FIS-B Rest

What is FIS-B Rest?

FIS-B Rest is the REST Server for FIS-B Decode.

FIS-B Decode is a system that collects FIS-B messages– storing and managing them in a Mongo database. FIS-B Rest is the REST Server that retrieves data from the database and returns JSON objects.

Flask is used as the web server.

For this tutorial we will be using Flask directly. When deployed by systemd (see the end of this document), gunicorn will be used as the web server.

Getting things running

The first item is to get FIS-B Decode operational. Once you have done that, you may proceed to installing FIS-B Rest.

Currently, both FIS-B Decode and FIS-B Rest must reside on the same machine. The only reason for this is that images are not stored in the database. If this ever changes, there is no longer any restriction requiring both to run from the same machine.

For this discussion, we will assume that you cloned FIS-B Rest into the `'fisb-rest'` directory underneath your home directory. We will also assume that all of the prerequisites, such as `pip3`, have been installed as part of the 'fisb-decode' installation.

For the following, change your directory (`cd`) to wherever you installed `fisb-rest` (usually `~/fisb-rest`).

Install Flask and friends:

```
pip3 install -r misc/requirements.txt
```

You may need to add `~/.local/bin` to your PATH (pip3 will point this out to you in the above step):

```
PATH=$PATH:~/.local/bin
export PATH
```

You will also want to put this into your `~/.bashrc` file.

If you want to be able to build the documentation, run the following:

```
pip3 install -r misc/requirements-sphinx.txt
```

Building the documentation is exactly like 'fisb-decode':

```
cd docs
./makedocs
cd ..
```

Next step is to make a symbolic link between Harvest's image files (usually located in `<somewhere>/fisb-decode/runtime/web-images`). Assuming you installed 'fisb-decode' under your home directory, the command would be:

```
ln -s ~/fisb-decode/runtime/web-images static/png
```

The `static` subdirectory is where Flask stores its static contents. At this point, this is only the image files and an `index.html` placeholder.

There is a configuration file in `fisb-rest` called `fisb_restConfig.py`. It should look something like:

```
#: Where to write error messages.
ERROR_FILENAME = 'FISB_REST.ERR'

#: MONGO URI
MONGO_URI = 'mongodb://localhost:27017/'
```

Make sure that `MONGO_URI` is set to the same value as you are using for 'fisb-decode'.

All that is left is to run Flask:

```
flask run
```

And you are done! Almost. You probably want to install something like postman to send requests and see the results.

The documentation contains details on all the server requests, as well as details on the returned JSON objects. The `fisb-rest/docs` directory has a PDF of the primary user guide material in the file `fisb-rest-intro.pdf`. To see the complete documentation, build the docs with `'./makedocs'` from the `docs` subdirectory.

The rest of this document describes important background information and details of the REST request and JSON results structure.

Static and Non-Static Requests

Almost all of the REST server calls are *non-static*. That means that they return active, timed data from FIS-B. A few calls are *static*. These do not return active FIS-B data, but other fixed results, such as information about the colors and text to display as legend data for images. You can easily tell the difference between the two, because all static calls start with '/static/', such as '/static/legend'. They both use the basic return structure described below, but static calls do not return an 'after' field (described shortly), since it has no meaning for them.

Basic Return Structure

The only JSON field guaranteed to be in each returned object is `"status"`. It will have one of two values: '0' if there were no errors, and '-1', if there were any errors. Note that this does not include regular web errors like 404 errors. Just errors where the query was close enough that a JSON object was returned.

If you do get an `"status"` code of '-1', you will also get an `"error"` field which contains an error message.

For example:

```
http://127.0.0.1:5000/taf/kind?lat=80
```

would return:

```
{
  "status": -1
  "error": "Need both lat and long parameters.",
}
```

If the `"status"` code is '0', you will get an object that contains `"num_results"` and `"after"` fields.

For example:

```
http://127.0.0.1:5000/taf/kingtut
```

returns:

```
{
  "status": 0
  "num_results": 0,
  "after": "2004-01-01T00:00:00Z",
}
```

`"num_results"` is the number of results. If there were any results, we would return a `"result"` or `"results"` field depending on the query. We will cover this more in a minute.

The use of the "after" field is critical to understand. Each non-error (non-static) FIS-B data query will return an `"after"` field. You can use this value in a query string to return only those results which occurred *after* the `"after"` time.

As an example, to get the most recent METAR for KIND you would send:

```
http://127.0.0.1:5000/metar/kind
```

and it might return:

```
{
  "status": 0
  "num_results": 1,
  "after": "2021-06-23T02:03:41.221000Z",
  "result": {
      "type": "METAR",
      "unique_name": "KIND",
      "contents": "METAR KIND 230154Z 20006KT 10SM FEW065 FEW250 17/08 A3003
                   RMK AO2 SLP168\n     T01720083=",
      "expiration_time": "2021-06-23T03:54:00Z",
      "observation_time": "2021-06-23T01:54:00Z"
  },
}
```

If we wanted to get the most recent KIND METAR we can just repeat the query. But if we wanted a result only if a new METAR has been posted, we would add the `after=` query string:

```
http://127.0.0.1:5000/metar/kind?after=2021-06-23T02:03:41.221000Z
```

If there is no new result, we will get:

```
{
  "status": 0
  "num_results": 0,
  "after": "2021-06-23T02:03:41.221000Z",
}
```

Once a new result arrives, we will get the full METAR.

