# JotForm Lambda Docs

## Overview:

These docs will be used as a reference to our JotForm Lambda function. The flow starts in JotForm.

Flow:

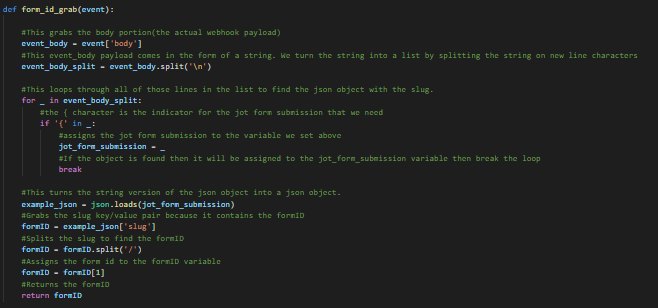
JotForm -> Lambda -> S3 -> Glue -> Athena -> QuickSight

The idea is each JotForm has a webhook that attaches to an endpoint URL that is made from API Gateway and lambda interacting. From there the lambda function determines if it is a new JotForm or a JotForm that already exists in S3. From there depending on if it is new or exists, it either utilizes the webhook payload(if exists), or API(if it is a new form). Below there will be a detailed section that talks about the different python files involved, and the functions within those files.

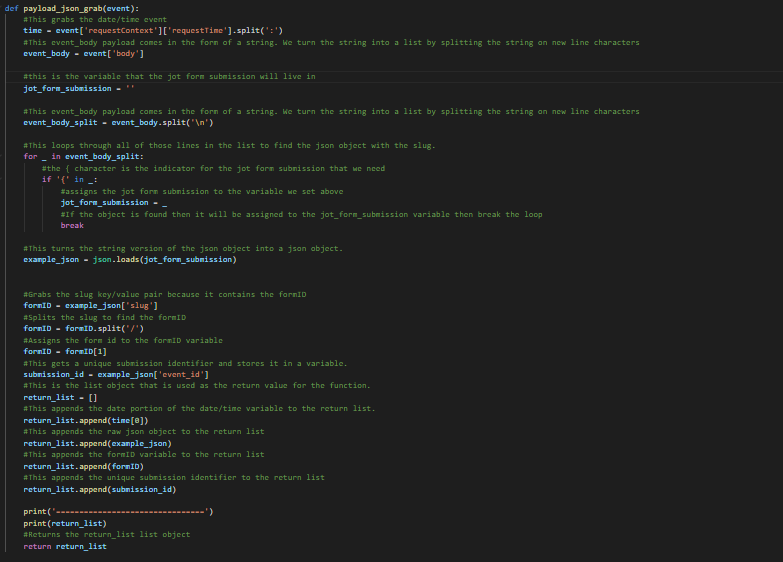
## webhook\_integration.py:

(Necessary Libs: JSON, Boto3)

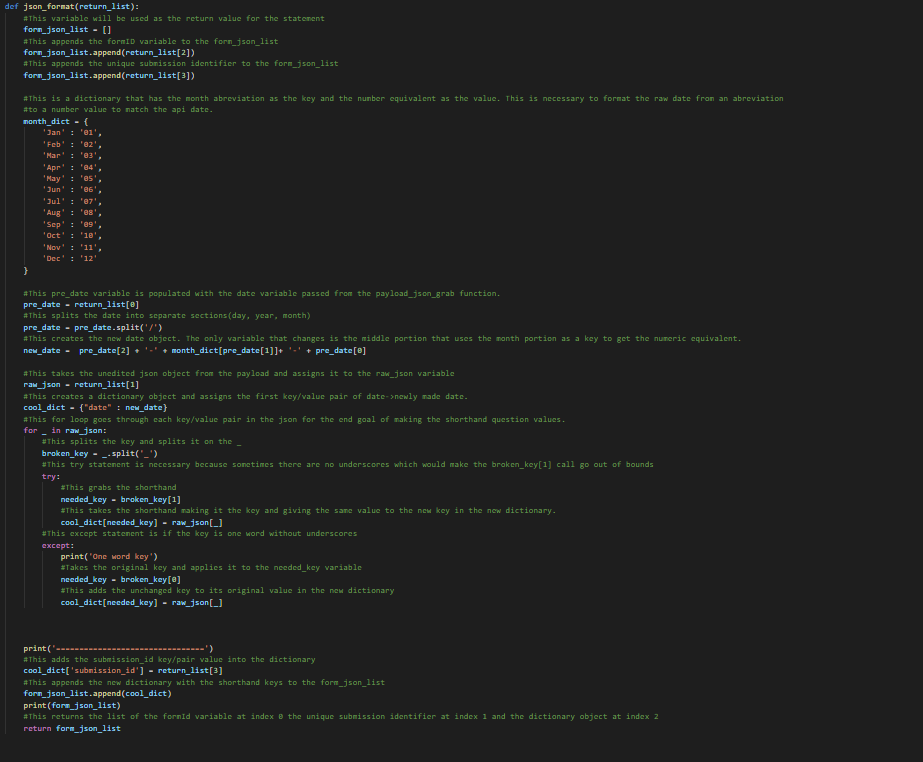
### form\_id\_grab(event):

