



For better resolution, less blur, see the comments at the end.

There is a good chance Twilio will not let you use their logo. Check before publishing. Also see later comment about copyright of Twilio page graphics.

## Getting Weather Forecast with Twilio WhatsApp API, Function, and Assets

Created by Max NG – 1 Jan 2019

Tracked changes by Dave Hood. Please think of these as suggestions for your own evaluation. Do not hesitate to ignore suggestions that don't serve your purpose.

My review of Max Ng's draft, used by permission.  
Max describes the target audience thus:

*The targeted audience is technical developers, Twilio wanted the article to be published in their developer community blog. So I would say the audience should be semi technical to technical. The blog is a sharing of ideas of what is possible and folks can either replicate it or use it in some form.*

Version Control		
24 Feb 2019	Max NG	First formatted draft

## Introduction

WhatsApp ~~messagingmessaging~~ is a very popular digital communication app in many parts of ~~ASIA~~Asia, especially here in Singapore. ~~It is an app that e~~Even my Mum ~~used it~~uses it daily for communication with her friends and family, organizing outings and sharing her life. She gets her daily weather forecast by regions-region, either by calling a local landline number or using another app on her ~~android~~Android. ~~It~~This app all started when ~~Mum~~she gets got caught out in a rainstorm when visiting my brother in a different town. She ~~laments-lamented~~that she should have called ahead to ask what the weather ~~is-was~~ like before traveling. [Does Singapore English prefer to spell it travelling?]

NEA defines regions in the 24-hour and 4-day forecast options as {west, east, central, south, north}. These options return considerably more complex responses. Your app deal only with 2-hour forecasts, which are subdivided into so-called areas (not Town Areas). At a minimum, I recommend you avoid the term region. Optionally, if you wish to round out your reader's view, you might say something about the other choices, and why you chose only the 2-hour forecast.

I went online to have a look at what ~~are-was~~ available on the Internet. Singapore's National Environment Agency (NEA) ~~here in Singapore that~~ manages the local landline audio weather forecast service and publishes ~~the~~-forecasted data. It actually also exposes the data sets via API for application developers at: <https://data.gov.sg/dataset/weather-forecast>.

I didn't want to manage any of the components, so I looked~~looking~~ at the Twilio platform, ~~they have~~which includes the WhatsApp ~~messaging~~ API, Function ~~-- for an~~-driven runtime environment and Assets for storing data sets [consistency: dataset is also ok, but you use data set elsewhere]. That ~~lookeds~~ like a good start to get something out and at the same time learn more about the Twilio platform during the Christmas holiday season.

## Workflow of WhatsApp Weather ~~forecast~~Forecast App

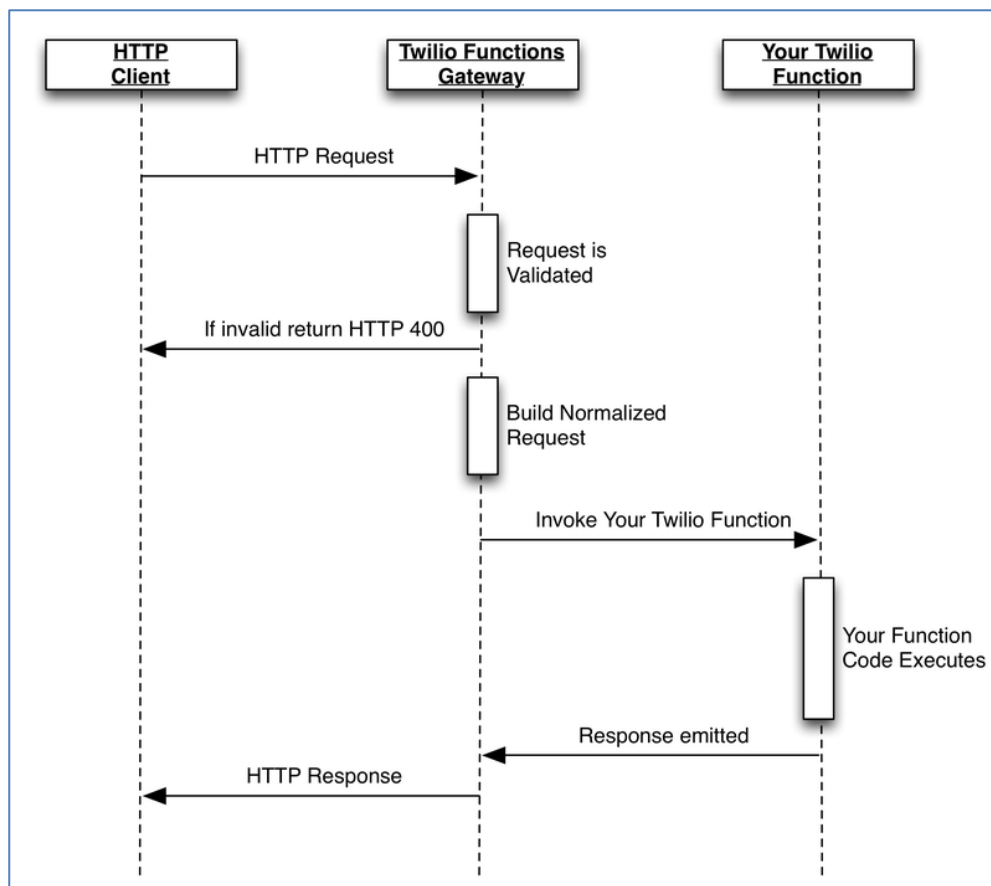
[consistency: you use title case elsewhere in Heading 1]

