, a subsidiary of Mattel Inc.

This solver is not affiliated with SCRABBLE®, Mattel Inc, Hasbro Inc.

**The Power Automate Flow**

In reference to my previous demo solution, the JD Salinger Word Generator, the flow is different only in that the two Office Scripts are different and the final Filter step has been removed. The filtering, now, takes place, in Excel, operated by the script rather than in the Flow. This was changed because all of the generated words needed to be output back into the Excel workbook rather than removed prior to input.

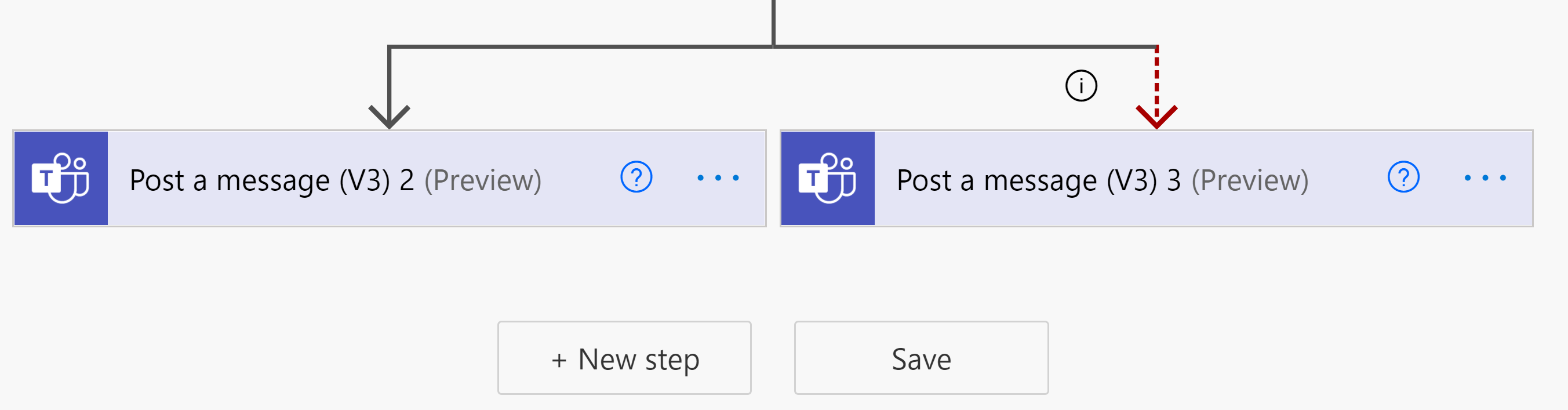
**Fig 2. The full Scrabble Solver Power Automate Flow Top half**



