

Document Number APN0022	Revision 1.1	Prepared By NGB	Approved By TK
Title Weather Data from an API with a Senquip Device			Page 1 of 7

# WEATHER DATA FROM AN API WITH A SENQUIP DEVICE

## 1. Introduction

An application programming interface is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software. A document or standard that describes how to build or use such a connection or interface is called an API specification.

Senquip devices can request data from third party web API's from within a script.

[Visual Crossing Weather](#) offers an API that provides historical and forecast weather data. The Timeline Weather API is a simple and powerful way to retrieve weather data. You can request data over any time window including windows that span the past, present, and future. The API will take care of the combining historical observations, current 15-day forecasts, and statistical weather forecasts to create a single, consolidated dataset via a single API call. Result data is provided in a common JSON format.

This application note discusses how to request a weather forecast from the Visual Crossing Weather API.

## 2. References

The following documents were used in compiling this Application Note.

Reference	Document	Document Number
A	<a href="#">Timeline Weather API</a>	Online Weather API Documentation
B	<a href="#">Scripting Guide</a>	Request Data from HTTP API

## 3. API Implementation

The Senquip SFW002 scripting language includes an HTTP.query function that allows the user to request data from a remote server. The function, which is available as part of the api\_endpoint.js library, has the following format:

```
HTTP.query({
  url: 'https://httpbin.org/robots.txt',
  ca_cert: 'http_ca.pem',
  headers: { 'X-Foo': 'bar' }, // Optional - headers
  success: function(body, full_http_msg) { print(body); http_result = body; },
  error: function(err) { print(err); http_err = err; }, // Optional
```

The API and other required library files are loaded at the beginning of the script. A few global variables are defined to hold elements of the returned weather report.

Document Number  
APN0022

Revision  
1.1

Prepared By  
NGB

Approved By  
TK

Title  
Weather Data from an API with a Senquip Device

Page  
2 of 7

```
1 load('senquip.js');
2 load('api_endpoint.js');
3 load('api_sys.js');
4 load('api_serial.js');
5
6 let tomorrow = "";
7 let windSpd = 0;
8 let windDir = 0;
```

Figure 1 - Including Required Library Files

### 3.1. URL

In this case, the URL is the URL for the Timeline Weather API and has the following format:

```
https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/[location]/[date1]/[date2]?key=YOUR_API_KEY
```

**location** (required) – is the address, partial address or latitude, longitude location for which to retrieve weather data. Although a fixed location is used in this example, it is likely that the latitude and longitude returned by the built in GPS will be used for the location.

**date1** (optional) – is the start date for which to retrieve weather data. If a date2 value is also given, then it represents the first date for which to retrieve weather data. If no date2 is specified then weather data for a single day is retrieved, and that date is specified in date1. All dates and times are in local time of the location specified. Dates should be in the format yyyy-MM-dd. For example, 2020-10-19 for October 19<sup>th</sup>. To limit the data received from the API, we will restrict the API to delivering a single day of weather and so will only use date1.

**key** (required) is your specific API key and is available in the account settings on your Visual Crossing Weather user account. Make sure to replace YOUR\_API\_KEY with your actual API key in the examples!

Additional options are available. We have used the **\$include=days** option to force the API to send a day report rather than an hourly report.

As a starting point the following URL is used:

```
url:
'https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline
/-32.7024,152.0657/2023-04-01?key=YOUR_API_KEY&include=days'
```

### 3.2. Certificates

Although the Timeline Weather API is a secure site, an initial attempt at accessing the API without a certificate was successful and so certificates for the site were not loaded onto the Senquip device.

If a certificate is required, it can typically be downloaded from the website using the show certificates button as shown in Figure 2.