~~The~~At a high level, ~~view of~~ the App works by ~~looking up a~~searching a JSON key-value array, based on the WhatsApp message body, comparing Town Area ~~Key~~ key [The schema uses only the term area.] and responding back with the ~~correct-forecast~~ value. This app uses the Twilio ~~Whatsapp~~-WhatsApp communication API and Twilio's Function and Assets services. Function is the runtime application logic using nodejs ~~to do the matching~~. Assets stores the daily weather data ~~source~~-obtained ~~from-via~~ NEA's API. At the time of creating the app, Twilio Assets ~~does-did~~ not support an API for updating the data sets, which-and can only be updated via the Twilio Console.

The same application flow can be replicated on other Cloud-cloud Platform platforms that have storage and runtime environments (Google App Engine or

Amazon Lambda). However, for simplicity and cost ~~benefits~~reasons, running everything ~~under-on~~ the Twilio ~~Platform-platform is~~ makes it very attractive.

The Twilio ~~Whatsapp~~WhatsApp communication API has a webhook that can perform event triggers to external HTTP URLs. The ~~below following~~ flow diagram is taken from Twilio online documentation. [if this document is for publication, you need to check for copyright permission, and cite the source precisely (e.g., with a URL)] ~~illustrates the flows.~~



The application logic is contained within Twilio Function. It is an ~~event-event-~~ driven environment, ~~and is~~ invoked when ~~there is an incoming~~ HTTPS request arrives. The HTTPS request can originate either directly from the Internet or from the Twilio WhatsappWhatsApp Gateway [The figure shows Twilio Functions Gateway. Would that be a better designation?].

The data source of the weather reports is stored on Twilio Assets, not shown on this diagram. ~~It-but~~ is loaded into the application logic whenever the Function is invoked.

The document file contains a section break here. Sections can serve useful functions, but they can also have unintended consequences that can be hard to track down, for example with consistent margins or page numbers. I recommend using section breaks sparsely.

You can get a new page by inserting a page break, but even here, my own preference is to check the *page break before* attribute box of the following

paragraph. If you always want a page break, you could build that attribute into your Heading 1 style.

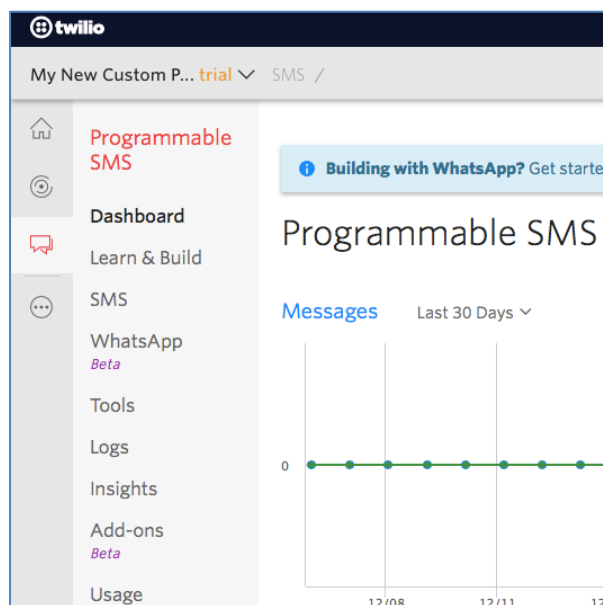
## Application Environment Setup

~~With Twilio Function, what I did essentially is used it even for my code development environment.~~ I did not bother to install nodejs and the necessary npm [Does your target audience know about npm? ReGotcha? 2FA, etc?] on my local laptop. ~~All I did all~~ development work ~~gets done~~ directly on ~~the Twilio~~ Function. When I'm done with my testing and satisfied with my development, all I need to do is change the URL of the new function on my WhatsApp webhook, essentially making the environment DevOps ready.

If I need to do further enhancement in future, I can set up another Function as a development environment, without ~~impacting-affecting~~ my production. This reduces environment setup overhead for development and production. The only setback, ~~one of the days~~ in January, ~~the~~ ReGotcha was unavailable for several hours and I did not have 2FA set up for my Twilio account. I was essentially locked out ~~from of~~ my development environment. ~~When I'm done with my testing and satisfy with my development, all I need to do is change the URL of the new function on my WhatsApp webhook, essentially turning the environment into DevOps ready.~~