**Fig 3. The full Scrabble Solver Power Automate Flow Bottom half**

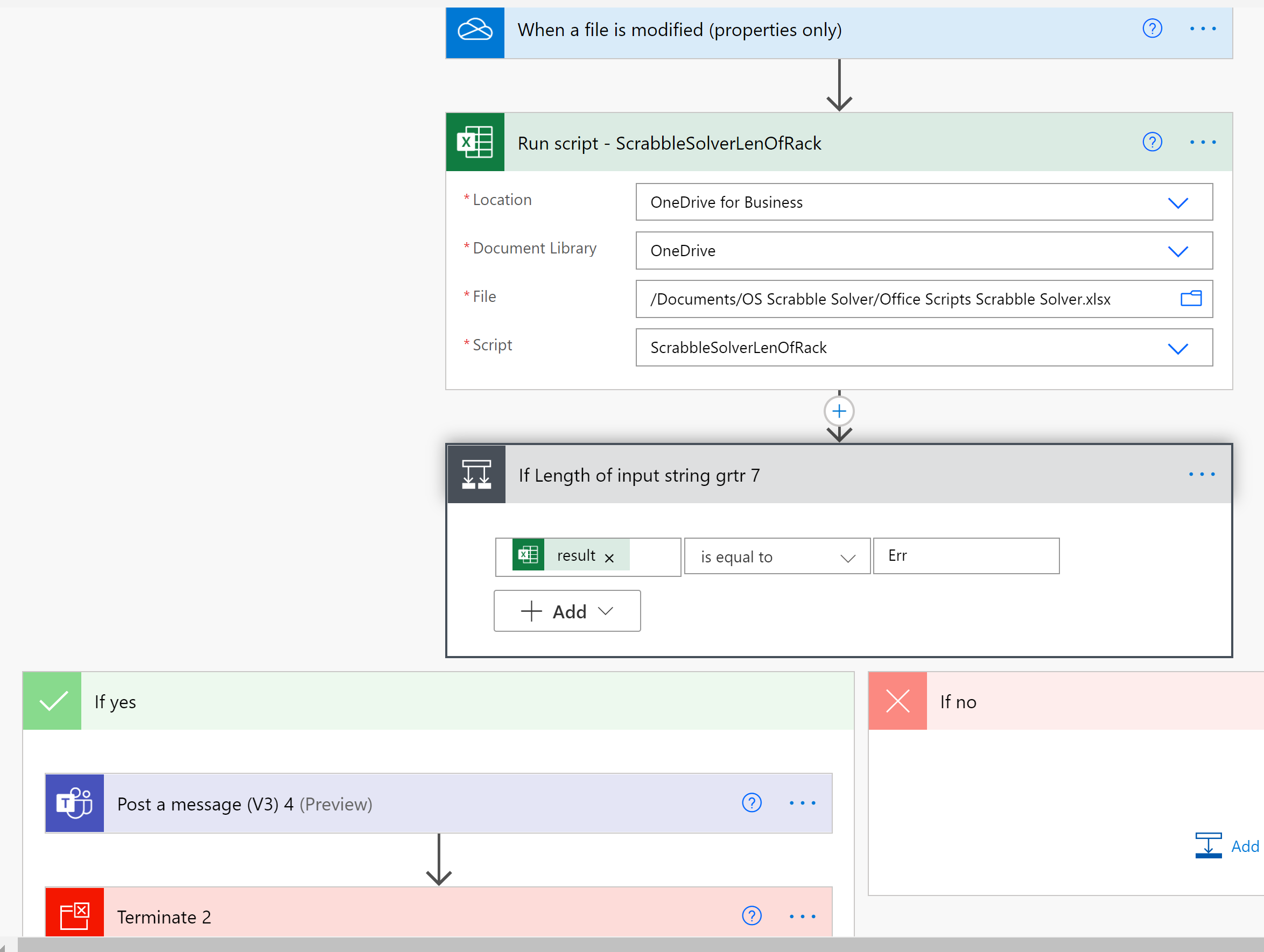


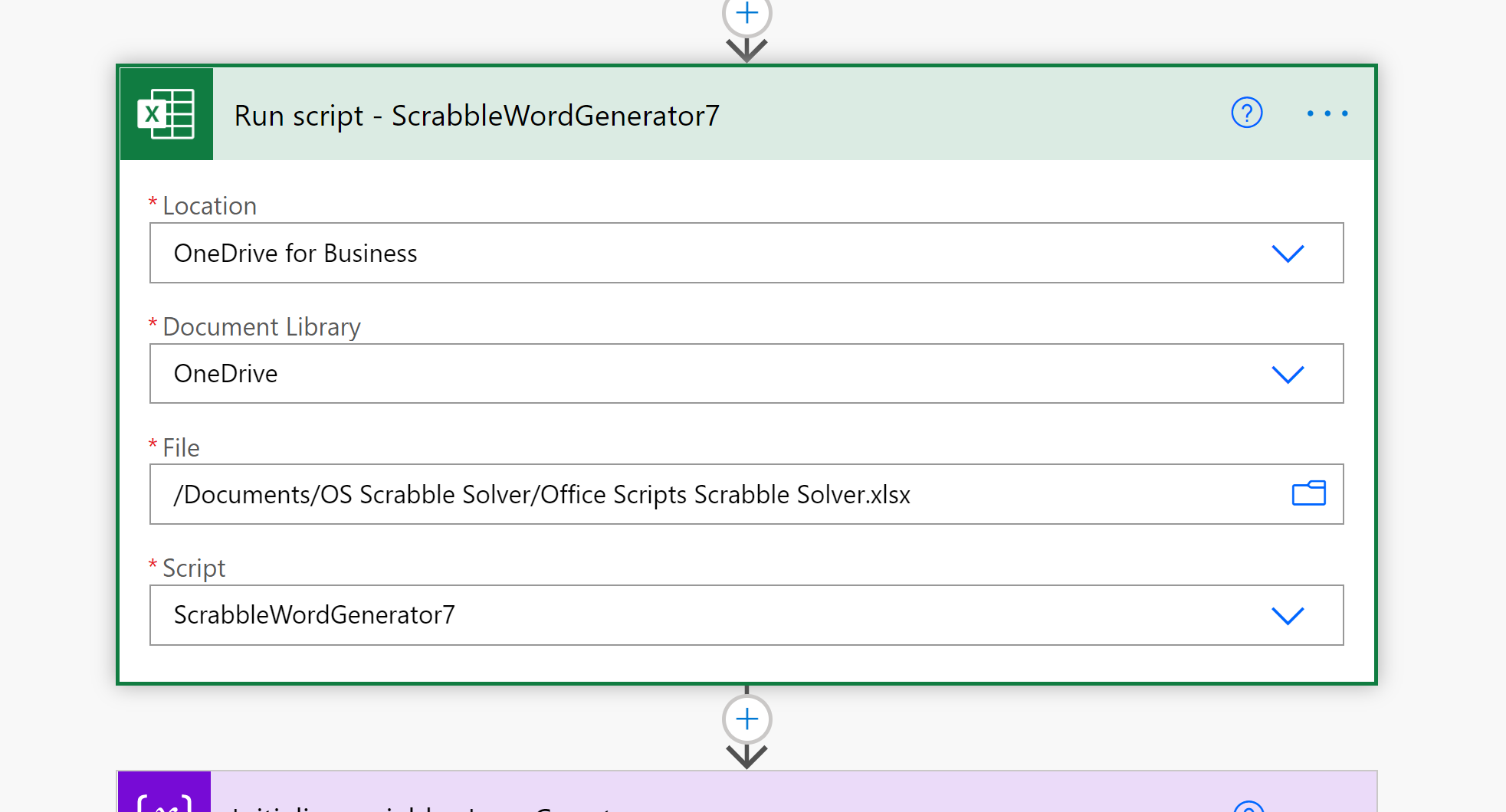
**Fig 4. The end of the Flow**



Stepping through each Flow action, in turn, we can see, In Fig 5, that the flow is triggered when a file, in this case the Excel Scrabble Solver workbook, changes. That is, when a 1 is entered into cell D5. The Office Script, then, runs. Immediately a script runs testing the length of the inputted string (the rack). If it is greater than 7 then the entire Flow terminates with a message posted to Teams.

**Fig 5. Flow trigger and Length of rack test**

Next, the ScrabbleWordGenerator7 script runs. This generates the anagrams and sub words for the provided input string.

**Fig 6. The acronyms and sub words are generated by a script**

As shown in Fig 2 the variables are declared as follows : *InputArray, InputArray1, InputArray2, InputArray3, InputArray4, InputArray5, InputArray6, DefnArray* and *HTTPResponse*. InputArray is populated with the result of the previous Office Script. InputArray1 is populated with a subset of values from the InputArray with the following formula. The other variables are, also, displayed here.

take(variables('InputArray'),1999)

InputArray2 variable is populated with the following formula:

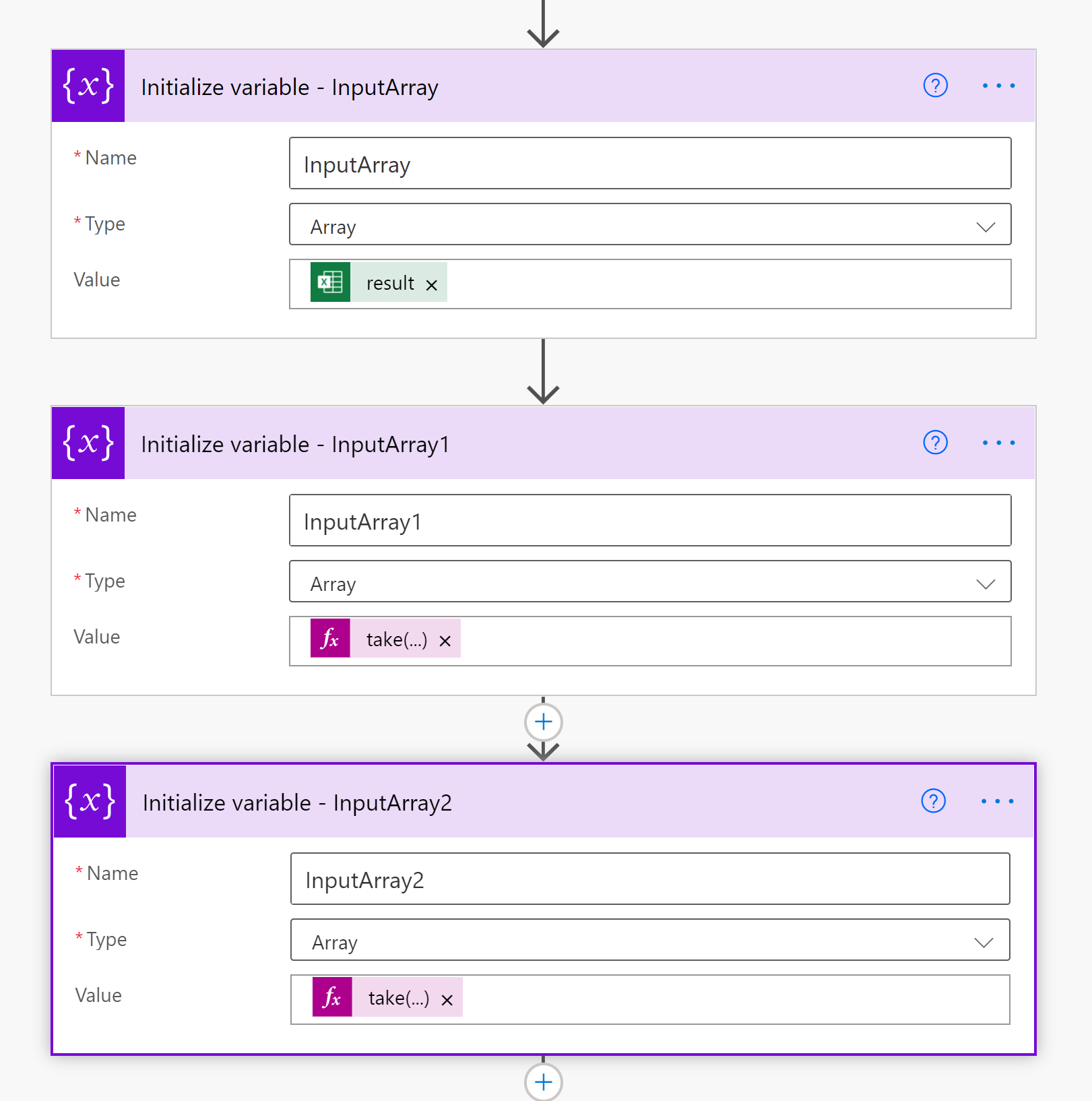
*take(skip(variables('InputArray'), 1999), 1999)*

InputArray3 variable is populated with the following formula: *take(skip(variables('InputArray'), 3998), 1999)*  
InputArray4 is populated with the following formula : *take(skip(variables('InputArray'), 5997), 1999)*

InputArray5 is populated with the following formula : *take(skip(variables('InputArray'), 7996), 1999)*

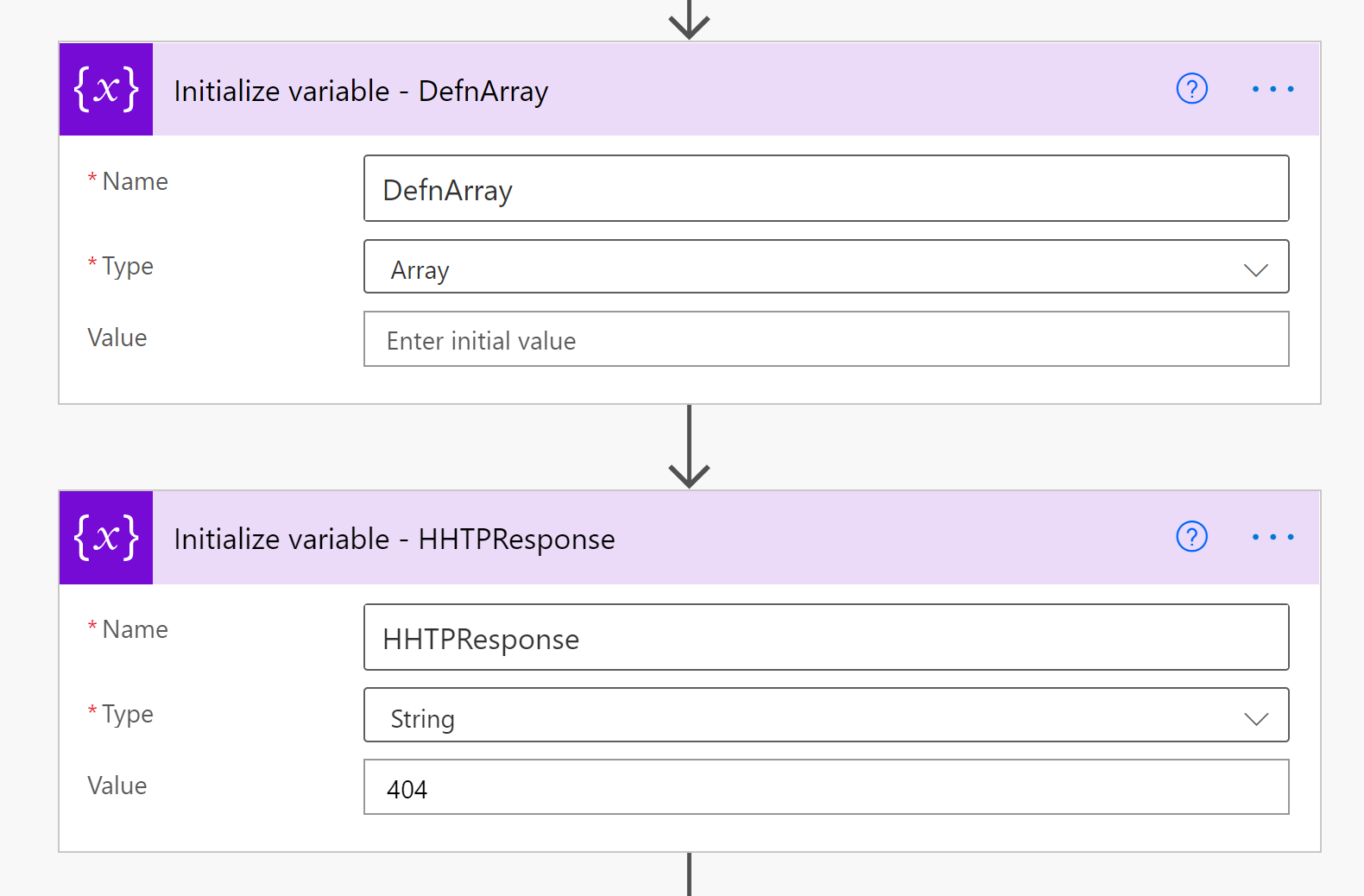
InputArray6 is populated with the following formula : *take(skip(variables('InputArray'), 9995), 1999)*