* This function grabs the unique JotForm ID from the webhook payload. The reason for this function is a quick variable reference for other functions.
* 
* Returns
  + formID -> String

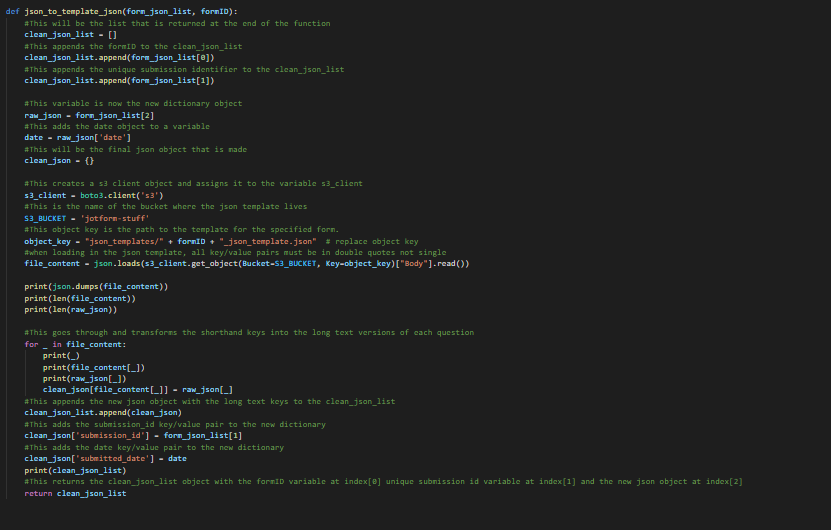
### payload\_json\_grab(event):

* This function is responsible for grabbing the webhook payload. It grabs the necessary data from the payload. There is a Json object that is grabbed, date object that is grabbed, and the unique payload event ID that is grabbed
* Returns
  + List
    - Date - > String
    - example\_json -> Dict
    - formID -> String
    - submission\_id -> String
* 

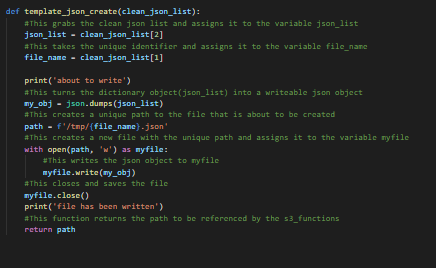
### json\_format(return\_list):

* This function changes the date to a universal date to match the API date. It also formats the Json object to a more malleable state. It also turns the questions within the given Json object into the shorthand version of the questions to match the dictionary created in the template Json created in the api\_integration.py file
* Returns
  + List(form\_json\_list)
    - formID -> String
    - submission\_id -> String
    - cool\_dict -> Dict
* 

### json\_to\_json\_template(form\_json\_list, formID):

* This function takes the Json made in the json\_format function and matches it with the generated Json template from the api\_integration.py file. This makes a uniform submission for all submissions generated from either the API or the webhook
* Returns
  + clean\_json\_list -> Dict
* 