A couple of points about the `after=` parameter:

- **Always use an 'after' value you got from FIS-B Rest**. Don't make them up. This ensures you will always get the latest result.
- If you get a new result, it means something changed from the old result. Maybe a new image, maybe a NOTAM that didn't have any text changes, but was resent and has a new expiration time.

Result and Results

There are two basic types of REST queries: those that return a single result, and those that return more than a single result. Queries that can at most return a single result will return the result as an object in the `"result"` field. If the query *might* return more than one result, it will return a list of objects in the `"results"` field. Note the difference in name. If a query that can return more than one result only returns one result, it will still return a `"results"` field with a list containing only one item in it.

You can tell which calls return a single result and which return multiple results by looking at the call definition. If the definition has **'(M)'** at the beginning, it will return multiple objects using the `"results"` field. It it starts with **'(1)'**, it will at most return a single item using the `"result"` field.

See the METAR example above as an example of a query only returning one item. If we sent:

```
http://127.0.0.1:5000/metar
```

We would get back a `"results"` field with a list of many results.

```
{
  "status": 0
  "num_results": 549,
  "after": "2021-06-23T03:09:30.380000Z",
  "results": [
      {
          "type": "METAR",
          "unique_name": "KMWK",
          "contents": "METAR KMWK 230135Z AUTO 00000KT 10SM CLR 18/14
                       A3003 RMK AO2\n     T01820144=",
          "expiration_time": "2021-06-23T03:35:00Z",
          "observation_time": "2021-06-23T01:35:00Z"
      },
      {
          "type": "METAR",
          "unique_name": "KOSH",
          "contents": "METAR KOSH 230153Z 00000KT 10SM BKN070 17/10 A2993=",
```

```
            "expiration_time": "2021-06-23T03:53:00Z",
            "observation_time": "2021-06-23T01:53:00Z"
        }

        << many results removed >>

    ],
}
```

Query Strings

Query strings appear after a question mark (‘`?`’) in a request and have a name, an equal sign (‘`=`’), and are followed with a value. Multiple query strings are separated by ampersand (‘`&`’) characters.

In FIS-B Rest, query parameters will modify the request in some way. Most query parameters only affect a small portion of requests. The ‘after=’ and ‘limit=’ query strings can be used with almost all queries. It will be noted in the documentation if the other query strings are useful for a particular REST call.

after=

Will return results that were created after this value. This value should be obtained **ONLY** from the `"after"` field of a returned JSON object. This field applies to all non-static REST queries.

Form:

```
after=<value from 'after' field from returned JSON object>
```

Example:

```
http://127.0.0.1:5000/metar?after=2021-06-23T22:21:43.282000Z
```

high=, low=

Will return objects only if they are between two altitude limits given in feet (inclusive). Only applies to objects that have a graphic component. They must always occur together, must be positive integers, and low must be <= high.

Typically, this applies to G-AIRMET, SIGMET/WST, AIRMET, NOTAM-TRA, NOTAM-TMOA, and NOTAM-TFR. It does not apply to NOTAM-D-SUA (for complicated reasons discussed when we describe this type of object).

Warning: Some TWGO (Text with Graphic Overlays) objects will get a text segment before the graphic portion arrives. So the query will not catch the altitude limits. Since the object could not possibly meet criteria (see next paragraph), it will be returned.

These query strings will not filter out any objects to which they do not apply. So if you do a query on METARs, or TWGO objects that don’t have any altitude information, the selected objects will be returned.

Form:

```
low=<low altitude value>&high=<high altitude value>
```

Example:

```
http://127.0.0.1:5000/g-airmet?low=12000&high=17999
```

lat=, lon=

If a latitude and a longitude is provided, AND the selected object is a polygon or a set of polygons, the object will be returned only if the latitude and longitude are within the polygon. You must supply both a latitude and longitude (as integer or floating point values), and they must have valid values (latitude -90 to 90, longitude -180 to 180).

These query strings will not filter out any objects to which they do not apply. So if you do a query on METARs, or TWGO objects that are not polygons, the selected objects will be returned.

Form:

```
lat=<latitude>&lon=<longitude>
```

Example:

```
http://127.0.0.1:5000/notam-d-sua?lat=40.1234&lon=-86.1234
```

limit=

Will limit the number of items returned to the specified amount. This only makes sense for those queries that may return more than one object (i.e. the definition contains '(M)'). The number must be an integer >= 1. There is a default limit of 10,000 for all queries (more than you will ever need). If you specify a value higher than this, it will be reduced to 10,000.

Form:

```
limit=<maximum objects to return>
```

Example:

```
http://127.0.0.1:5000/all?limit=500
```

FISB Object Principles

We will next discuss the individual REST directives and the results they return. Different objects have different fields depending on their type, but all objects have a number of fields in common. We will discuss those here and not mention them again.

Again, there are two types of REST requests, those that are FIS-B related, and those that are static. The fields mentioned below are only FIS-B related.

`"expiration_time"`

Time the message should expire in ISO-8601 UTC. FISB Rest will not send an update when an object expires. It is up to you to delete expired objects. All objects will have this field.