**Fig 7. Initializing InputArray, InputArray1 and InputArray2**



DefnArray and HTTPResponse variables are, also, initialized. The DefnArray variable holds the list of retrieved definitions for each of the generated 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 a Dictionary REST API from Dicionaryapi.dev : [Free Dictionary API](https://dictionaryapi.dev/).

**Fig 8. Initializing *DefnArray* and *HTTPResponse* variables**



In Fig 9, below, the first Apply to each loop iterates over the first 1,999 words to find their definitions using the HTTP Connector. Those definitions are added to the DefnArray. In the case of a 7 letter length input there will be a total of 5,568 words to find the definitions for. This, first, Apply to Each loop deals with the first 1,999 of that list. For the case of a 7 letter long input string, the total duration of the Flow run including the running of the Office Scripts was 1 hour 12 minutes and 56 seconds. This can vary considerably based on the time of day and day of week.

Where no definition is available the response, from the HTTP connector is 404. This value is contained within the status code response. In the condition step the formula “string(outputs('HTTP')['statusCode'])” is compared to the value 404 as represented by the HTTPResponse variable. If true then the “Word does not exist” text is added as item (value) to the DefnArray. If the condition is false then a definition has been found and it is extracted from the HTTP Body response value using the formula below.

The definition string extraction formula:

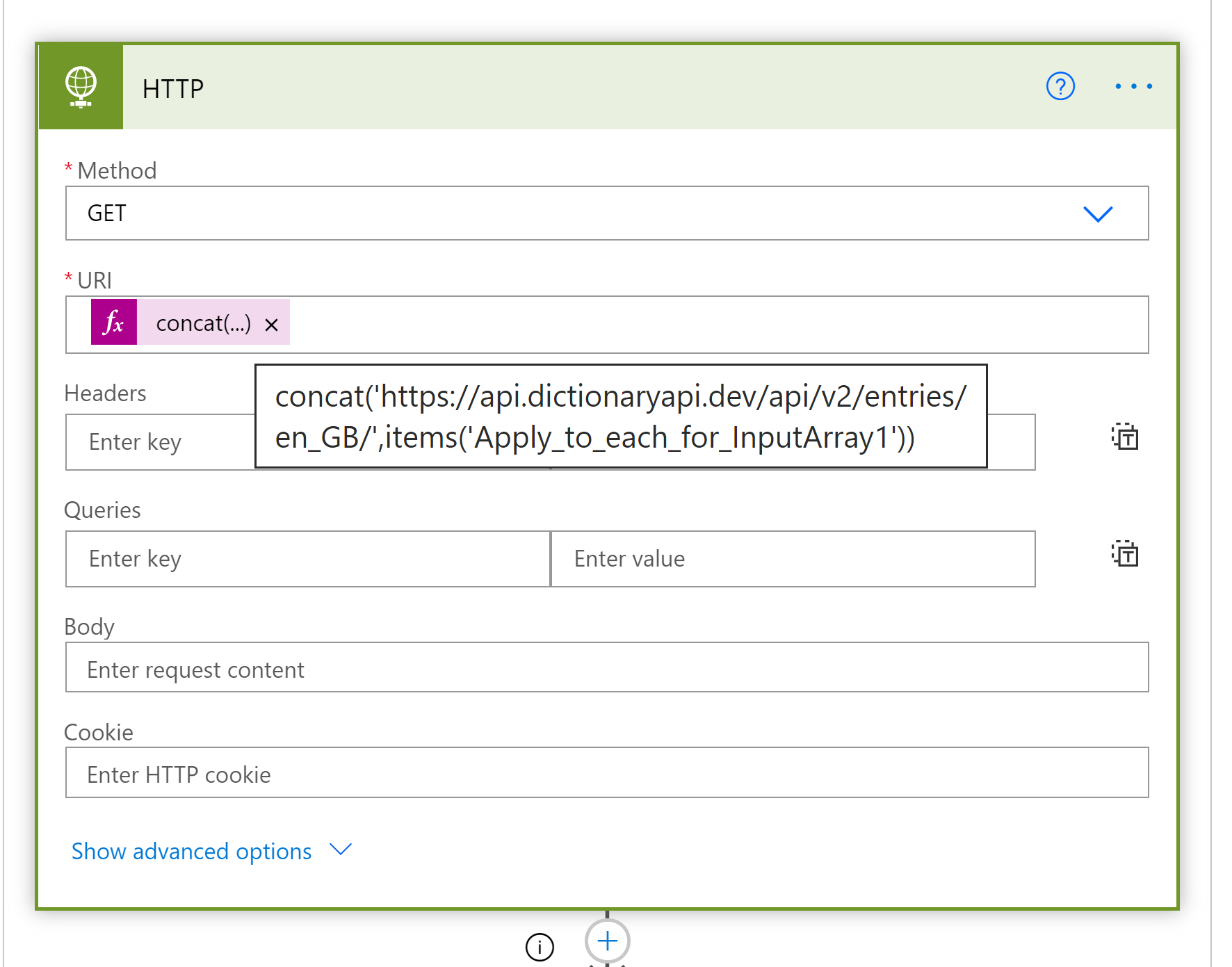
Substring(substring(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),29,sub(lastIndexOf(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),']'),29)),0,indexOf(substring(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),29,sub(lastIndexOf(Substring(string(body('HTTP')),indexOf(string(body('HTTP')),'definitions'),sub(lastIndexOf(string(body('HTTP')),']'),indexOf(string(body('HTTP')),'definitions'))),']'),29)),'"'))

