MesoWest
Great Lakes Fire & Fuels

GLFF Data API

Joe Young
10 June 2017

This covers the version of the API that exists on this date. Subsequent developments and enhancements are not addressed.

The API

The GLFF API is designed to return observation and forecast data for metadata, conventional WX, CFFWI indexes and externally-generated NFDRS indexes at selected stations within the GLFF system.

The API works like a web page, except you have to manually type in the URL, and we only return data, instead of formatting, colors, etc. This document will guide you through what URL parameters you can give the API to acquire different kinds of data. Some features are not documented, and because it is under development, some functions may not work as advertized.

Let us know in the event of either something not working correctly or something not doing what you need it to do.

API Services

We'll go into this in more detail later, but the API performs several functions, all of which are dedicated to getting data from our database archives into user applications:

- Metadata service to acquire station location, ownership and description information
- Daily timeseries for daily (18z) weather, precipitation and CFFDRS indices. Includes both observed and up to 3 days of forecast data.
- Hourly timeseries for the same, but with hourly weather information and FWIs. Includes both observed and up to 2.5 days of forecasts.
- NFDRS (daily) observed/computed values and, when available, forecasts
- REDapp daily files for a single station - can be imported into the statistics tool of REDapp
- Adjective service for deriving fire danger adjectives based on rules controlled by Fire Managers
- Info service for getting lists of metadata characteristics.

API Authentication

The API uses a standard token authentication procedure that is connected to MesoWest/SynopticLabs' core API services. To get a token to use with the API, first create an API account and key via https://synopticlabs.org/api/signup/ and then use that key with the auth service of Mesonet API (https://api.synopticlabs.org/v2/auth) to generate an unlimited number of non-expiring tokens.

Tokens both verify to us who you are, and allow you to manage your API use in whatever way you can imagine. You can learn more about our tokens here https://synopticlabs.org/api/guides/?authentication .

Accessing the API

The GLFF API can be accessed from this root URL:

https://glffapi.synopticlabs.org/v1/

If you visit this address, you'll just get an error, because you also have to give the API your token and specify which resource you want to use. Finally, each resource accepts "query string arguments" which go at the end of the path.

Query string arguments (QSA) are the bits of a url that look like &key=value. Use this guide to know which key/value pairs work, and how they change the data returned. These arguments are used to specify which stations we read data from, what dates are used and how forecasts are handled in the output.

In the end, an API request URL will look like this:

```
https://glffapi.synopticlabs.org/v1/ [Your Token]/[API Resource]/?[API Arguments]
```

API Output Format

By default all of our data are returned in a JavaScript-based data format called JSON (JavaScript Object Notation). JSON is a popular data encoding format which can be read by almost all programming languages (often with easy to acquire add ons). JSON is also relatively human-readable, but we

encourage you to install a JSON viewer plugin to your browser (http://stackoverflow.com/questions/2547769/browser-json-plugins).

In addition to JSON, many API resources support returning data as formatted CSV data. These CSVs are custom designed for the data they are returning, but should be readable by most standard tabular data managers.

The API will return data as JSON (and will return all errors as JSON!) unless you pass the output QSA like this: &output=csv.

Common Query String Arguments

Before we go into the individual resources, we want to review a few arguments that can be passed to almost any API service, and will have the same effect on the output. These include the arguments that allow you to select stations, and those that allow you to modify output. Below in the resource documentation we will indicate which of these groups of arguments are accepted.

Station Selection

The API is a powerful way to get selected stations. You can combine any of these arguments to further refine which stations you receive data from.

You can comma-separate any of these arguments to select multiple of the input value. Longitudes are negative in the western hemisphere (positive east of PM).

Query Arg	Description of use
`id`	The internal ID number of a station
`stid`	The GLFF station ID string. 3-6 characters
`state`	2-letter state abbreviation (MI, WI, MN)
`country`	2-letter country abbreviation (we have Canadian stations: CA)
`mnet`	The numeric Network identifier. This is related to MesoWest's networks list. (some quick ones 1=NWS/FAA, 2=RAWS). Use the mesonet API's networks service to decode: api.synopticlabs.org/v2/networks
`cwa`	NWS CWA/Forecast Office 3-letter identifier
`nwszone`	NWS Forecast Zone(s)
`nwsfirezone`	NWS Fire Zone Indentifier(s)
`gacc`	Single or comma-separated GACC 4-5 letter ID
`psa` `subgacc`	Single or comma-separated PSA or SubGACC ID
`radius`	Two forms: ● Lat,lon,radius [km] - get all stations within a radius of a lat/lon centroid ● GLFF stid, radius[km] - get all the stations within a distance of a given station. (Does include the given station) Can be combined with a &limit=N argument to restrict the result to the N closest stations. Cannot be comma-separated.
`bbox`	Lat/Lon bounding box; format: Upper-Lat, West-Lon, Lower-Lat, East-Lon. Cannot be comma-separated.
`active`	String value that changes how we filter stations. The following options are supported: ● "all" Return all stations irrespective of status