## Setting up your device with Twilio WhatsApp Sandbox

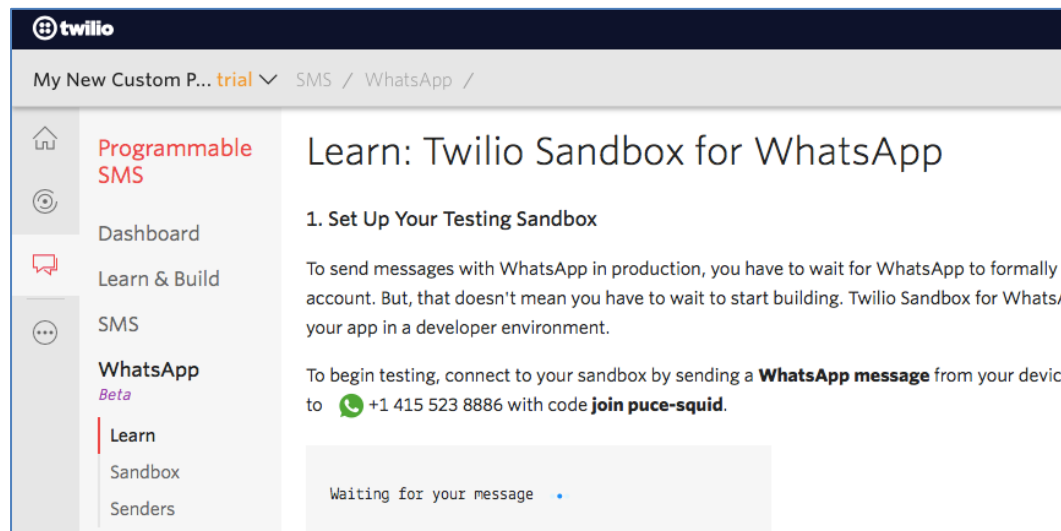
First, Login-log in to your Twilio Account, and select Products – Programmable SMS. Under the ~~Panel-panel~~ on the left, you should see the WhatsApp option. Let's configure ~~the~~ Twilio WhatsApp to interact with your-the ~~Whatsapp~~WhatsApp Account that we assume already exists on your device.



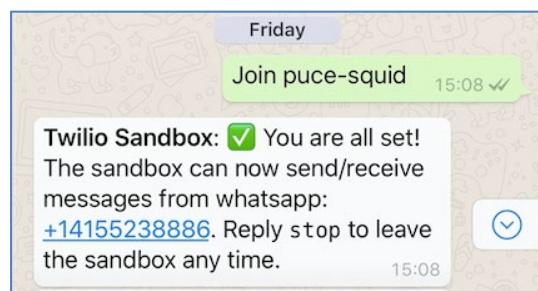
I scaled the figures up for legibility, and added borders. See additional discussion of screenshots at the end.

We ~~will~~ need to register and Opt-In our individual WhatsApp accounts to Twilio WhatsApp. Enforced by WhatsApp, this security feature allows Twilio to send messages to your WhatsApp account, ~~this is a security feature enforced by WhatsApp~~.

To do this, we ~~will~~ need to connect to our “sandbox” by sending a WhatsApp message from our device to +14155238886, with a message join puce-squid or to whatever Phone phone Number number your sandbox screen is showing on your Twilio Console, with a message join puce-squid.



Once you have successfully “registered” your device with your Twilio ~~Whatsapp~~ WhatsApp sandbox, your ~~whatsapp~~ WhatsApp account on your device should receive this message.



At this stage, your ~~whatsapp~~ WhatsApp account has been put into a context with your Twilio Sandbox, ~~and t~~ This will last 24 hours, allowing Twilio to send you ~~Whatsapp~~ WhatsApp messages. To see the list of successfully registered ~~Whatsapp~~ WhatsApp accounts with your Twilio ~~Whatsapp~~ WhatsApp sandbox, you can navigate to Sandbox under ~~Whatsapp~~ WhatsApp panel on the left (following figure). I have registered Mum’s ~~whatsapp~~ WhatsApp account with my sandbox so she can ask for weather forecast with her ~~whatsapp~~ WhatsApp. Robustness: Does someone have to re-register every 24 hours? How do you set up a permanent association?

This figure would benefit from the graphics techniques described at the end.  
Good to use the same color for all redaction bars.

Second, create a Twilio Asset to store the JSON data set containing ~~Town~~  
~~A~~areas and ~~Weather-weather~~ ~~Forecast~~forecasts

The ~~format of the~~ data set is a JSON object file. ~~This-It~~ is obtained by querying NEA's public API to obtain the key-value table listing ~~Town-Area-forecast~~  
~~pairss~~. The NEA developer website page has more details on how to obtain the data using ~~either~~ a query based on date, ~~or~~ date\_time, ~~or neither~~.

Using cURL, I obtained this JSON data

```
curl -v -X GET https://api.data.gov.sg/v1/environment/2-hour-weather-forecast?date_time=2018-12-23T11:40:46 -o weather.json
```

This will output the JSON data from NEA website into a file called weather.json. The file is then reformatted and uploaded to Twilio Assets [Reformatted how? Do you need to say more about these details?]. Uploading of the JSON file is currently ~~is~~ only supported from the Twilio Console. Update of file to Twilio Assets using an API is currently on the Twilio roadmap.

The format of weather.json looks like this, note the name of the JSON array ""forecasts"".