`"type"`

Basic type of message. These are items like `METAR`, `TAF`, `NOTAM`, `SIGMET`, `G-AIRMET`, etc. The type of a message dictates the fields that it will have. All objects will have this field.

`"unique_name"`

This is a unique identifier within a particular 'type'. If you combine the 'type' and 'unique_name' strings you will get a primary key valid across all FISB objects. Internally, FISB Rest combines the 'type' and 'unique_name' fields with a dash to get a primary key. All objects will have this field.

`"geojson"`

All graphical objects other than images (which are ‘png’ files) will have a ‘geojson’ field. This is in standard geojson format. **ALL** geojson objects have at their outer layer a `FeatureCollection` with a `features` list. The `features` list will have one or more geojson `Feature` objects. This even includes object types like METARs that will only have one `Feature`. The reason behind this is to make vector object processing more uniform.

Polygon and Point objects are common. G-AIRMET can produce both Polygons and LineStrings. So can PIREPs (almost all PIREPs are point objects, but you can have a ‘route’ PIREP which will be rendered as a LineSting). Each `FeatureCollection` will only have one type of geometry (Point, Polygon, or LineString).

Also note that some objects can have more than one geometry. For example, a NOTAM-TFR may have multiple geometries with different active times.

The geojson `"properties"` field will vary dependent on the ‘type’ of object. These will be documented for each individual object type. There are a number of geojson `"properties"` fields that are common enough to be discussed now.

`"altitudes"`

List of 4 items: Highest altitude, highest altitude type (MSL or AGL), lowest altitude, and lowest altitude type (MSL or AGL). Except for NOTAM-TMOA and NOTAM-TRA, both altitude types will be the same.

`"start_time"`

Start time of the activity. This may be different than any time mentioned in the encompassing object. May not have an accompanying `"stop_time"`.

`"stop_time"`

Stop time of the activity. This may be different than any time mentioned in the encompassing object. May not have an accompanying `"start_time"`.

A common scenario that occurs is in NOTAM-TFRs. Imagine a VIP is travelling to a city, then going to a convention center to give a speech, and then traveling back to the airport. A NOTAM-TFR will be issued with three geographies: one each (with identical coordinates) for arrival and departure at the airport, and one for the convention center. Each will have different start and stop times, and the altitudes for the convention center speech might be different than the airport altitudes.