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, because the JSON syntax changes, dynamically, depending on the number of *partOfSpeech* items in the JSON response. That is, the JSON changes layers and layer depth 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 an overly onerous and unnecessary task to create. An alternative, much easier, method was possible using a formula. The formula extracts the first definition it encounters ignoring all subsequent definitions for that same response body. Each response can contain up to 7 or, even, more definitions. That is, only the first one was extracted.

**Fig 9. Apply to each for InputArray1**Graphical user interface, application, Teams

Description automatically generated

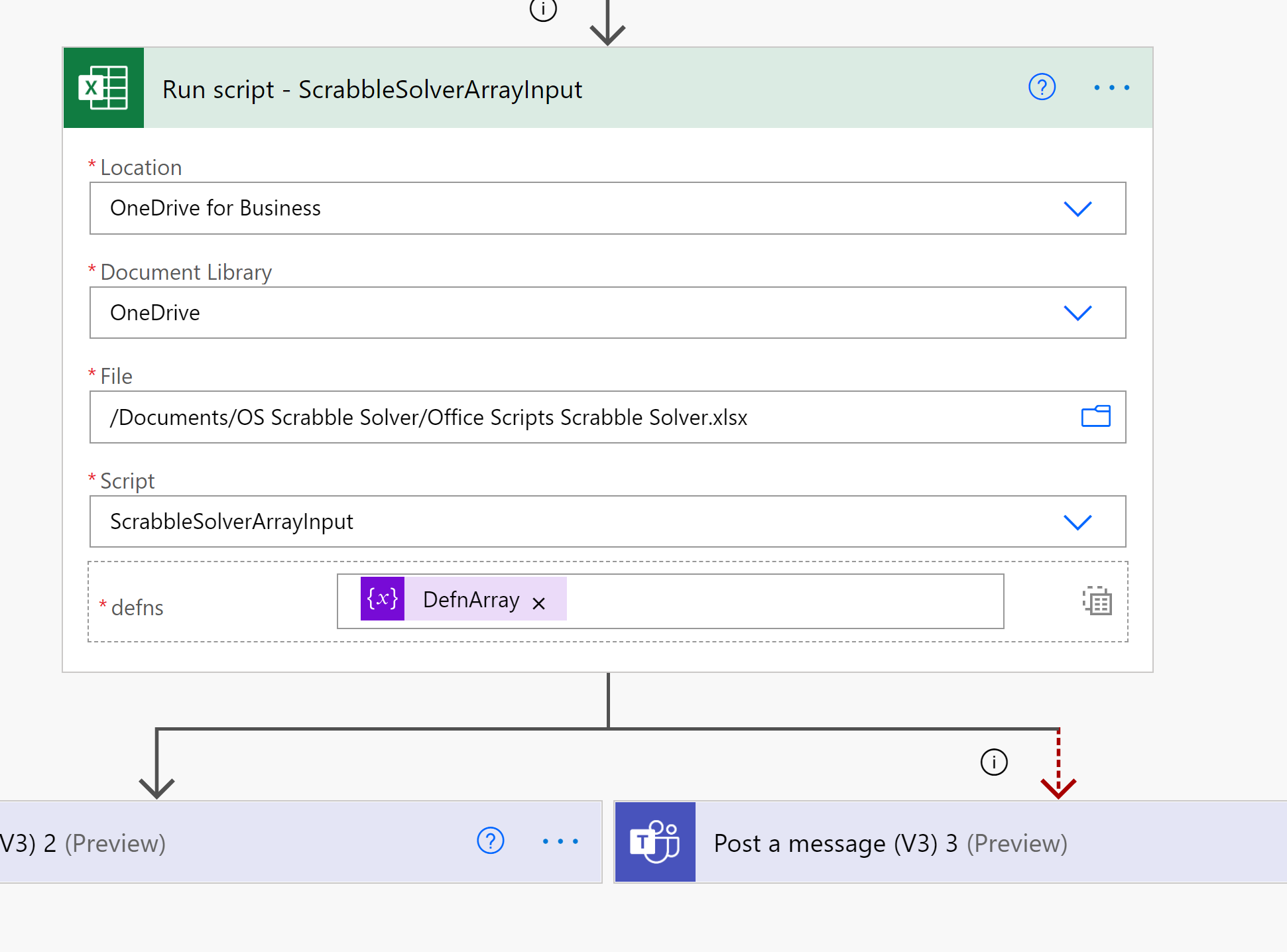
**Fig 10. The HTTP Connector**



The second and subsequent Apply to each loop are identical dealing with the next set of generated words. Only three of these loops are required for the case of a seven letter long input string. The remaining loops exist for the case that the Flow would, in the future, need to manage longer input stings. For the case of the Scrabble game the maximum letter length is seven.

Once the DefnArray has been populated it is passed to the *ScrabbleSolverArrayInput* script which outputs the array items into the Excel worksheet.

**Fig 11. The final Office Script returns the found definitions back into Excel**



A Flow success or fail message is posted to Teams once the Flow has finished running.