	• "fwi" Only return stations actively computing FWIs on this date • "inactive" Only stations that are inactive (not including QC stopped) • Unspecified/other all stations that are not listed as inactive (QC stopped stations not included)

Output Formatting

Different output formats are required for different applications. While JSON can be read natively by most programming languages, it is senseless to an application like Excel. For those applications we have an output argument that can accept several possible arguments.

Different services may place additional restrictions on non-JSON output formats.

All errors will return JSON data as their error response.

Output Argument	Description
`json`	Standard output format of nested JSON objects and lists
`csv`	Text-based comma-separated output that can be read by data analysis GUI programs. Like JSON, the formatting of the output is specific to the API resource you use.

Additional formats are expected, including special formats for RedAPP and WIMS data entry.

We welcome requests for other formats as well.

Getting Help

If you have any questions about using the API, or concerns about the data the API is providing, feel free to contact us via email at support@synopticlabs.org. We will make every effort to respond in a timely fashion, and accommodate any requests as reasonable.

GLFF API Resources

Now we will go through each API resource in a little more detail

Metadata

`/metadata/`

Serves basic station data. Accepts standard station selection arguments and output. Additionally accepts the following arguments:

Query Arg	Description of use
`attime`	YYYYMMDD-formatted date stamp to return index processing status on the given date. Index processing status is given in the status block.

Example: https://glffapi.synopticlabs.org/v1/demotoken/metadata/?stid=kmsn

Hourly Timeseries

`/hourly/`

Serves hourly WX and CFFDRS FWI values (when computed) from matching stations. In the GLFF system hourly values from stations that report more than once per hour are the last observation in that hour. NWS/FAA stations capable of reporting every 5 minutes (high frequency METAR) are used in non-HF mode, so the traditional hourly airport observations are used. Precipitation is the integration of precipitation over the hour represented.

Accepts standard station selection arguments and output formatting (json + CSV), along with the following time and function selectors

Query Arg	Description of use
`attime`	YYYYMMDDHH (UTC) timestamp, data are grabbed for that hour only. Can be “latest” to get the most recent hour.
`start`	YYYYMMDDHH (UTC) start date of a timeseries. Can be used three ways: ● Combine with &end= argument to create a time span ● Give a value of “latest” representing the current hour ● Give a comma and number (or two numbers), which represents number of hours from the start point. First value is hours BEFORE the given timestamp. Comma+second number would be hours AFTER. Examples: ○ 2016051000,24 = 05-09 00:00 -> 05-10 00:00 ○ 2016051000,-24 = 05-10 00:00 -> 05-11 00:00 ○ 2016051000,24,24 = 05-09 00:00 -> 05-11 00:00 ○ latest,12 = last 12 hours, ending on the current hour.
`end`	YYYYMMDDHH timestamp to indicate the end of a series. Alternatively you can use a value of “latest” to end your series on the most recent date.
`forecast`	String value that instructs how forecasts should be acquired. ● “future” = Grab forecasts for times within your time span that were greater than we have observations. (DEFAULT) ● “all” = Grab forecasts for the entire span - lowest leadtime forecast is used

	• Integer = Grab forecasts for this many hours after the END of the span. Limited to 48 hours. • “none” = do not acquire forecasts
`forecastonly`	Same behavior as forecast=all (forecasts grabbed for entire span) but does not grab observations. Lowest leadtime forecast is grabbed for all hours.
`shownull` `show_nulls`	Default FALSE. If you include this argument (including = true), then we will return observations with invalid CFFDRS values. Omitted, the API only returns days where CFFDRS have been computed. All stations within your station selection parameters are returned even if no actual observations are returned.
`showqc`	Default FALSE. Include this argument to include hours when the station is flagged as QCSTOP.

Hourly requests are limited to 1,000 station-days (# of stations * # of days). Larger requests will return an error.

Daily Timeseries

`/daily/`

Produces daily CFFDRS and WX values for the GLFF system. In the GLFF system daily values represent 18 UTC (12 CDT / 13 EDT) hourly value for state variables, with precipitation representing measured precipitation from 19:00Z the day before through 18:59Z the day indicated.