Document Number APN0022	Revision 1.1	Prepared By NGB	Approved By TK
Title Weather Data from an API with a Senquip Device			Page 3 of 7

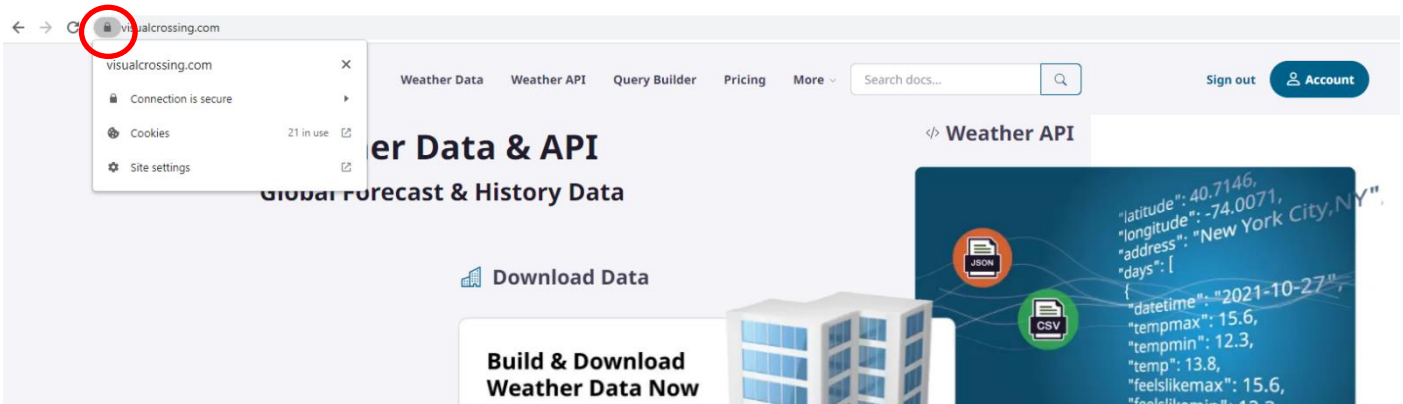


Figure 2 - Downloading Certificates

The certificate would be loaded on to the Senquip device using the *Endpoint* settings page on the Senquip Portal. Choose the *Configure HTTPS* option to load the certificates onto the Senquip device. Note that the *HTTP POST* option does not need to be enabled and the *HTTP Address* does not need to be specified. The HTTP activity will instead be performed from within a script.

HTTP

HTTP POST  Enabled

HTTP Address

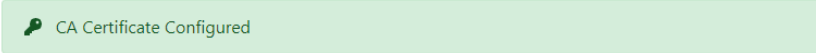


Figure 3 - Choose the Configure HTTPS Option

### 3.3. Headers

Headers are optional and are not required in this instance.

### 3.4. Return Functions

Two functions are available to process the outcome of the *HTTP.query* function. If the query returns success, a function can be written to process the body of the returned message. In this case, the body is a message formatted in JSON format. Likewise, if the function returns error, then a function can be written to process the error message.

In this implementation, the *HTTPS.query* function is called from a trigger function so that it only runs when the user requests a weather forecast. The body is parsed so that individual elements can be extracted. The *body* variable does not persist after the *success* function returns, so any data should be copied into variables such as *windSpd*, *windDir* etc.

Document Number  
APN0022

Revision  
1.1

Prepared By  
NGB

Approved By  
TK

Title  
Weather Data from an API with a Senquip Device

Page  
4 of 7

```

10 function get_weather() {
11   HTTP.query({
12     url: "https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/-32.7168,152.1869?key=<YOUR_API_KEY>&include=current",
13     success: function (body) {
14       if (body !== "") {
15         // Check if the body is a JSON string, by looking at the first character
16         if (body.at(0) !== "{".at(0)) {
17           // Not a JSON string, so it is probably a plain text error message from the API
18           // Copy the body string using slice, as the body string will not persist after this function returns
19           http_status = body.slice(0, -1);
20         } else {
21           // Parse the JSON string into an object and extract the important parameters
22           let weather = JSON.parse(body);
23           tomorrow = weather.days[0].description;
24           windSpd = weather.days[0].windspeed;
25           windDir = weather.days[0].winddir;
26           http_status = "Fetched";
27         }
28       }
29     },
30     error: function (err) {
31       http_status = err;
32     },
33   });
34 }
35
36 SQ.set_trigger_handler(function (tp) {
37   if (tp === 1) {
38     get_weather();
39   }
40 }, null);

```

Figure 4 - HTTPS.query Embedded in Trigger Function

On error, the error code is sent to the serial port for debugging purposes.