**Further Notes**

The HTTP Get Connector uses the following url and syntax, for example,

[**https://api.dictionaryapi.dev/api/v2/entries/en\_GB/trying**](https://api.dictionaryapi.dev/api/v2/entries/en_GB/trying)

**GitHub Repo location**

[**https://github.com/MrAnalyticals/OfficeScripts/tree/main/ScrabbleSolver**](https://github.com/MrAnalyticals/OfficeScripts/tree/main/ScrabbleSolver)

**YouTube Video Location**

[**https://youtu.be/WKB9sGoTlgY**](https://youtu.be/WKB9sGoTlgY)

**The Office Scripts**

**ScrabbleSolverLenOfRack**

function main(workbook: ExcelScript.Workbook):string

{

 let Solver = workbook.getWorksheet('Solver')

 let RackLen = Solver.getCell(3,4).getValue().toString().length

 if(RackLen>7){

 return 'Err'

 }

 else{return 'Good'}

}

**ScrabbleWordGenerator7**

function main(workbook: ExcelScript.Workbook): wordsArray[] {

  let Solver = workbook.getWorksheet('Solver')

  let returnArray: wordsArray[] = []

  let randomSalingerWord: String = ''

  let randomLetterNumber: number

  let randomLetter: string

  let test: string = ''

  let testy: string = ''

  let d4 = Solver.getRange('d4').getValue().toString()

  let d5 = Solver.getRange('d5').getValue().toString()

  console.log('d4: ' + d4)

  if (d5 == '1'){

    Solver.getAutoFilter().clearCriteria()

    Solver.getRange("d5").clear(ExcelScript.ClearApplyTo.contents)

  console.log('D5 cleared')

  Solver.getRange("B9:C11000").clear(ExcelScript.ClearApplyTo.contents)

let D4Str: string = d4.toString()

//let returnArray = []

//Runs anagram for all characters. Level 1

if(D4Str.length <3){

  return

}

returnArray = allAnagrams(workbook,D4Str)

let D4StrLen = D4Str.length

// D4StrLen - 1 characters

let D4StrLenMinus1 = D4StrLen - 1

let shortWordsArray = []

let allAnagramsArray = []

//Iterate through the number of characters to find anagrams for smaller words

if (D4StrLenMinus1 < 2) {

      return

}

if (D4StrLenMinus1 > 2){

let k: number

shortWordsArray = subWords(workbook, D4Str)

let shortWordsArrayLen:number = shortWordsArray.length

 for (let k = 0; k < shortWordsArrayLen; k++) { //Level 2

  allAnagramsArray = allAnagrams(workbook, shortWordsArray[k])

  //console.log('k allAnagramsArray: ' + `${k + allAnagramsArray[k]}`)

  //console.log(k +'k allAnagramsArray: ' + allAnagramsArray)

  returnArray = returnArray.concat(allAnagramsArray)

  }

}

if (D4StrLenMinus1 > 2) {

let l:number

let shortWordsArray1 = []

let shortWordsArray2 = []

shortWordsArray1 = subWords(workbook, D4Str)

let shortWordsArrayLen: number = shortWordsArray1.length

if (shortWordsArrayLen > 2) {

for (let l = 0; l < shortWordsArrayLen-1; l++) { //Level 3

  shortWordsArray2 = subWords(workbook, shortWordsArray1[l])

  if (shortWordsArray2[0].length > 2) {

  allAnagramsArray = allAnagrams(workbook, shortWordsArray2[l])

  //console.log(l + 'l allAnagramsArray Level 3: ' + `${l + allAnagramsArray[l]}`)

  //console.log(l + 'l allAnagramsArray Level 3: ' + allAnagramsArray)

  returnArray = returnArray.concat(allAnagramsArray)

  }

}}

let m: number

let shortWordsArray3 = []

if (shortWordsArrayLen > 2) {

  for (let m = 0; m < shortWordsArray2.length - 1; m++) {//Level 4

    shortWordsArray3 = subWords(workbook, shortWordsArray2[m])

    if (shortWordsArray3[0].length > 2) {

      allAnagramsArray = allAnagrams(workbook, shortWordsArray3[m])

      //console.log(m + 'm allAnagramsArray Level 4: ' + `${m + allAnagramsArray[m]}`)

      //console.log(m + 'm allAnagramsArray Level 4: ' + allAnagramsArray)

      returnArray = returnArray.concat(allAnagramsArray)

    }

  }

}

let n: number

let shortWordsArray4 = []

if (shortWordsArrayLen > 2) {

  for (let n = 0; n < shortWordsArray3.length - 1; n++) {//Level 5

    shortWordsArray4 = subWords(workbook, shortWordsArray3[n])

    if (shortWordsArray4[0].length > 2) {

      allAnagramsArray = allAnagrams(workbook, shortWordsArray4[n])

      //console.log(n + 'n allAnagramsArray Level 5: ' + `${n + allAnagramsArray[n]}`)

      //console.log(n + 'n allAnagramsArray Level 5: ' + allAnagramsArray)

      returnArray = returnArray.concat(allAnagramsArray)

    }

  }

}

}

//Count number of letters in supplied word

//console.log('returnArray: ' + returnArray)

let returnArrayItem: number

let countOfThousands: number = returnArray.length/1000 //1

let countOfThousandsRem: number=  returnArray.length % 1000 //584

//console.log('countOfThousands: ' + countOfThousands)

//output returnArray to worksheet in chunks of 1000 to avoid the overload error.