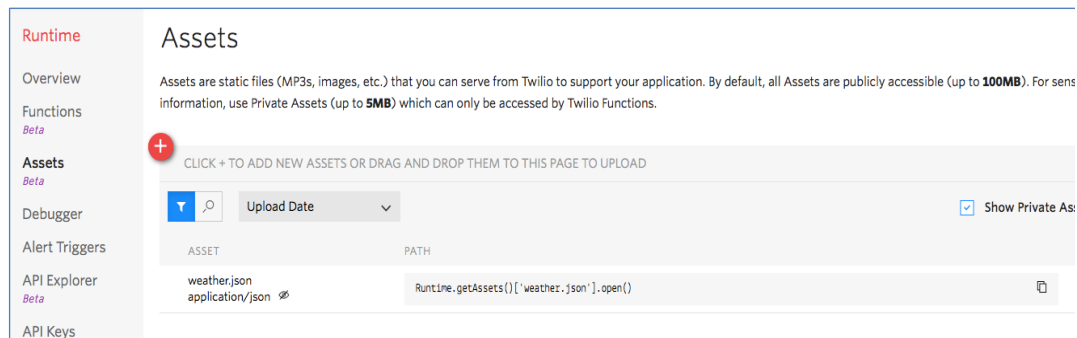
```
{ "forecasts": [
  { "area": "ang mo kio", "forecast": "Windy" },
  { "area": "bedok", "forecast": "Windy" },
  { "area": "bishan", "forecast": "Windy" },
  { "area": "boon lay", "forecast": "Windy" },
  { "area": "bukit batok", "forecast": "Windy" },
  { "area": "bukit merah", "forecast": "Windy" },
  { "area": "bukit panjang", "forecast": "Windy" },
  { "area": "bukit timah", "forecast": "Windy" },
```

Query of the above link returns something that looks like:

`{"area": "Toa Payoh", "forecast": "Partly Cloudy (Day)"}`.

Compared to your list, observe the difference in area capitalization and the extra text *(Day)* in the forecast.

When you download the JSON data from NEA, it is formatted in JSON string~~s~~. I've reformatted it in JSON Objects. You could write a parser to perform this. [the code below uses an existing method `JSON.parse(file)`] To validate the file in JSON format, you can use this online tool: <https://jsonlint.com>.



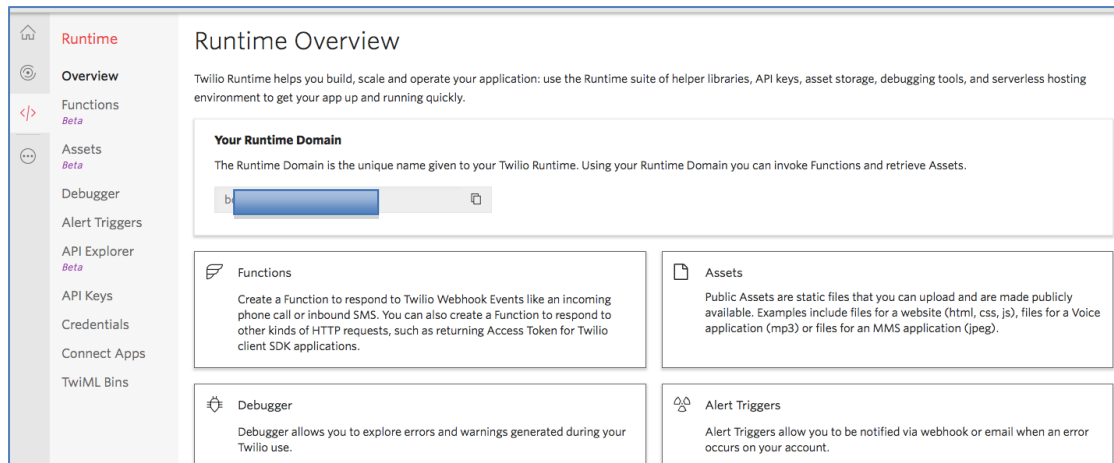
This figure is mostly illegible. If you cannot improve the figure to add value, consider deleting it.

When you uploaded a file to Twilio Assets, you have the option to make the asset ~~to be~~ either public or private. If you choose a Public asset, you allow access to the file using standard HTTPS GET from anywhere on the Internet. If you check the box to set it to Private, the asset will only be accessible by your Twilio function. Setting this option can only be done during the first creation.

I ~~have chosen~~ Private Asset for the weather.json file. Note the information shown on the console, [`Runtime.getAssets()['weather.json'].open()`]; you will need to use this method to retrieve your private asset. We will explore this later in our code.

Third, create a Function for ~~our~~ the application code

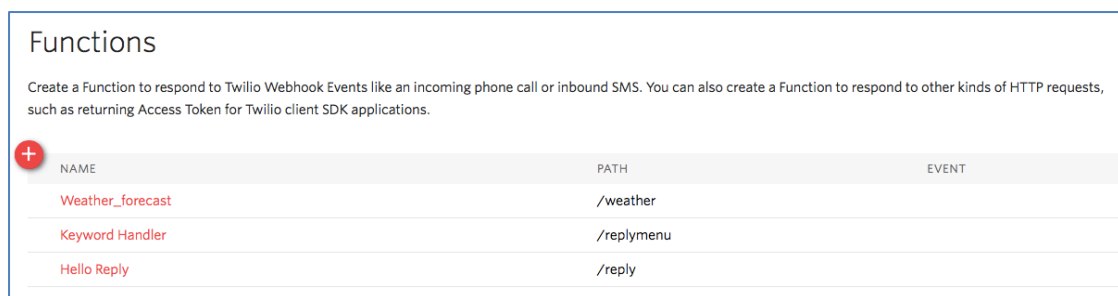




Keep in mind that the original screenshot still exists behind the redaction mask. If you distribute this as a Word file, anyone can easily see the original. Consider the techniques described at the end.