A snippet of the body returned by a successful query is shown below.

```

{
  "queryCost":1,
  "latitude":-32.7024,
  "longitude":152.0657,
  "resolvedAddress":"-32.7024,152.0657",
  "address":"-32.7024,152.0657",
  "timezone":"Australia/Sydney",
  "tzoffset":11.0,
  "days": [
    {
      "datetime":"2023-04-01",
      "datetimeEpoch":1680267600,
      "tempmax":70.4,
      "tempmin":60.3,
      "temp":65.2,
      "feelslikemax":70.4,
      "feelslikemin":60.3,
      "feelslike":65.2,
      "dew":58.2,
      "humidity":78.7,
      "precip":0.216,
      "precipprob":61.9,
      "precipcover":37.5,

```

### 3.5. Displaying the Weather Data

Weather data that was extracted from the JSON weather report in the trigger function is dispatched to the Senquip Portal using three custom parameters.

```
29 SQ.set_data_handler(function (data) {
30     SQ.dispatch(1, tomorrow);
31     SQ.dispatch(2, windSpd);
32     SQ.dispatch(3, windDir);
33
34 }, null);
```

Figure 5 - Dispatching Weather Data

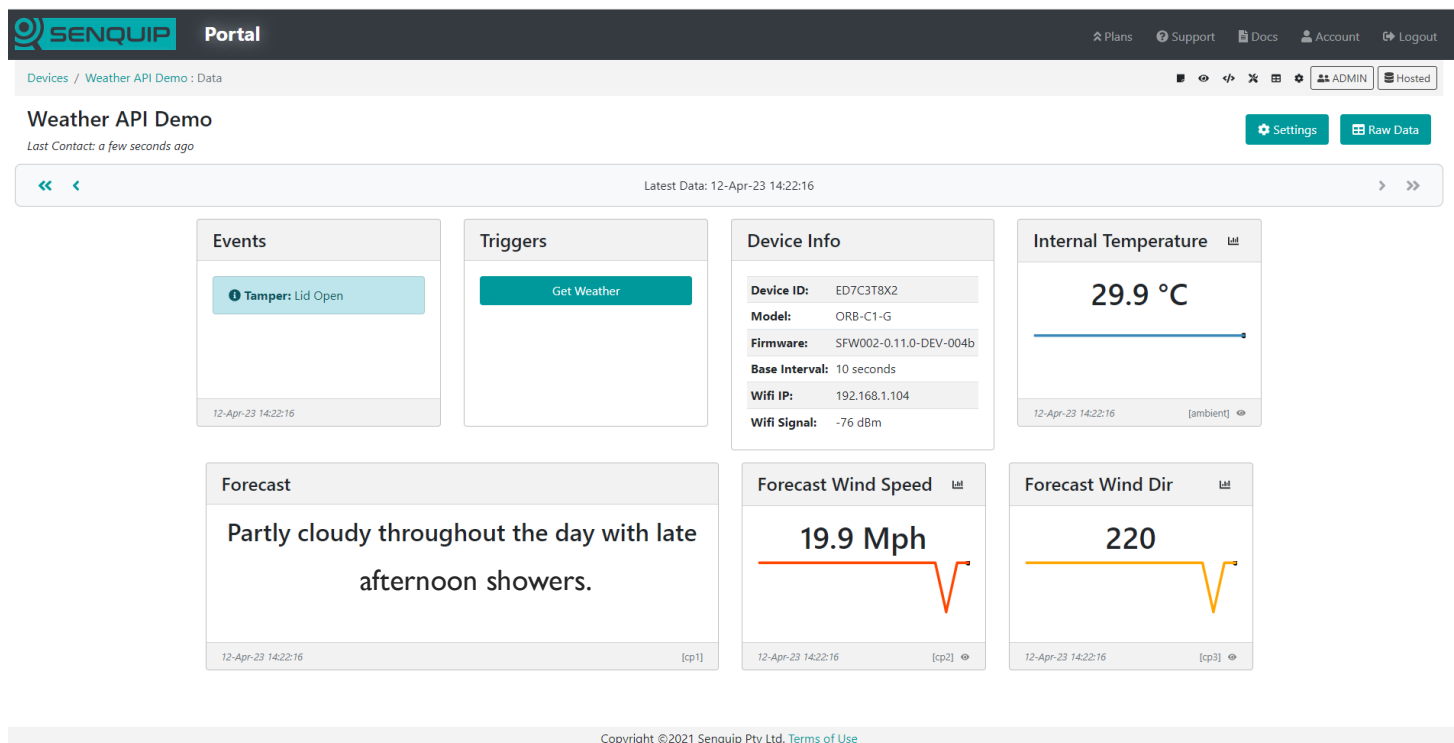


Figure 6 - Weather Report Shown on the Senquip Portal

In a real application, the script could be expanded to act based on weather. For example, a sunshield could be closed, or a pump turned on.

Document Number  
APN0022

Revision  
1.1

Prepared By  
NGB

Approved By  
TK

---

Title  
Weather Data from an API with a Senquip Device

Page  
6 of 7

## 4. Conclusion

Requesting data from an API using a script on a Senquip device is simple. Although this application note focussed on a simple weather report, some useful applications of API's include:

Energy APIs: Provide data related to energy consumption, production, and pricing.

Weather APIs: Provide real-time and forecasted weather data for a specific location.

Geolocation APIs: Provide location-based information, such address details.

Time and Date APIs: Provide accurate time and date information, including time zones, daylight saving time changes, and historical time data.