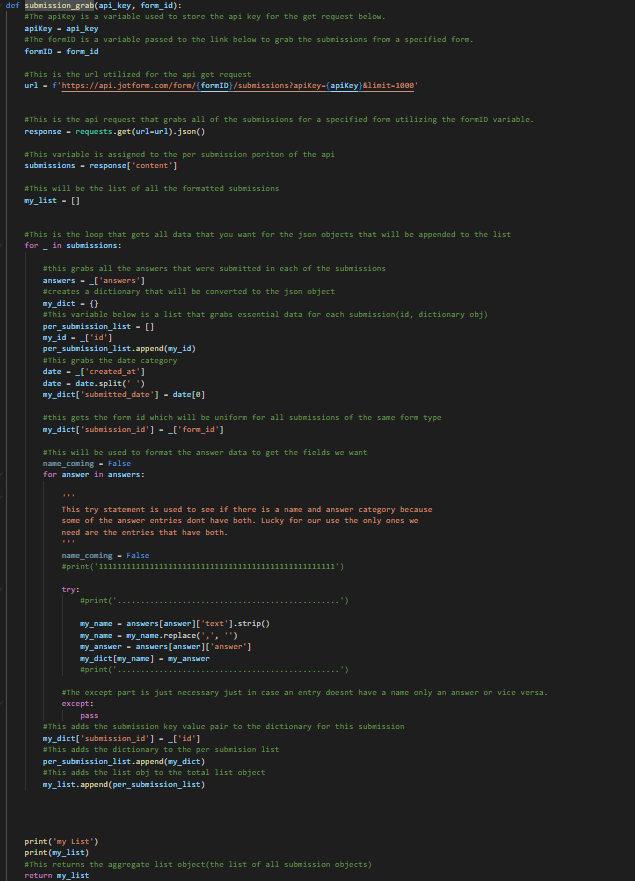
### template\_json\_create(clean\_json\_list):

* This function takes the Json object from the json\_to\_json\_template function and writes it to a temporary file within the lambda directory.
* Returns
  + path -> String
* 

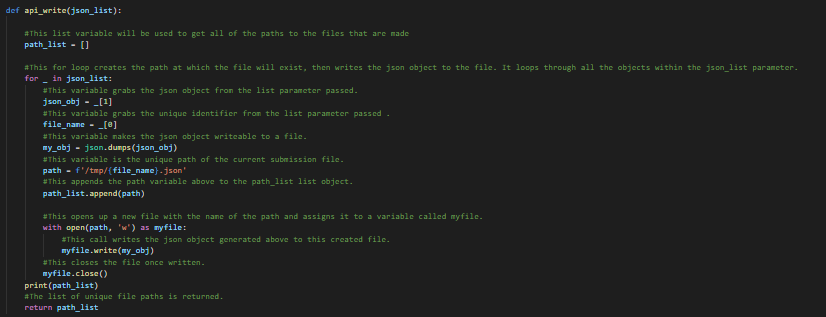
## api\_integration.py:

(Necessary Libs: JSON, Requests, Boto3)

### submission\_grab(api\_key, form\_id):

* This function is used to go through and grab all of the submissions for a JotForm and turn all of the submissions into formatted Json objects.
* Returns
  + List
    - per\_submision\_list -> list
* 

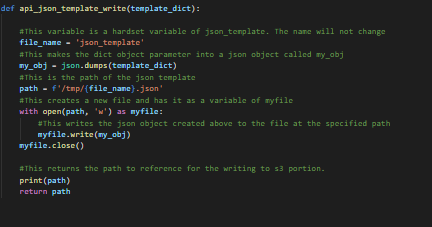
### api\_write(json\_list):

* This function is responsible for writing all the Json objects created in the submission\_grab function to a temporary lambda directory.
* Returns
  + path\_list -> list
* 

### api\_template\_build(api\_key, form\_id):

* This function is responsible for building the Json template that will be used to make the webhook Json objects. This Json template will map the shorthand questions to their long text counterparts.
* Returns
  + template\_dict -> Dict
* 

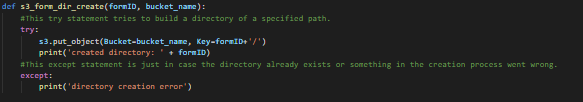
### api\_json\_template\_write(template\_dict):

* This function is responsible for writing the Json template built in the api\_template\_build function and write it to a temporary lambda directory.
* Returns
  + path -> String
* 

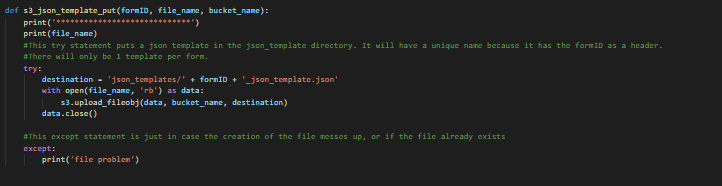
## s3\_methods.py:

(Boto3)

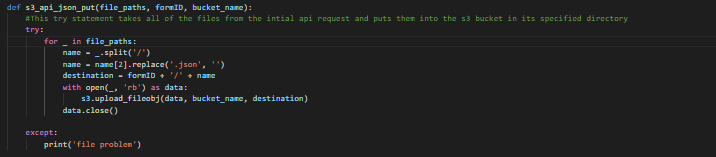
### s3\_form\_dir\_create(formID, bucket\_name):

