

## Parsing JSON with SAS® in 2017

John Kennedy, Mesa Digital

### ABSTRACT

Technology service company BuiltWith reported that, as of March 2017, 94.1% of the top million most visited websites use JavaScript. All of these JavaScript websites use JSON as their standard data format to store and transmit data. Base SAS has many powerful features for interpreting JSON, including the JSON libname, introduced in SAS 9.4. This paper describes, in brief, several methods for reading in JSON data to SAS, weighing their pros and cons and guiding users on when to use each. It also involves Henry Winkler.

### INTRODUCTION

John is a SAS programmer for a large financial company who recently lost a large multiple of his yearly bonus in Henry Winkler's Fonzie scheme. To try to recoup his money, John plans to use SAS Predictive Analytics to predict the winning combination for the New York State Lottery, valued at \$73 million as of March 2016. John is really desperate.

The New York Lottery maintains a feed of the winning jackpot combinations on the data.ny.gov API page. The winning combination data, as well as the metadata that describes it, are stored in a JSON file at a static URL.

Technology service company BuiltWith reported that, as of March 2017, 94.1% of the top million most visited websites use JavaScript. Within JavaScript, the standard data format is JSON. As a result, the vast majority of the top million global websites use the JSON format to transfer and store data. Before SAS 9.4, reading in JSON data either involved a clunky DATA step or using a PROC step to convert JSON to a format of similar structure that SAS could read. None of these methods worked across the board or captured the full depth of JSON functionality, but can be useful in certain situations.

### PROC JSON DISCLAIMER

To read in his lottery data, John will not be using Proc JSON. Proc JSON only exports JSON data, and cannot be used to import it. Which method John will use to import his JSON depends on his needs and how much he knows about the data's structure.

### USING DATA STEP TO PARSE PRIMITIVE JSON

If John knows the exact structure of the JSON data, he can use a DATA step or PROC DS2 to read JSON data into a dataset. Base SAS has many powerful string processing features, which can be used to process primitive, linear JSON. Linear JSON has only a single layer of attributes, and doesn't contain the data types that make JSON so useful.

```
data temp;
  length attribute value $80;
  infile datalines;
  input @;
  if _infile_ ne: '{' and _infile_ ne: '}';
  input attribute: $quote. x $ value: $quote.;
  put attribute 'is' value;

datalines;
{
```

```

    "name" : "John",
    "position" : "intern"
}
;
run;

```

<pre> Name is John Position is intern </pre>
--

### Output 1. Output from a DATA step to process linear JSON

## USING PROC DS2 TO IMPORT JSON

If John knew that his JSON sample had a consistent number of lines, he could account for an extra data structure, like an array, with this simple data processing, but the ease of use of the JSON libname and other methods make advanced use of a DATA step overkill. If John didn't know how many lines his JSON sample had, or if he wanted to read in from a Web-based stream, he could use Proc DS2 to parse JSON.

```

method parseJSON();
  dcl int tokenType parseFlags;
  dcl nvarchar(128) token;
  rc=0;
  *
  do while (rc=0);
    json.getNextToken( rc, token, tokenType, parseFlags)

    * subject line;
    if (token eq 'attribution') then
      do;
        *New York State Gaming Commission
        json.getNextToken( rc, token, tokenType, parseFlags);
        name=token;
      end;
  end;

```

Again, this all depends on John knowing the exact format of the JSON. Proc DS2 is useful when the functionality of DS2's features - embedded SQL procedure, threading - overweighs the ease of use of the JSON libname. If John wanted to use more of the robust functionality of JSON, or if he didn't know the structure of the JSON data, he could convert it to XML then use the SAS XML libname engine to import it into a SAS dataset.

## USING XML ENGINE TO IMPORT JSON

### CONVERTING TO XML

Structurally, JSON and XML share many similarities, although JSON has an extra array data type that XML lacks. We can use a PROC step, such as Proc Groovy, to convert JSON data to XML, then use the SAS XML engine to interpret the data. Proc Groovy is a procedure step that allows for the execution of Groovy language from within a SAS Program. Groovy is a programming language for the Java platform, supported by Apache.

## READING THE JSON AND CONVERTING TO XML

```
proc groovy;
  /* showing proc groovy where my json java files are */
  add classpath = "javajson.jar";

  submit; /* starting groovy */
  import org.json.JSONObject;
  import org.json.XML;

  /* loading json from file */
  def json_string = new File("test.json").text;
  /* converting json to xml */
  def json = new JSONObject(json_string);
  def xml = XML.toString(json);

  /* saving to file */
  def out = new PrintWriter("test.xml");
  /* adding xml headers */
  out.println("<xml>", xml, "</xml>");
  out.close();

endsubmit;
```

For creating a dataset from the JSON, creating a new XML file and then reading it from SAS is necessary, as it circumvents going through the Macro layer and passing the data directly to SAS through Java. After the Groovy step, a dataset can easily be created from the JSON data using the SAS XML Engine.

```
libname x xml "test.xml";
libname dat "test.dat";

/* creating dataset of phone numbers for person */
data dat.phoneNumbers;
set x.phoneNumbers;
run;

proc print data=dat.phoneNumbers;
run;
```

## WHEN TO USE XML METHOD

The XML engine conversion method should only be used when the solution requires a function within the Groovy procedure. The JSON libname is easier to use than converting to XML, and can be used with more robust JSON data, as the XML conversion method cannot convert the array object within JSON.

## USING THE JSON LIBNAME

The JSON libname is the newest, sexiest, and easiest way to import JSON data into SAS. It handles the parsing of the data, can account for a variable data structure, and can be combined with other procedures to increase its functionality.

```
filename response temp;

proc http
```

```
url="http://data.ny.gov/api/views/kwxv-fwze/rows.json?accessType=DOWNLOAD"  
method= "GET"  
out=response;  
run;  
  
libname lotto JSON fileref=response;  
  
proc print data=lotto.data;  
run;
```

## WHEN TO USE JSON LIBNAME METHOD

Unless the problem requires extra functionality beyond the dataset manipulation in SAS, the JSON libname method should be used to read in JSON data. It takes all the string processing done with the PROC DS2 or DATA step methods out of the equation, and allows for more structural flexibility than the XML conversion method. Although, due to its age, its documentation is slim, it is already the new standard for reading in JSON data into SAS.

## CONCLUSION

This paper is a brief summary of several strategies to read in JSON data to a SAS dataset. Depending on the length of your data, and how much of its structure you know ahead of time, you can use a DATA or PROC step, but the JSON libname offers the most flexibility and efficiency, and should be tried first.

## ACKNOWLEDGMENTS

Thanks to Salman Maher, Rick Langston, Anand Vijaraghavan, Henry Winkler, and Chris Hemedinger.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

John Kennedy  
Mesa Digital  
john@mesa.digital  
<http://mesa.digital>