Toggle to Twilio Function. On the left panel [left panel of what?], select **All Products**, and then **Runtime**. You will be presented with a menu of all available Runtime developer tools.

Select Functions from the menu. When presented with the option, choose Add a new Function, and select a blank template. Note the domain of this new Function. The domain for the Function will be used on our earlier WhatsApp console for the webhook triggered. Add a unique path for this function. The complete domain and path name will be the URL used to invoke this Function. I called my Function “*Weather\_forecast*”.




Blurred, too small to easily read

During the development and testing phase, unchecking the ticked box “Access Control – Check for valid Twilio signature” [this text suggests that the checkbox appears in the figure above. If you don’t want to include a screenshot of the options frame (probably the right thing), consider moving the text into the previous paragraph?] allows us direct access to this function. The Twilio platform will not check for a valid signature token when receiving aan https request on this function. For quick testing, you can then POST a-HTTPS by sending a cURL message to the same URL to emulate a WhatsApp message send.

## Weather\_forecast

**Properties**

FUNCTION NAME

PATH  /weather 

**Configuration**

ACCESS CONTROL ☐ Check for valid Twilio signature

Word does not do well at reliably overlaying multiple graphic images with precise positioning. In addition, the original information is still there, as you can see above. See further discussion about graphics in my comments at the end.

### Application Code breakdown

In terms of document structure, this is the only level 3 heading under its level 2 parent, generally not a good thing. In terms of format, it looks as if it is a level 2 heading. But your L2 headings form a sequence of steps, so this one doesn't fit. I suggest you turn it into Normal text, perhaps boldface for emphasis.

We ~~will~~ need to access ~~our earlier~~the JSON data sets ~~previously~~ stored in ~~our~~ Twilio Assets ~~marked as Private~~[does this need to be restated?], and parse the file into JSON objects [The text under heading *Second...* suggests that this has already been done. Should you have more detail? If so, where?]. ~~This will be~~The JSON objects are used for key comparison. The "body" of the WhatsApp message ~~will be~~and the JSON object keys are converted to LowerCase for comparison.

```
exports.handler = function(context, event, callback) {
    const body = event.Body ? event.Body.toLowerCase() : null ;
```

1) Can you not write `var body = ...?`

2) If event.Body were null, would the TRUE branch of the option do the right thing anyway, creating `body = null`?

3) Nothing in the subsequent code checks for `body == null`.

```
//Opening the JSON file stored in Twilio Assets
let file = Runtime.getAssets()['weather.json'].open();
```

```
let jsonobj1 = JSON.parse(file);
// Parse the JSON file to Javascript object array
```

```
console.log(jsonobj1)
// Check the file has been parsed correctly on the console
```

I realized that the constant body ~~declared~~ cannot be used for comparison, so I declared ~~another~~ variable that inherits ~~from the~~ body.

```
var key = body;
//In Twilio functionFunction, the body constant-, though
inheriting from the message event.BODY-event, but cannot be
used for later usescomparison. declaring Declaring a var to
inherit body
```

The response message contains *body*, all lower case. In terms of capitalization, it would probably be best to repeat the text from the original query, as entered by the user, or from the NEA data set *area*. But responding in lower case is surely suboptimal from a human factors viewpoint.

The variable keyval is the matched forecast value; ~~and the~~ variable msg ~~will~~ determines what message to ~~be returned back~~ to WhatsApp.

```
var keyval = []
//the array to obtain the value property of the key
var msg = ""
// the response message type to be send sent back
```

~~This is a new enhancement added to the application.~~ Mum ~~has also~~ requested an option to ~~print out~~ list all available weather ~~forecast~~ forecasts. The application logic checks for the specific keyword all in the WhatsApp message, ~~this keyword will be used to determine the action to take~~. If the WhatsApp body ~~contains is~~ [contains would include substrings, incorrect for areas such as *Falls Church*] "Allall", JSON ~~stringify~~ stringifies the complete JSON array into the variable keyval.

```
//Check for a special key
if (key === "all") {
  // Print out the whole JS array
  keyval = JSON.stringify(jsonobj1.forecasts);
  // console.log('%j', keyval + " all");
}
```

Display cleanup, e.g., line breaks? – See thoughts at the end about further development.

Exception: what if key is null?

Mum also wants to know what town areas are available for weather forecasts, so this is another enhancement added to lists all of the area known keywords for Town Areas. Mum likes to know what Town Area is available for weather forecast. If the body key is contains the word Area area, the app returns all the keys areas in the JSON array and sets the variable msg with to value "0" value. "0" would be a string. The code treats msg as an integer.]

```
// check for key "area" to print out all the avail regions
else if
  (key === "area") {
    var x = "area" // ???
```

```

        var tempkeyval = {}
        for (x in jsonObj1.forecasts) {
            tempkeyval += jsonObj1.forecasts[x].area + " , "
            msg = 0 // set the variable to determine which message to send
        }
    }
    back

```

Undesirable space ahead of each comma. How do you deal with the comma at the end of the list?

```

//      console.log(tempkeyval)
    }
}

```

For all other keywords, loop through the JSON array, and match the key, and print out the value. Set the variable msg with to value "1" value.

```

// looping through the JSON Javascript object array
else
    for (var i = 0; i < jsonObj1.forecasts.length; i++){
        var obj =jsonObj1.forecasts[i]
        if (obj.area.toLowerCase() === key.toLowerCase()) { // key is
            already lower case
            //Using lowercase for matching
            //obtaining the value property of the key matched in the
            key-value array

            keyval = obj.forecast
            msg = 1
            //set the variable to determine which message to send back
        }
    }
}

```

Efficiency: should you declare obj outside the loop, to avoid having to allocate it on each iteration? Should you break out of the loop on first match, rather than continuing?