* This function is responsible for creating a directory for a JotForm. The directory’s name will be the formID, which will store all of the submissions.
* 

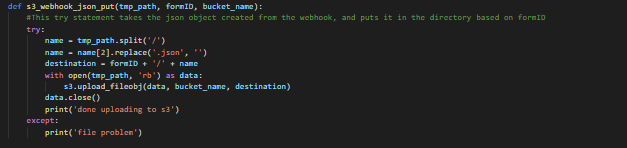
### s3\_json\_template\_put(formID, file\_name, bucket\_name):

* This function is responsible for taking the template generated in the api\_integrations.py file and putting it in the json\_templates/ directory in the s3 bucket.
* 

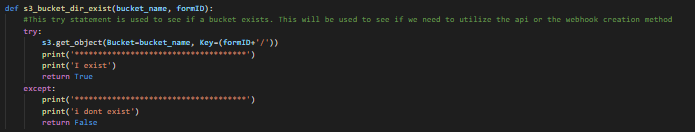
### s3\_api\_json\_put(file\_paths, formID, bucket\_name):

* This function is responsible for putting all of the submissions generated from the initial api call into the proper directory dependent on what the JotForm ID is.
* 

### s3\_webhook\_json\_put(tmp\_path, formID, bucket\_name):

* This function is responsible for putting the webhook submission into the s3 directory dependent on what the JotForm ID is.
* 

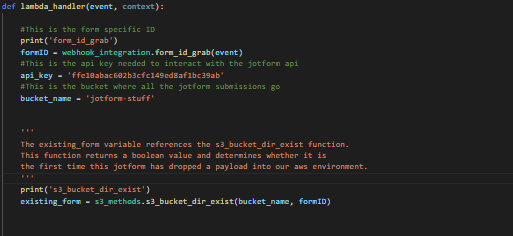
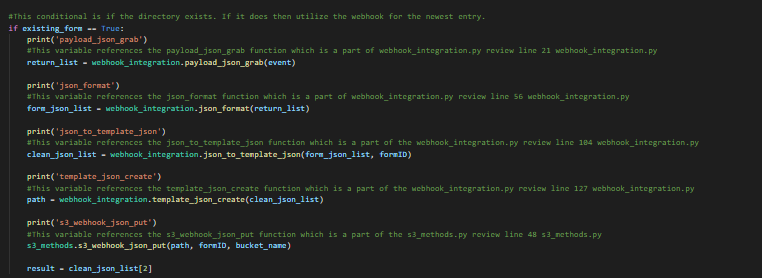
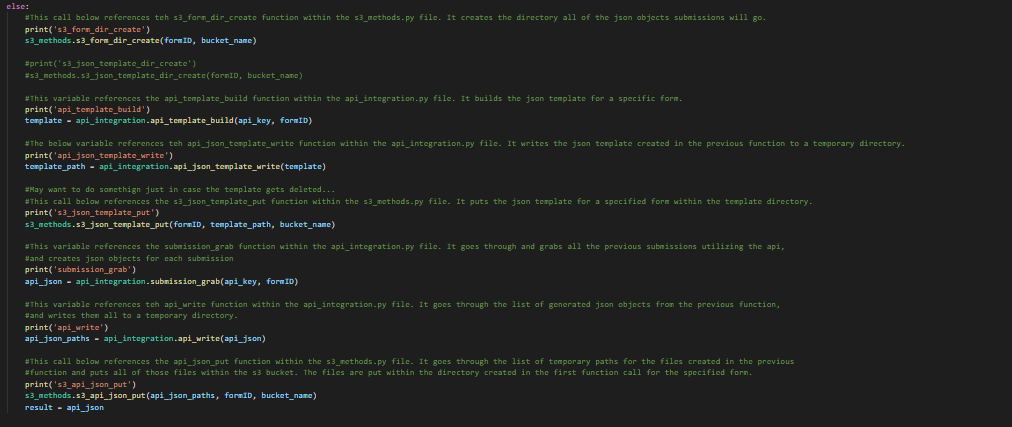
### s3\_bucket\_dir\_exist(bucket\_name, formID):

* This function is responsible for checking to see if a directory exists already in a given bucket.
* Returns
  + Boolean
* 

## lambda\_function.py:

(Necessary Libs: JSON, Boto3, s3\_methods, api\_integrations, webhook\_integrations)

### lambda\_handler(event, context):

* This function is responsible for executing all other functions in the proper order.
* Returns
  + Json object
* 
* 
* 
* 