//let rowVal: number = 8

for (let thousandsCounter = 1; thousandsCounter < countOfThousands + 1; thousandsCounter++){

  for (let returnArrayItem = ((thousandsCounter-1)\*1000);returnArrayItem < thousandsCounter \* 1000; returnArrayItem++){

    Solver.getCell(returnArrayItem+8, 1).setValue(returnArray[returnArrayItem])

  }

  console.log('thousandsCounter: ' + thousandsCounter)

}

//returnArray.forEach((cellobj) => {

//  Solver.getCell(rowVal,1).setValue(cellobj)

//  returnArrayItem++

//})

//remove any duplicates //10952

Solver.getRange("B8:C11000").removeDuplicates([0], true)

//build returnArray

console.log('Started to build returnArray directly from worksheet')

let builtArray= []

let outputRange = Solver.getRange("B9:B11000").getValues()

let RowCounter: number = 0

for (let Cellval of outputRange) {

  if (Cellval.toString() == '') {

    break

  } else {

    RowCounter++

  }

  }

//let rowNumber = RowCounter.toString

RowCounter = RowCounter+8

console.log('rowNumber: ' + RowCounter)

let builtArrayRange = 'B9:B' + RowCounter

builtArray = Solver.getRange(builtArrayRange).getValues()

// remove [" "] from each value?

let builtArrayStr = builtArray.toString() //asdfert

builtArrayStr = builtArrayStr.replace('["','')

builtArrayStr = builtArrayStr.replace('"]', '')

let builtArrayNew = []

builtArrayNew = builtArrayStr.split(',')

//

returnArray =  builtArrayNew

console.log('Length of returnArray: ' + returnArray.length)

console.log('returnArray: ' + returnArray)

// Set horizontal alignment to ExcelScript.HorizontalAlignment.center for range B:B on selectedSheet

Solver.getRange("B:B").getFormat().setHorizontalAlignment(ExcelScript.HorizontalAlignment.center);

Solver.getRange("B:B").getFormat().setIndentLevel(0);

// Set horizontal alignment to ExcelScript.HorizontalAlignment.left for range B1:B7 on selectedSheet

Solver.getRange("B1:B7").getFormat().setHorizontalAlignment(ExcelScript.HorizontalAlignment.left);

return returnArray

}}

interface wordsArray {

  word: string

}

function allAnagrams(workbook: ExcelScript.Workbook,inputWord:string) {

  let shortwordArray = []

  if (inputWord.length < 2) {

    return [inputWord]

  } else {

    let allAnswers = []

    for (let i = 0; i < inputWord.length; i++) {

      let letter = inputWord[i]

      let shorterWord = inputWord.substr(0, i) + inputWord.substr(i + 1, inputWord.length - 1)

      shortwordArray = allAnagrams(workbook, shorterWord)

      for (let j = 0; j < shortwordArray.length; j++) {

        allAnswers.push(letter + shortwordArray[j])

      }

    }

    return allAnswers

  }

}

function subWords(workbook: ExcelScript.Workbook, inputString: string) {

  //number param. currently not being used.

  let shorterwordsArray = []

  let i: number

  let j: number

  let loopString: string

  let inputStringLen = inputString.length

  //inputString[i]  //obtains(reads)each letter in the string

  //Start from character 1 dropping each nth character, from the left, through the entire input string.

  let newShortWord = []

  let newWordSplitA = []

  let newShortWordSplit = inputString.split('') //asdfert

  let newInputString:string = inputString

  for (let i = 0; i < inputStringLen; i++) {

  //Use substring methd to extract ith elmnt from inputstring

  newInputString = inputString

    //console.log('newShortWordSplit[i]: ' + newShortWordSplit[i])

  let nSWS:string = newShortWordSplit[i].toString()

  newInputString = newInputString.replace(nSWS,'')

    //loopString = loopString + newInputString

    //console.log('loopString: ' + loopString)

    newShortWord.push(newInputString)

    loopString = ''

  }

return newShortWord

}

**ScrabbleSolverArrayInput**

function main(workbook: ExcelScript.Workbook, defns: definitionsArray[]) {

  let Solver = workbook.getWorksheet('Solver')

  if(defns.length==0){

    return

  }

let j: number = 0

let k: number = 0

for (let defnsItem of defns) {

  k = j + 8

  Solver.getCell(k, 2).setValue(defns[j])

  j++

}

  // Toggle auto filter on Solver

  Solver.getAutoFilter().apply("B8:C11100");

  // Create a new temporary sheet view

  //Solver.enterTemporaryNamedSheetView() //need this

  // Apply custom filter on Solver

  Solver.getAutoFilter().apply("C8", 1, { filterOn: ExcelScript.FilterOn.custom, criterion1: "<>Word does not exist" });

  console.log('ScrabbleSolverArrayInput Routine finished')

}

interface definitionsArray {

  definition: string

}

**References**

Ref : Free REST API Dictionaries [Free Dictionary API](https://dictionaryapi.dev/), [WordsAPI](https://www.wordsapi.com/)

Ref : [How Many Letter Tiles Are in Scrabble? (thesprucecrafts.com)](https://www.thesprucecrafts.com/how-many-letter-tiles-are-in-scrabble-410933)

Ref : [Words With 'dsfghy' - ScrabbleWordFinder.org](https://scrabblewordfinder.org/solver)

Ref : [(1) Office Scripts: Update large Excel range in performant way - YouTube](https://www.youtube.com/watch?v=BP9Kp0Ltj7U)

By Sudhi Ramamurthy

Ref : [JavaScript and HTML DOM Reference (w3schools.com)](https://www.w3schools.com/jsref/)

Ref : [javascript - Generate all possible combinations of letters in a word - Code Review Stack Exchange](https://codereview.stackexchange.com/questions/57161/generate-all-possible-combinations-of-letters-in-a-word)

**All Scrabble-accepted Two letter words**

aa, ab, ad, ae, ag, ah, ai, al, am, an, ar, as, at, aw, ax, ay, ba, be, bi, bo, by, ch, da, de, di, do, ea

ed, ee, ef, eh, el, em, en, er, es, et, ew, ex, fa, fe, fy, gi, go, gu, ha, he, hi, hm, ho, id, if, in, io, is, it, ja, jo, ka, ki, ko, ky, la, li, lo, ma, me, mi, mm, mo, mu, my, na, ne, no, nu, ny, ob, od, oe, of, oh, oi, ok, om, on, oo, op, or, os, ou, ow, ox, oy, pa, pe, pi, po, qi, re, sh, si, so, st, ta, te, ti, to, ug, uh, um, un, up, ur, us, ut, we, wo, xi, xu, ya, ye, yo, yu, za, ze, zo.