Accepts standard station selection arguments and output formatting (JSON + CSV), along with the following time and function selectors

Query Arg	Description of use
`attime`	YYYYMMDD timestamp, data are grabbed for that hour only. Can be “latest” to get the most recent date.
`start`	YYYYMMDD start date of a timeseries. Can be used three ways: ● Combine with &end= argument to create a time span ● Give a value of “latest” representing the current date ● Give a comma and number (or two numbers), which represents number of days from the start point. First value is days BEFORE the given timestamp. Comma+second number would be days AFTER. Examples: ○ 20160510,5 = 05-05 -> 05-10 ○ 20160510,-5 = 05-10-> 05-15 ○ 20160510,5,5 = 05-05 -> 05-15 ○ latest,12 = last 12 days, ending on the current day.
`end`	YYYYMMDD timestamp to indicate the end of a series. Alternatively you can use a value of “latest” to end your series on the most recent date.
`forecast`	String value that instructs how forecasts should be acquired. ● “future” = Grab forecasts for times within your time span that were greater than we have observations. DEFAULT ● “all” = Grab forecasts for the entire span - lowest leadtime forecast is used ● Integer = Grab forecasts for this many hours after the END of the span. Limited to 5 days. ● “none” = do not acquire forecasts
`forecastonly`	Same behavior as forecast=all (forecasts grabbed for entire span) but does not grab observations. Lowest leadtime forecast is grabbed for all days.

`leadtime`	Specify the leadtime of the forecasts grabbed, for any forecast behavior. Leadtime indicates how long before the forecasted date the forecast was published. The leadtime of any forecast is included in the returned data ● "latest": (DEFAULT) grab the smallest-leadtime forecast for the date ● "all": return every leadtime for each date ● Specific number of hours: we will grab forecasts for each matching date at the given leadtime. If no forecast made for that date at that leadtime, no forecast is returned.
`shownull` `show_nulls`	Default FALSE. If you include this argument (including = true), then we will return observations with invalid CFFDRS values. Omitted, the API only returns observations where CFFDRS have been computed. All stations within your station selection parameters are returned even if no actual observations are returned.
`showqc`	Default FALSE. Include this argument to include hours when the station is flagged as QCSTOP.
`rtma`	Boolean, specify as true and we will include a special data section which is the WX values and CFFDRS indexes for the station derived from the point in the RTMA grid. Used to compare RTMA values at points with observations.

Daily requests are limited to 100,000 station-days (# of stations * # of days). Larger requests will return an error.

NFDRS Timeseries

`/nfdrs/`

GLFF assembles and archives NFDRS observations and 1-day forecasts for every station in our network that has such forecast produced at WFAS. While we do not run the computations, we do attempt to make the daily data available in a timely manner, and include all variables supplied in the original data.

NFDRS observations and index values are derived for 2PM local time, and is published at 0 Z which corresponds to 18 CDT / 19 EDT, so early evening in the summer months. For simplification regarding the change in timezone between stations, we serve NFDRS data as only dates. As a result, the API returns timestamps is what should be considered local time, though only a date is provided. A date like 2016-05-10 would actually be the 2016-05-11 14:00 local value, but the API hides this from the user.

Accepts standard station selection arguments (stations which never have NFDRS values will still be returned, but will never contain any data) and output formatting (json + CSV), along with the following time and function selectors

Query Arg	Description of use
`attime`	YYYYMMDD timestamp, data are grabbed for that hour only. Can be “latest” to get the most recent date.
`start`	YYYYMMDD start date of a timeseries. Can be used three ways: ● Combine with &end= argument to create a time span ● Give a value of “latest” representing the current date ● Give a comma and number (or two numbers), which represents number of days from the start point. First value is days BEFORE the given timestamp.

	Comma+second number would be days AFTER. Examples: ○ 20160510,5 = 05-05 -> 05-10 ○ 20160510,-5 = 05-10-> 05-15 ○ 20160510,5,5 = 05-05 -> 05-15 ○ latest,12 = last 12 days, ending on the current day.
`end`	YYYYMMDD timestamp to indicate the end of a series. Alternatively you can use a value of “latest” to end your series on the most recent date.
`forecast`	String value that instructs how forecasts should be acquired. ● “future” = Grab forecasts for times within your time span that were greater than we have observations. DEFAULT ● “all” = Grab forecasts for the entire span - lowest leadtime forecast is used ● Integer = Grab forecasts for this many hours after the END of the span. Limited to 5 days. ● “none” = do not acquire forecasts
`forecastonly`	Same behavior as forecast=all (forecasts grabbed for entire span) but does not grab observations. Lowest leadtime forecast is grabbed for all days.
`leadtime`	Only one forecast is produced for NFDRS data, and is given a boolean leadtime of 1. Should in the future there be more than one forecast for a given NFDRS date, then this argument will function as above. Do not use.