Air Quality APIs: Provide data about air pollution levels, including measurements of pollen and dust concentrations.

Astronomical APIs: Provide data related to celestial events, such as sunrise, sunset, moon phases, and astronomical events like eclipses.

Traffic APIs: Traffic APIs, such as Google Maps Traffic API, can provide real-time traffic information, including traffic congestion, incidents, and travel times.

Social Media APIs: Social media APIs, such as Twitter API and Instagram API, provide access to social media data, including posts, images, hashtags, and location-based data.

## Appendix 1: Source Code

```
load("senquip.js");
load("api_endpoint.js");
load("api_timer.js");

let tomorrow = "";
let windSpd = 0;
let windDir = 0;
let http_status = "Not Fetched";

function get_weather() {
  HTTP.query({
    url: "https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/-
32.7168,152.1869?key=<YOUR API KEY>&include=current",
    success: function (body) {
      if (body !== "") {
        // Check if the body is a JSON string, by looking at the first character
        if (body.at(0) !== "{".at(0)) {
          // Not a JSON string, so it is probably a plain text error message from the API
          // Copy the body string using slice. The body will not persist after this function returns
          http_status = body.slice(0, -1);
        } else {
          // Parse the JSON string into an object and extract the important parameters
          let weather = JSON.parse(body);
          tomorrow = weather.days[0].description;
          windSpd = weather.days[0].windspeed;
          windDir = weather.days[0].winddir;
          http_status = "Fetched";
        }
      }
    },
    error: function (err) {
      http_status = err;
    },
  });
}

SQ.set_trigger_handler(function (tp) {
  if (tp === 1) {
    get_weather();
  }
}, null);

SQ.set_data_handler(function (data) {
  SQ.dispatch(1, tomorrow);
  SQ.dispatch(2, windSpd);
  SQ.dispatch(3, windDir);
  SQ.dispatch(4, http_status);
}, null);
```