An example of the `"geojson"` field (with multiple geometries) and the others described above is:


```
{
  "type": "NOTAM",
  "subtype": "TFR",
  "unique_name": "1-3325"
  "number": "1/3325",
  "contents": "NOTAM-TFR 1/3325 081500Z OH..AIRSPACE VANDALIA, OH..
               TEMPORARY FLIGHT RESTRICTION. PURSUANT TO 14 CFR SECTION
               91.145, MANAGEMENT OF ACFT OPS IN THE VICINITY OF AERIAL
               DEMONSTRATIONS AND MAJOR SPORTING EVENTS, ACFT OPS ARE
               PROHIBITED WI AN AREA DEFINED AS 5NM RADIUS OF
               395428N0841316W (DQN130010.4) SFC-16000FT EFFECTIVE:
               2107081500 UTC UNTIL 2107082030 UTC, 2107091400 UTC UNTIL
               2107092100 UTC, 2107101600 UTC UNTIL 2107102100 UTC, AND
               2107111600 UTC UNTIL 2107112100 UTC. DUE TO USAF
               THUNDERBIRDS AND HIGH SPEED AERIAL DEMONSTRATIONS AT THE
```

```
            CENTERPOINT ENERGY DAYTON AIRSHOW. UNLESS AUTH BY
            ATC-DAYTON TOWER 119.9. JIM TUCCIARONE, TEL 216-390-8410,
            IS THE POINT OF CONTACT. THE DAYTON /DAY/ ATCT, TEL
            937-415-6750, FREQ 119.9 IS THE CDN FAC.
            2107081500-2107112100",
"start_of_activity_time": "2021-07-08T15:00:00Z",
"end_of_validity_time": "2021-07-11T21:00:00Z",
"expiration_time": "2021-07-11T21:00:00Z",
"geojson": {
    "features": [
        {
            "geometry": {
                "coordinates": [
                    [-84.219818, 39.991235], [-84.198666, 39.989631],
                    [-84.178329, 39.984879], [-84.159593, 39.977165],
                    [-84.143177, 39.966783], [-84.129715, 39.954136],
                    [-84.119722, 39.93970],  [-84.11358, 39.924059],
                    [-84.111525, 39.907786], [-84.113631, 39.891518],
                    [-84.119814, 39.875878], [-84.129836, 39.861468],
                    [-84.143309, 39.84884],  [-84.159714, 39.838477],
                    [-84.178422, 39.830779], [-84.198716, 39.826038],
                    [-84.219818, 39.824438], [-84.24092,  39.826038],
                    [-84.261214, 39.830779], [-84.279922, 39.838477],
                    [-84.296327, 39.84884],  [-84.3098, 39.861468],
                    [-84.319822, 39.875878], [-84.326005, 39.891518],
                    [-84.328111, 39.907786], [-84.326056, 39.924059],
                    [-84.319914, 39.939709], [-84.309921, 39.954136],
                    [-84.296459, 39.966783], [-84.280043, 39.977165],
                    [-84.261307, 39.984879], [-84.24097, 39.989631]
                ],
                "type": "Polygon"
            },
            "properties": {
                "altitudes": [
                    16000,
                    "MSL",
                    0,
                    "MSL"
                ],
                "element": "TFR",
                "id": "1-3325",
                "start_time": "2021-07-08T15:00:00Z",
                "stop_time": "2021-07-08T20:30:00Z"
            },
            "type": "Feature"
        },
        {
            "geometry": {
                "coordinates": [
                    [-84.219818, 39.991235], [-84.198666, 39.989631],
                    [-84.178329, 39.984879], [-84.159593, 39.977165],
                    [-84.143177, 39.966783], [-84.129715, 39.954136],
                    [-84.119722, 39.93970],  [-84.11358, 39.924059],
                    [-84.111525, 39.907786], [-84.113631, 39.891518],
                    [-84.119814, 39.875878], [-84.129836, 39.861468],
                    [-84.143309, 39.84884],  [-84.159714, 39.838477],
                    [-84.178422, 39.830779], [-84.198716, 39.826038],
                    [-84.219818, 39.824438], [-84.24092,  39.826038],
                    [-84.261214, 39.830779], [-84.279922, 39.838477],
                    [-84.296327, 39.84884],  [-84.3098, 39.861468],
```

```
                    [-84.319822, 39.875878], [-84.326005, 39.891518],
                    [-84.328111, 39.907786], [-84.326056, 39.924059],
                    [-84.319914, 39.939709], [-84.309921, 39.954136],
                    [-84.296459, 39.966783], [-84.280043, 39.977165],
                    [-84.261307, 39.984879], [-84.24097, 39.989631]
                ],
                "type": "Polygon"
            },
            "properties": {
                "altitudes": [
                    16000,
                    "MSL",
                    0,
                    "MSL"
                ],
                "element": "TFR",
                "id": "1-3325",
                "start_time": "2021-07-09T14:00:00Z",
                "stop_time": "2021-07-09T21:00:00Z"
            },
            "type": "Feature"
        },
        {
            "geometry": {
                "coordinates": [
                    [-84.219818, 39.991235], [-84.198666, 39.989631],
                    [-84.178329, 39.984879], [-84.159593, 39.977165],
                    [-84.143177, 39.966783], [-84.129715, 39.954136],
                    [-84.119722, 39.93970],  [-84.11358, 39.924059],
                    [-84.111525, 39.907786], [-84.113631, 39.891518],
                    [-84.119814, 39.875878], [-84.129836, 39.861468],
                    [-84.143309, 39.84884],  [-84.159714, 39.838477],
                    [-84.178422, 39.830779], [-84.198716, 39.826038],
                    [-84.219818, 39.824438], [-84.24092,  39.826038],
                    [-84.261214, 39.830779], [-84.279922, 39.838477],
                    [-84.296327, 39.84884],  [-84.3098, 39.861468],
                    [-84.319822, 39.875878], [-84.326005, 39.891518],
                    [-84.328111, 39.907786], [-84.326056, 39.924059],
                    [-84.319914, 39.939709], [-84.309921, 39.954136],
                    [-84.296459, 39.966783], [-84.280043, 39.977165],
                    [-84.261307, 39.984879], [-84.24097, 39.989631]
                ],
                "type": "Polygon"
            },
            "properties": {
                "altitudes": [
                    16000,
                    "MSL",
                    0,
                    "MSL"
                ],
                "element": "TFR",
                "id": "1-3325",
                "start_time": "2021-07-10T16:00:00Z",
                "stop_time": "2021-07-10T21:00:00Z"
            },
            "type": "Feature"
        },
        {
            "geometry": {
```

```
                    "coordinates": [
                        [-84.219818, 39.991235], [-84.198666, 39.989631],
                        [-84.178329, 39.984879], [-84.159593, 39.977165],
                        [-84.143177, 39.966783], [-84.129715, 39.954136],
                        [-84.119722, 39.93970],  [-84.11358, 39.924059],
                        [-84.111525, 39.907786], [-84.113631, 39.891518],
                        [-84.119814, 39.875878], [-84.129836, 39.861468],
                        [-84.143309, 39.84884],  [-84.159714, 39.838477],
                        [-84.178422, 39.830779], [-84.198716, 39.826038],
                        [-84.219818, 39.824438], [-84.24092,  39.826038],
                        [-84.261214, 39.830779], [-84.279922, 39.838477],
                        [-84.296327, 39.84884],  [-84.3098, 39.861468],
                        [-84.319822, 39.875878], [-84.326005, 39.891518],
                        [-84.328111, 39.907786], [-84.326056, 39.924059],
                        [-84.319914, 39.939709], [-84.309921, 39.954136],
                        [-84.296459, 39.966783], [-84.280043, 39.977165],
                        [-84.261307, 39.984879], [-84.24097, 39.989631]
                    ],
                    "type": "Polygon"
                },
                "properties": {
                    "altitudes": [
                        16000,
                        "MSL",
                        0,
                        "MSL"
                    ],
                    "element": "TFR",
                    "id": "1-3325",
                    "start_time": "2021-07-11T16:00:00Z",
                    "stop_time": "2021-07-11T21:00:00Z"
                },
                "type": "Feature"
            }
        ],
        "type": "FeatureCollection"
    },
}
```


`"cancel"`

This field **only** applies to TWGO (Text With Graphic Overlay) objects. This includes ‘type’ field values of:

- `NOTAM` (all subtypes)
- `FIS_B_UNAVAILABLE` (FIS-B Product Unavailable)
- `AIRMET`
- `SIGMET` (includes WST (Convective Sigmet))
- `CWA` (Center Weather Advisory),
- `G_AIRMET`

If this field is present in a message, the message must be cancelled. It is only present in messages being cancelled. In practice, I have only seen messages cancelled for NOTAMS, G-AIRMETS, and CWAs. But the standard states all TWGO messages are fair game.