REDapp Service

`/redapp/`

The REDapp service produces a formatted CSV output that REDapp is able to import as a weather stream, and produce FBP calculations for the entire day.

Since REDapp has a limit of around one day's worth of data, we only return a single day, with dates indicated in local time when specified

Query Arg	Description of use
`stid`	Required, a GLFF station ID from which the observations will be extracted. Considerations are not made for whether there are valid observations for a station, however STIDs that do not exist in the GLFF system will return an error.
`date`	Required. YYYYMMDD format date which represents the 00:00 to 23:00 (local) date that will be requested. For the required daily FWIs, they are requested up to their valid hour, and then the next day's daily FWI (DC, DMC and BUI) values will be displayed.
`offset`	Optional. An integer (positive or negative) span of hours representing the UTC offset of your location. The output file cannot indicate this time offset, however the values will correctly reflect this timezone.

Adjective Service

/adjective/

This service computes fire danger adjectives based on the rules and settings maintained by GLFF fire weather managers. There are custom rules for each state, and for certain groups of stations within each state. An API user can pass a date and location rule, and receive the determined fire weather adjectives for that location.

Query Arg	**Description of use**
`date`	Optional. An MMDD date string which is used to determine

	which time-of-year rules are applied. If omitted, today's date is used.
`state`	Optional. Rules are defined at the first order by state. State rules cannot be customized by fire managers, so specifying a state means you will be applying the state default adjective rule set. If omitted, a station ID must be passed.
`stid`	Optional. A single station id, which will define the rules applied when evaluating your given characteristics. A station first applies its state defaults, and then, if they exist, is overruled by updates inserted by fire managers for that state. If omitted, a state must be provided.
`ffmc`	Optional. Value of FFMC index. 0 if omitted.
`dmc`	Optional. Value of DMC index. 0 if omitted.
`isi`	Optional. Value of ISI. 0 if omitted.
`bui`	Optional. Value of BUI. 0 if omitted.
`fwi`	Optional. Value of FWI. 0 if omitted.
`situations`	JSON-encoded list of objects/dictionaries that provide multiple instances of the same set of characteristics described above. This is valuable for determining the fire danger adjectives for multiple scenarios at once. It is recommended that this argument be passed as a POST variable. Example: `&situations=[{"date":"0505","state":"WI"},{"date":"0504","stid":"kmsn","isi":20,"uid":"d"},{"date":"0505","sta`

	`te":"WI"}]`
`uid`	Optional. A string that can be given to be used as the index in the returned JSON object. This is particularly useful when requesting multiple situations, as you may not have tracked the order you inserted the situations.

Information Service

/info/

The info service is a quick utility for accessing general metadata information that cannot be easily accessed via other resources or SynopticLabs APIs. This service only returns JSON data.

Query Arg	**Description of use**
`list`	Single or comma-separated list of variables we will describe, each in a unique JSON node in the data response. The following arguments are supported: ● state ● mnet ● mnetshortname - String identifier of a mesonet ● country ● cwa ● nwszone ● nwsfirezone ● gacc ● psa or subgacc

	All of which correspond to metadata and station selection criteria shown above

JSON Output Overview

All non-error API requests have two elements in the JSON response:

- A stations object containing a list of objects that are the stations returned


```
"stations": [
  {
    "stid": "KMSN",
    "name": "Madison, Dane County Regional-Truax Field",
    "stninfo_id": 5049,
    "mnet_id": "1",
    "county": "Dane",
    "state": "WI",
    "country": "US",
    "wims_id": null,
    "description": null,
```


- A summary object giving some information about the request


```
"summary": {
    "code": 200,
    "message": "Success",
    "objects": 1,
    "proctime_ms": 74,
    "other": false
}
```


The summary set may have different information depending on the request, but will always have a code in the 200s for a successful request. For example, a code 201 means no stations were found for the request.

The data provided by GLFF are to be used by trained professionals for advisory purposes only. Data provided via this API and other GLFF web displays are freely available for public consumption but we do not advocate the use of any of these data without prior understanding and training regarding fire weather and how to use fire weather indexes.