Exceptions: should you have an else clause, in case there is no match? This could also be where you deal with key == null. You could use values of msg to trigger one or more suitable error responses.

Now that we have the matched value ~~to be returned~~, we ~~will~~ return ~~back~~ the appropriate reply with corresponding ~~"value"~~ and the ~~customized proper~~ message. If ~~the~~ variable msg has ~~a~~ value ~~of~~ 0, we ~~will~~ respond ~~back~~ with the ~~customized list of areas message~~; otherwise, ~~we~~ respond ~~back~~ with ~~another the area forecast message~~.

```

    var resp1 = "The Areasareas/Regions for Weatherweather
        forecasts are: " + tempkeyval;

    var resp = "The weather in " + body + " is " +keyval;

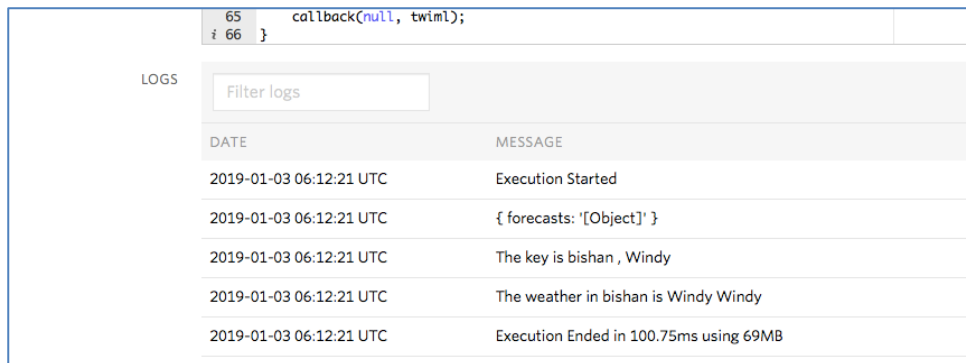
    let twiml = new Twilio.twiml.MessagingResponse();

    if (msg === 0) {
        twiml.message(resp1)
    }
    else
        twiml.message(resp);
    console.log(resp + " " + keyval)
    callback(null, twiml);

```

}

A few things to take note of: the console.log will output the message to the console below the Twilio Function console display. This can be helpful for any debugging you need to do for your code.



The screenshot shows a code editor with a function definition and a logs panel below it. The code is:

```
65 callback(null, twiml);  
66 }
```

The logs panel is titled 'LOGS' and has a 'Filter logs' input. It contains a table with the following data:

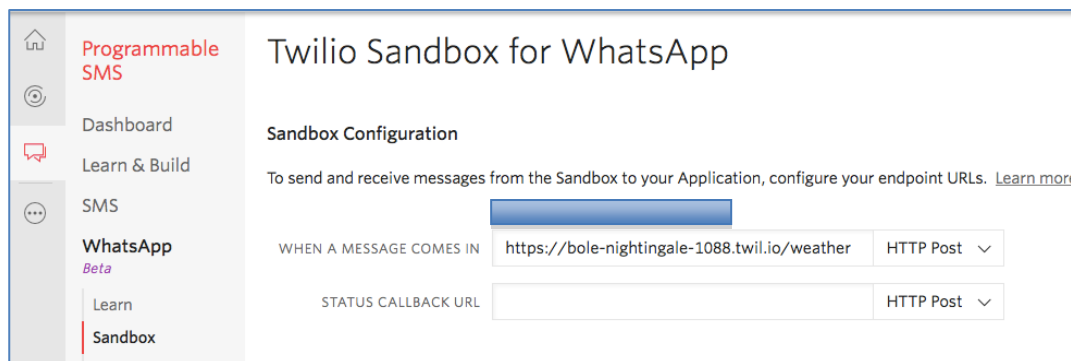
DATE	MESSAGE
2019-01-03 06:12:21 UTC	Execution Started
2019-01-03 06:12:21 UTC	{ forecasts: '[Object]' }
2019-01-03 06:12:21 UTC	The key is bishan , Windy
2019-01-03 06:12:21 UTC	The weather in bishan is Windy Windy
2019-01-03 06:12:21 UTC	Execution Ended in 100.75ms using 69MB

At the time of writing this, Twilio Function has allowed a maximum of 5 seconds runtime. If your function runs for more than 5 seconds, it will auto terminate.

### Lastly, have Twilio WhatsApp trigger the webhook to the right function

Copy the URL, the domain and the path from the Weather\_forecast Function toolbox and navigate back to the WhatsApp Sandbox. Paste the URL in the box labelled "When a message comes in". [spelling with one l or two?]

This will trigger an https POST request towards our function in the URL whenever Twilio WhatsApp receives an incoming message.