The value of the `"cancel"` field is just the `"unique_name"` field. You should immediately delete the message of the specified ‘type’ and ‘unique_name’ from your database.

Whenever you get one of the TWGO `"type"` fields, the first thing you should do is to check the object for a `"cancel"` field. If you find one, cancel the message (which might not even exist in your records), and do no further processing on the message. All the other fields are not important.

Here are a couple of examples of messages with the `"cancel"` field present. A G-AIRMET cancellation:

```
{
    "type": "G_AIRMET",
    "unique_name": "21-9897",
    "cancel": "21-9897",
    "expiration_time": "2021-06-21T17:10:21Z"
}
```

And a NOTAM cancellation:

```
{
    "type": "NOTAM",
    "unique_name": "21-12860",
    "cancel": "21-12860",
    "expiration_time": "2021-06-21T17:23:18Z"
}
```

Note that the NOTAM won't have a `"subtype"` field. It isn't needed. The `"unique_id"` is sufficient and will work across all NOTAM subtypes.

`"station"`

Some objects, such as CRL and RSR objects, are dependent on a particular ground station. The best identifier for the station is its latitude and longitude. The value of the `"station"` field is the latitude and longitude combined with a tilde character such as `'40.0383~-86.255593'`. One advantage of this scheme is that the standard in some cases requires you to show the latitude and longitude of all ground stations being received, and you can un-parse the ground station id to get this information.

REST API and Message Descriptions

All items

```
(M) /all
```

Will return all current reports. This is essentially a dump of the database.

The way this is typically used is to perform an `/all` at the start, then use use the `"after"` field to get periodic updates. If you don't want to get all results at once, you can use the 'after=' and 'limit=' query parameters together.

METARs

```
(M) /metar
(1) /metar/<4 character id>
```

Return all METAR reports or a single METAR report.

Fields:

`"observation_time"`
: Time the observation was made.

Example:


```
{
      "type": "METAR",
      "unique_name": "KLAF",
      "observation_time": "2021-06-24T16:54:00Z",
      "contents": "METAR KLAF 241654Z VRB06G17KT 10SM CLR 28/16
                   A3004 RMK AO2 SLP168\n     T02780161=",
      "expiration_time": "2021-06-24T18:54:00Z",
      "geojson": {
          "features": [
              {
                  "geometry": {
                      "coordinates": [-86.9475, 40.4125],
                      "type": "Point"
                  },
                  "properties": {
                      "id": "KLAF",
                      "name": "KLAF"
                  },
                  "type": "Feature"
              }
          ],
          "type": "FeatureCollection"
      }
}
```


Notes:

- Will have a `"geojson"` field if configured for locations. This will always be a 'Point'.
- The expiration time is typically 2 hours after the observation time.

TAFs

```
(M) /taf
(1) /taf/<4 character id>
```

Return all Terminal Area Forecast (TAF) reports, or a single report.

Fields:

`"issued_time"`
: Time the forecast was issued by NWS.

`"valid_period_begin_time"`
: Starting time of the forecast.

`"valid_period_end_time"`
: Ending time of the forecast. This is also the expiration time.

Example:

```
{
  "type": "TAF",
  "unique_name": "KIND",
  "issued_time": "2021-06-24T11:20:00Z",
  "valid_period_begin_time": "2021-06-24T12:00:00Z",
  "valid_period_end_time": "2021-06-25T18:00:00Z",
  "contents": "TAF KIND 241120Z 2412/2518 16007KT P6SM FEW200\n
               FM241900 19012G20KT P6SM SCT250\n
               FM250600 18010KT P6SM VCSH OVC100\n
               FM251500 19014G22KT P6SM VCSH OVC045\n
               FM251700 20014G23KT P6SM VCSH OVC028=",
  "expiration_time": "2021-06-25T18:00:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [-86.2816, 39.72518],
                  "type": "Point"
              },
              "properties": {
                  "id": "KIND",
                  "name": "KIND"
              },
              "type": "Feature"
          }
      ],
      "type": "FeatureCollection"
  }
}
```


Notes:

- Will have a `"geojson"` field if configured for locations. This will always be a ‘Point’.

Winds Aloft Forecasts

```
(M) /wind-06
(1) /wind-06/<3 character id>
(M) /wind-12
(1) /wind-12/<3 character id>
(M) /wind-24
(1) /wind-24/<3 character id>
```

Return winds aloft forecast for all stations or a single station. Winds aloft forecasts are issued 6, 12, and 24 hours in advance. Wind forecasts use a 3 character id, rather than 4.

Fields:

`"model_run_time"`
: Time the winds aloft model was run to generate the report.

`"issued_time"`”
: When the report was issued.

`"valid_time`”
: Time at which the forecast is designed to model. This is a single point in time.