**Scrabble Words With No Vowels and No "Y"**

BRR, BRRR, CRWTH, CRWTHS, CWM, CWMS, HM, HMM, MM, NTH, PFFT, PHPHT, PHT, PSST, SH, SHH, TSK, TSKS, TSKTSK, TSKTSKS,

**Scrabble Words With No Vowels That Use "Y"**

BY, BYRL, BYRLS, BYS, CRY, CRYPT, CRYPTS, CYST, CYSTS, DRY, DRYLY, DRYS, FLY, FLYBY, FLYBYS, FLYSCH, FRY, GHYLL, GHYLLS, GLYCYL, GLYCYLS, GLYPH, GLYPHS, GYM, GYMS, GYP, GYPS, GYPSY, HYMN, HYMNS, HYP, HYPS, LYMPH, LYMPHS, LYNCH, LYNX, MY, MYRRH, MYRRHS, MYTH, MYTHS, MYTHY, NYMPH, NYMPHS, PLY, PRY, PSYCH, PSYCHS, PYGMY, PYX, RHYTHM, RHYTHMS, RYND, RYNDS, SCRY, SHY, SHYLY, SKY, SLY, SLYLY, SPRY, SPRYLY, SPY, STY, STYMY, SYLPH, SYLPHS, SYLPHY, SYN, SYNC, SYNCH, SYNCHS, SYNCS, SYNTH, SYNTHS, SYPH, SYPHS, SYZYGY, THY, THYMY, TRY, TRYST, TRYSTS, TYPP, TYPPS, TYPY, WHY, WHYS, WRY, WRYLY, WYCH, WYN, WYND, WYNDS, WYNN, WYNNS, WYNS, XYLYL, XYLYLS, XYST, XYSTS

**Appendix 1 The “microso” run**

|  |  |
| --- | --- |
| **Words Found** | **Definition** |
| morisco | A Moor in Spain, especially one who had accepted Christian baptism. |
| micros | A microcomputer. |
| corso | (in Italy and some other Mediterranean countries) a social promenade. |
| rosco | Rolling stock (leasing) company. |
| scroo | A small stack of corn, hay, or straw. |
| moros | A Muslim inhabitant of the Philippines. |
| moors | A tract of open uncultivated upland, typically covered with heather. |
| rooms | Share a room, house, or flat, especially a rented one at a college or similar institution. |
| smoor | Braise or stew (meat or fish). |
| osmic | Relating to odours or the sense of smell. |
| micro | A microcomputer. |
| romic | The Romic alphabet or system. |
| moro | A Muslim inhabitant of the Philippines. |
| moor | A tract of open uncultivated upland, typically covered with heather. |
| room | Share a room, house, or flat, especially a rented one at a college or similar institution. |
| mooi | Pretty; good; nice. |
| mico | Among the people of the Creek Confederacy and some of their neighbours: a chief or ruler. |
| micr | Magnetic ink character recognition. |
| crim | A criminal. |
| cor | Expressing surprise, excitement, admiration, or alarm. |
| roc | A gigantic mythological bird described in the Arabian Nights. |
| ocr | short for optical character recognition |
| orc | (in fantasy literature and games) a member of an imaginary race of humanlike creatures, characterized as ugly, warlike, and malevolent. |
| mro | Mauritanian ouguiya(s). |
| mor | Humus formed under acid conditions. |
| rom | A member of the Roma people; a Romani person. |
| omr | Omani rial(s). |
| moi | Me (used in questions when accused of something that one knows one is guilty of) |
| imo | International Maritime Organization. |
| iom | Isle of Man. |
| omi | In early imperial Japan: (a hereditary title for) the head of clan or family that claimed imperial ancestry, who usually also served as a high-ranking official in the Yamato court. Also: an official title for the leader of such a clan. Compare muraji , o-muraji , o-omi . |
| mic | A microphone. |
| mci | Millicurie(s), a quantity of a radioactive substance having one thousandth of a curie of radioactivity. |
| cim | Computer-integrated manufacturing (or manufacture). |

**Appendix 2 The “micros” run [20 minutes duration]**

|  |  |
| --- | --- |
| **Words Found** | **Definition** |
| Micros | A microcomputer. |
| Coirs | Fibre from the outer husk of the coconut, used in potting compost and for making ropes and matting. |
| Csiro | (in Australia) Commonwealth Scientific and Industrial Research Organization. |
| Miros | An evergreen coniferous New Zealand tree which yields useful timber. |
| Osmic | Relating to odours or the sense of smell. |
| Crims | A criminal. |
| Scrim | Strong, coarse fabric, chiefly used for heavy-duty lining or upholstery. |
| Micro | A microcomputer. |
| Romic | The Romic alphabet or system. |
| Rocs | A gigantic mythological bird described in the Arabian Nights. |
| Orcs | (in fantasy literature and games) a member of an imaginary race of humanlike creatures, characterized as ugly, warlike, and malevolent. |
| Miso | Paste made from fermented soya beans and barley or rice malt, used in Japanese cooking. |
| Mics | A microphone. |
| Micr | Magnetic ink character recognition. |
| Crim | A criminal. |
| Csr | Comprehensive spending review. |
| Rcs | (in the UK) Royal College of Scientists. |
| Rsc | Royal Shakespeare Company. |
| Scr | Seychellois rupee(s). |
| Mrs | The title used before a surname or full name to address or refer to a married woman without a higher or honorific or professional title. |
| Rms | Royal Mail Ship. |
| Rsm | (in the British army) Regimental Sergeant Major. |
| Mis | Management information systems. |
| Msi | (In Italy) the Italian Social Movement, an extreme right-wing anti-communist political party founded in 1948 (renamed the Alleanza Nazionale (National Alliance) in 1994). |
| Ism | A distinctive practice, system, or philosophy, typically a political ideology or an artistic movement. |
| Sim | A video game that simulates an activity such as flying an aircraft or playing a sport. |
| Mic | A microphone. |
| Mci | Millicurie(s), a quantity of a radioactive substance having one thousandth of a curie of radioactivity. |
| Cim | Computer-integrated manufacturing (or manufacture). |

**Appendix 3 The “micro” run**