The screenshot shows the 'Twilio Sandbox for WhatsApp' configuration page. On the left is a sidebar with links: Programmable SMS, Dashboard, Learn & Build, SMS, WhatsApp Beta, Learn, and Sandbox. The main content area is titled 'Sandbox Configuration' and includes the instruction: 'To send and receive messages from the Sandbox to your Application, configure your endpoint URLs. Learn more'. There are two configuration fields:

- 'WHEN A MESSAGE COMES IN' with the value 'https://bole-nightingale-1088.twil.io/weather' and a dropdown set to 'HTTP Post'.
- 'STATUS CALLBACK URL' with an empty text box and a dropdown set to 'HTTP Post'.

[See previous comment about Word graphics](#)

## Testing the Weather Forecast Application

Now that we have the application set up, let's ~~do testing on the Application~~ test it. Remembering the application flow diagram, we can invoke the Function either by sending a direct HTTPS request or a WhatsApp message to the Twilio Sandbox. We will test both in stages. Also good to check all, area, null, and erroneous input responses.

### **Sending an HTTPS request direct to Twilio Function.**

It's good to omit periods at the end of headings, also good to be consistent with the rest of the document, which has no trailing periods on headings.

Remember our Twilio Function URL (domain and path). We will use that to invoke the function. Our application code examines the HTTP body and matches it against the JSON key-value array.

1. Sending a HTTPS POST request with JSON data with **"Bishan"** as the key A line numbered 1 requires at least one line numbered 2 somewhere further down. I believe your both structure is captured in the headings instead.

```
curl -v -X POST https://xxxx-nightinxxxx-xxxx.twil.io/weather-H "Content-Type: application/json" -d '{"Body":{"Bishan":'}
```

Trying 18.210.141.130...

```
* Connected to xxxx-nightinxxxx-xxxx.twil.io (18.210.141.130) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
* Server certificate: *.twil.io
* Server certificate: Amazon
* Server certificate: Amazon Root CA 1
* Server certificate: Starfield Services Root Certificate Authority - G2
> POST /weather HTTP/1.1
> Host: xxxx-nightinxxxx-xxxx.twil.io
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 17
>
* upload completely sent off: 17 out of 17 bytes
< HTTP/1.1 200 OK
< Date: Thu, 03 Jan 2019 06:38:21 GMT
< Content-Type: text/xml; charset=utf8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Shenanigans: none
< Strict-Transport-Security: max-age=15768000
< Server: Twilio Assets v0.1
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
<
* Connection #0 to host xxxx-nightinxxxx-xxxx.twil.io left intact
<?xml version="1.0" encoding="UTF-8"?><Response><Message>The weather in bishan is Windy</Message></Response>
```

From our Twilio Function console, you can also observe what has been sent to console log.

LOGS	
Filter logs	
DATE	MESSAGE
2019-01-03 06:38:56 UTC	Execution Started
2019-01-03 06:38:56 UTC	{ forecasts: '[Object]' }
2019-01-03 06:38:56 UTC	The key is bishan , Windy
2019-01-03 06:38:56 UTC	The weather in bishan is Windy Windy
2019-01-03 06:38:56 UTC	Execution Ended in 82.67ms using 70MB

Now that we have successfully received a response with the corresponding weather forecast, ~~lets~~let's tried try it using our WhatsApp ~~Account~~account.

### Sending a wWeather forecast request in WhatsApp

Use the device that has already been registered with Twilio Sandbox. Send the area ~~where~~whose forecast you want to query ~~the weather forecast~~. Let's use area **Bedok**.



The [object Object] display is distracting. Crop this screenshot, or create a new screenshot without it? Maybe show more than one query, perhaps one of them invalid?

## Conclusion

Twilio WhatsApp API, Function, and Assets, ~~it simplified~~simplify integrating ~~an~~ application on WhatsApp ~~messaging~~messaging and webhooking it to ~~the~~ application logic ~~running~~on ~~the~~ Runtime environment.

Further enhancement can be made to the application including fetching the API data set directly using AXIOS or fetch, and having a Key Menu wrap-around to guide ~~the~~ users.

Can you guess where ~~did~~ Twilio hosts the Functions, Assets, and TwilioCDN?

## **Further Global Comments**

### **Viewpoint**

I like your informal style; I think it will fit your target audience well. But I'm a little uncomfortable with shifts in viewpoint. Sometimes, it's "I" – I developed this app to help Mum (nice!). Sometimes, it's "you" – Set up your Twilio account as follows. Sometimes, it's "we" – We see a console display of the transaction.

It's okay to change viewpoints – not too often – if you have a reason to change, and if your text has a natural transition bridge that keeps your readers in sync.

### **Additional topics**

A few ideas about which you may or may not wish to add further text. Doubtless some of these go far beyond the scope you have in mind; think of them as checklist items. Some of these are mentioned, especially in the conclusion, but more could doubtless be said.

The paper does not describe the user view, or the app's state machine as viewed from the user's perspective. Should it? Have you wrapped a UI around this app for your Mum?

Could the user interface be extended, zooming into a map, reporting regions or city areas depending on zoom level? How about lookahead text search?

If the search string did not match exactly, for example with the wrong number of space characters, could you do an approximate match, to allow drilling down on a second pass? Indeed, for short messages such as in this app, you could just go ahead and display full results from perhaps not more than 3 or 4 approximate matches (of course with line breaks between records).