`"for_use_from_time"`
: Starting time the forecast can be used.

`"for_use_to_time"`
: Time the forecast should no longer be used. This is also the expiration time.

Example:


```
{
  "type": "WINDS_12_HR",
  "unique_name": "CMH",
  "model_run_time": "2021-06-24T12:00:00Z",
  "issued_time": "2021-06-24T13:58:00Z",
  "valid_time": "2021-06-25T00:00:00Z",
  "for_use_from_time": "2021-06-24T21:00:00Z",
  "for_use_to_time": "2021-06-25T06:00:00Z",
  "contents": "   1919 2122+13 2712+11 9900+04 2606-09 3109-19
                292735 312945 315757",
  "expiration_time": "2021-06-25T06:00:00Z"
}
```


Notes:

- Will have a ‘Point’ `"geojson"` field if configured for location.
- The header is not provided since there are multiple options for display. A typical header could look like:

  ```
  3000    6000    9000   12000   18000   24000  30000  34000  39000
  1919 2219+17 2217+12 2208+04 3012-09 2819-20 281435 363145 317257
  ```

PIREPs

```
(M) /pirep
```

Returns all available PIREPs. Will have a `"geojson"` field if configured for location (and the location can be decoded). This is most commonly a ‘Point’, but in the case of a route, may also be a LineString.

Fields:

`"station"`
: Nearest weather reporting location.

`"report_type"`
: Either `UA` for normal PIREP or `UUA` for urgent.

`"report_time"`
: Time the report was made. There are two ways FIS-B Decode can be configured. The way the standard suggests is to just keep the report active until an hour or so after it is last transmitted. This can result in PIREPs hanging around for 4 hours or more. It can also be configured to delete the PIREP so many minutes after the report time (2 hours is a good value). This is the preferred method.

Example of a PIREP that is a Point:

```
{
  "type": "PIREP",
  "unique_name": "djfHdke8mQ2Z",
  "contents": "PIREP MSN 241940Z MSN UA /OV MSN080020/TM 1940/FL220/TP
               E545/TA M15/IC LGT RIME DURD 220-180",
  "expiration_time": "2021-06-24T21:40:00Z",
  "fl": "220",
  "ic": "LGT RIME DURD 220-180",
  "ov": "MSN080020",
  "report_time": "2021-06-24T19:40:00Z",
  "report_type": "UA",
  "station": "MSN",
  "ta": "M15",
  "tm": "1940",
  "tp": "E545",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [-88.895286, 43.218243],
                  "type": "Point"
              },
              "properties": {
              },
              "type": "Feature"
          }
      ],
      "type": "FeatureCollection"
    }
}
```


Example of a PIREP that is a route with a geojson type of LineString:


```
{
  "type": "PIREP",
  "unique_name": "KQeZQflpleq1",
  "ov": "ACO090020-ACO310010",
  "report_time": "2021-06-25T10:32:00Z",
  "report_type": "UA",
  "station": "AKR",
  "tb": "LGT-MOD 350-390",
  "tm": "1032",
  "tp": "NMRS",
  "fl": "350",
  "contents": "PIREP ACO 251032Z AKR UA /OV ACO090020-ACO310010
               /TM 1032/FL350/TP NMRS/TB LGT-MOD 350-390",
  "expiration_time": "2021-06-25T12:32:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [
                      [-80.765163, 41.156786],
                      [-81.38991, 41.194716]
                  ],
                  "type": "LineString"
              },
              "properties": {
                  "id": "KQeZQflpleq1"
```

```
            },
            "type": "Feature"
        }
    ],
    "type": "FeatureCollection"
  }
}
```

Notes:

- While FIS-B Decode can parse about 90-95% of all locations, it can not parse them all. PIREPs (especially by tower controllers) do not always follow a set format, since they can be hand entered.
- The identifier immediately after 'PIREP' ('PIREP MSN' or 'PIREP ACO' in our examples) is totally made-up garbage by the FIS-B creator. **Do not use it**. The `"station"` field is from the FAA and is safe to use.
- The report is parsed into its basic fields. If a field name is not in the report, it will not be listed. These are:
    - `"ov"`: Location of the PIREP.
    - `"tm"`: Time the PIREP activity occurred or was reported.
    - `"fl"`: Flight level.
    - `"tp"`: Type of aircraft.
    - `"tb"`: Turbulence report.
    - `"sk"`: Sky conditions.
    - `"rm"`: Remarks.
    - `"wx"`: Flight visibility and flight weather.
    - `"ta"`: Temperature.
    - `"wv"`: Wind direction and speed.
    - `"ic"`: Icing report.

SIGMET/WST, AIRMET, CWA

```
(M) /sigmet
(M) /airmet
(M) /cwa
```

Provides all available SIGMET/WSTs, AIRMETs, and CWAs (Center Weather Advisories). From a returned object perspective, they are all identical except for their subject matter. SIGMET includes WSTs (Convective SIGMETs).

One important thing to remember is that all of these objects can have both a text and graphics portion (that are sent separately by FIS-B). Only the text portion is mandatory. Per the standard, if a text portion is received, it is immediately sent out. If a graphic portion arrives, it is combined with the text portion and both are sent out as a single report. If a graphic portion never gets a matching text portion, it is never sent out.

In the example below, the only difference if this was only a text only AIRMET would be that the `"geojson"` field would be missing.

Fields:

`"issued_time`"
: When the report was issued.

`"for_use_from_time"`
: Starting time the forecast can be used.

`"for_use_to_time"`
: Time the forecast should no longer be used. This is also the expiration time.

`"subtype"`
: This only applies to ‘type’ SIGMET. The value of subtype will be one of:

  - `SIGMET`
  - `WST`

Example:


```
{
  "type": "AIRMET",
  "unique_name": "21-9178",
  "issued_time": "2021-06-24T20:31:00Z",
  "for_use_from_time": "2021-06-24T20:45:00Z",
  "for_use_to_time": "2021-06-25T03:00:00Z",
  "contents": "AIRMET KBOS 242031 BOST WA 242045\nAIRMET TANGO UPDT
              3 FOR TURB VALID UNTIL 250300\nAIRMET TURB...ME NH VT
              MA RI CT NY LO NJ PA OH LE WV MD DC DE VA\nNC SC GA FL
              AND CSTL WTRS\nFROM 80NW PQI TO CON TO 80ESE SIE TO
              30ENE ILM TO 20W CTY TO\n130ESE LEV TO 40W CEW TO 50SW
              PZD TO GQO TO HMV TO HNN TO CVG TO\nFWA TO 30SE ECK TO
              YOW TO YSC TO 80NW PQI\nMOD TURB BTN FL270 AND FL430.
              CONDS CONTG BYD 03Z THRU 09Z.",
  "expiration_time": "2021-06-25T03:00:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [
                      [-69.494019, 47.707443], [-71.575241, 43.219528],
                      [-73.22525, 38.574371],  [-77.313538, 34.541016],
                      [-83.431549, 29.597855], [-87.830887, 28.326874],
                      [-87.454605, 30.823517], [-84.979935, 31.063843],
                      [-85.152969, 34.961243], [-82.128983, 36.436844],
                      [-82.025986, 38.753586], [-84.70253, 39.015884],
                      [-85.187988, 40.979004], [-82.235413, 42.900925],
                      [-75.896301, 45.441513], [-71.690598, 45.43808],
                      [-69.494019, 47.707443]
                    ],
                  "type": "Polygon"
              },
              "properties": {
                  "altitudes": [
                      43000,
                      "MSL",
                      27000,
                      "MSL"
                  ],
                  "id": "21-9178",
                  "start_time": "2021-06-24T20:45:00Z",
                  "stop_time": "2021-06-25T03:00:00Z"
              },
              "type": "Feature"
```

```
            }
        ],
        "type": "FeatureCollection"
    }
}
```

Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- **lat=** and **lon=** are valid query strings. If present, only those results which contain the supplied point will be returned.
- **high=** and **low=** are valid query strings. If present, only those results that fall within a certain altitude range will be returned.

G-AIRMET

```
(M) /g-airmet
(M) /g-airmet-00
(M) /g-airmet-03
(M) /g-airmet-06
```

Return all G-AIRMETS. The 00, 03, and 06 variants will only return G-AIRMETs of that type.

Fields:

`"subtype"`
: One of 0, 3, or 6, dependent if this is a 00, 03, or 06 hour G-AIRMET. '/g-airmet' will select all of these. '/g-airmet-00', '/g-airmet-03', and '/g-airmet-06' will only select a particular type.

`"issued_time"`"
: When the report was issued.

`"for_use_from_time"`
: Starting time the forecast can be used.

`"for_use_to_time"`
: Time the forecast should no longer be used. This is also the expiration time.

Example:

```
{
  "type": "G_AIRMET",
  "unique_name": "21-10892",
  "subtype": 0,
  "issued_time": "2021-06-25T02:45:00Z",
  "for_use_from_time": "2021-06-25T03:00:00Z",
  "for_use_to_time": "2021-06-25T06:00:00Z",
  "expiration_time": "2021-06-25T06:00:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [
                      [-84.529495, 46.609497], [-86.84967, 45.799942],
```

```
                    [-87.399673, 44.399872], [-84.859772, 43.919907],
                    [-82.389908, 45.259552], [-84.529495, 46.609497]
                ],
                "type": "Polygon"
            },
            "properties": {
                "altitudes": [
                    2000,
                    "AGL",
                    0,
                    "AGL"
                ],
                "element": "LLWS",
                "id": "21-10892",
                "start_time": "2021-06-25T03:00:00Z",
                "stop_time": "2021-06-25T06:00:00Z"
            },
            "type": "Feature"
        }
    ],
    "type": "FeatureCollection"
  }
}
```


Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- There is only a single graphical entry for each G-AIRMET.
- Most G-AIRMETs return Polygons, but freezing level G-AIRMETs may return a Polygon or LineString.
- The `"properties"` geojson field may contain the following fields:

  `"conditions"`

  If the reason for the G-AIRMET is IFR or Mountain Obscuration conditions, this field will list the conditions responsible. This will be a list with one or more of the following elements:

  - `'UNSPCFD'`: Unspecified
  - `'ASH'`: Ash
  - `'DUST'`: Dust
  - `'CLOUDS'`: Clouds
  - `'BLSNOW'`: Blowing snow
  - `'SMOKE'`: Smoke
  - `'HAZE'`: Haze
  - `'FOG'`: Fog
  - `'MIST'`: Mist
  - `'PCPN'`: Precipitation

  `"element"`

  Single string present for each G-AIRMET which describes the reason it was issued. These will be one of:

  - `'TURB'`: Turbulence
  - `'LLWS'`: Low level wind shear

    - `'SFC'`: Strong surface winds
    - `'ICING'`: Icing
    - `'FRZLVL'`: Freezing Level
    - `'IFR'`: IFR conditions
    - `'MTN'`: Mountain Obscuration

- **lat=** and **lon=** are valid query strings. If present, only those results which contain the supplied point will be returned.