You don't include date/time information in the response message. Given that the NEA data set is static, there is no way for the user to know whether it is current. At a minimum, it would help your user if you displayed the valid period (presented as UTC in the data set, but preferably converted to local date and time for display). (You would need to include a bit more of the schema in in this document.)

Can you transparently deal with the static database problem? Neither you nor your Mum will be happy having to manually update it every day. When you resolve the question of dynamic data set updates, date/time queries will be a good enhancement. Text entry for these fields is clumsy, but graphical selection from a clock/calendar display could help your user exploit the feature.



What would you do if you wished to scale the app beyond a fairly small number of reporting points? In terms of execution, I mentioned a couple of efficiency topics above. To assist the user, map-based zooming would help.

### **Better resolution from screenshots**

Problem: Screenshots in documents lack professional appearance. They are always blurred and very often illegible. Rather than adding value to the document, they detract.

Illegibility may be caused by excessive scaling down, easily corrected – that's what I did in your screenshots above. Blurring is a more difficult issue; though hard to eliminate, crispness can at least be improved.

Before taking a screenshot, magnify the display as much as possible, while yet retaining the necessary information. Crop and scale the screenshot to fit the document. Coupled with greater screenshot scaling down, greater screen magnification improves the ratio of screen pixels to printer dots.

The examples below originated from the same page, displayed full-screen in Chrome on a 1920x1280 monitor. After capture via Windows print-screen, each image was pasted as a device-independent bitmap, then cropped, bordered, and scaled to a constant ultimate width. Notice the resolution improvement as the original magnification increases.

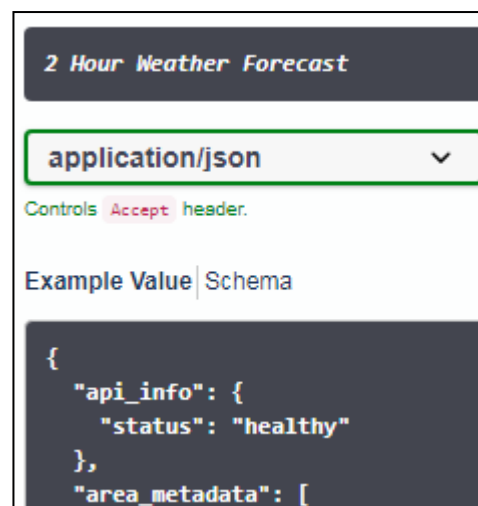


Figure 1 – 100% screenshot, 100% scale

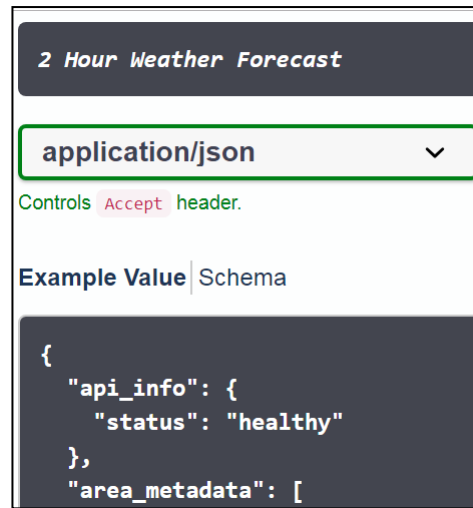


Figure 2 – 200% screenshot, 50% scale

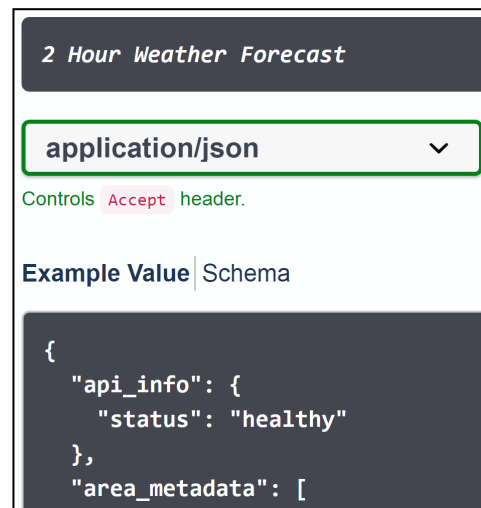


Figure 3 – 400% screenshot, 25% scale

These examples use only the tools built into Windows and Microsoft Word. Significant benefits are available from any external graphics tool (if not Photoshop, see pointers to freeware at <https://www.lifewire.com/best-free-photo-editors-for-windows-1702523>). Cropping discards extraneous information, reducing size and improving privacy.

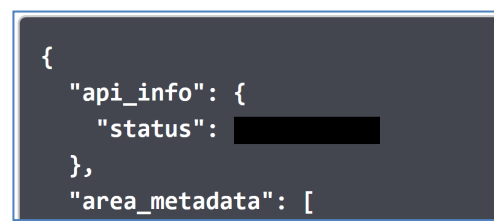


Figure 4 – 400% screenshot with redaction

Simple graphical editing can be used to highlight important areas, or, as in this document, black out some of the strings for privacy. Figure 4 shows how the value of *status* could be concealed. Privacy is perfect, as the resulting image contains nothing of the original text.