- **high=** and **low=** are valid query strings. If present, only those results that fall within a certain altitude range will be returned.

NOTAM (in general)

```
(M) /notam
(M) /notam/<4 character id>
```

Returns all NOTAMs of all types. If an id is specified, will find all NOTAMs associated with that id (i.e. the `"location"` field inside a NOTAM). Not all NOTAMs have a location.

No examples will be given for this section. See the more detailed types of NOTAMs for examples.

There are basically two types of FIS-B NOTAMs. NOTAM-TFRs and all the rest. NOTAM-TFRs in FIS-B are repackaged by the FIS-B creator and have differences with the other NOTAMs in terms of format.

I further divide NOTAM-Ds into two types: regular NOTAM-Ds and NOTAM-D-SUAs. The NOTAM-D-SUAs are different, because they can have an optional geojson field that is not produced by FIS-B, but loaded as an auxillary file (if configured by 'fisb-decode'). They also have some unique characteristics which must be considered.

Fields common to all NOTAMs:

`"subtype"`
: The type of NOTAM. Will be one of:

    - `"TFR"`: NOTAM-TFR
    - `"D"`: NOTAM-D
    - `"D-SUA"`: NOTAM-D with SUA information.
    - `"FDC"`: NOTAM-FDC
    - `"TRA"`: NOTAM-TRA
    - `"TMOA"`: NOTAM-TMOA

`"number"`
: This is the 'official' number of the NOTAM. It is what should be shown to users. Do not use the `"unique_name"`.

`"start_of_activity_time"`
: Directly parsed from the NOTAM. The FIS-B standard doesn't use this time. It will instead use the `"start_time"` field in the object, but this applies only to objects with a graphics portion. 'fisb-decode' can use the parsed time directly, but it depends on the configuration. Please see the fisb-decode documentation.

`"end_of_validity_time"`

Directly parsed from the NOTAM. The FIS-B standard doesn't use this time. It will instead use the `"stop_time"` field in the object, but this applies only to objects with a graphics portion. 'fisb-decode' can use the parsed time directly, but it depends on the configuration. Please see the fisb-decode documentation.

Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.

NOTAM-TFR

```
(M) /notam-tfr
```

NOTAM-TFRs may or may not be associated with a geojson object. If they are, the object may have multiple geometries.

NOTAM-TFRs are truncated after a certain number of characters and will end with the text `'(INCMPL)'`.

Example:


```
{
  "type": "NOTAM",
  "unique_name": "0-5116",
  "subtype": "TFR",
  "number": "0/5116",
  "contents": "NOTAM-TFR 0/5116 220551Z PART 1 OF 4 SECURITY..SPECIAL
              SECURITY INSTRUCTIONS FOR UNMANNED AIRCRAFT SYSTEM (UAS)
              OPERATIONS FOR MULTIPLE LOCATIONS NATIONWIDE. THIS NOTAM
              REPLACES NOTAM FDC 9/7752 TO PROVIDE UPDATED INSTRUCTIONS.
              PURSUANT TO 49 U.S.C. SECTION 40103(B)(3), THE FAA CLASSIFIES
              THE AIRSPACE DEFINED IN THIS NOTAM AND IN FURTHER DETAIL AT
              THE FAA WEBSITE IDENTIFIED BELOW AS 'NATIONAL DEFENSE
              AIRSPACE'. OPERATORS WHO DO NOT COMPLY WITH THE FOLLOWING
              PROCEDURES MAY FACE THE FOLLOWING ENFORCEMENT ACTIONS: THE
              UNITED STATES GOVERNMENT MAY PURSUE CRIMINAL CHARGES,
              INCLUDING CHARGES UNDER 49 U.S.C. SECTION 46307; AND THE
              FAA MAY TAKE ADMINISTRATIVE ACTION, INCLUDING IMPOSING
              CIVIL PENALTIES AND REVOKING FAA CERTIFICATES OR
              AUTHORIZATIONS TO OPERATE UNDER TITLE 49 U.S.C. SECTIONS
              44709 AND 46301. IN ADDITION, PURSUANT TO 10 U.S.C. SECTION
              130I, 50 U.S.C. SECTION 2661, AND 6 U.S.C. SECTION 124N, THE
              DEPARTMENT OF DEFENSE (DOD), DEPARTMENT OF ENERGY (DOE),
              DEPARTMENT OF HOMELAND SECURITY (DHS), OR DEPARTMENT OF
              JUSTICE (DOJ), RESPECTIVELY , MAY TAKE SECURITY ACTION AT OR
              IN THE VICINITY OF SPECIFIC LOCATIONS PRE-COORDINATED WITH
              THE FAA WITHIN A SUBSET OF THE DEFINED AIRSPACE, OR IN
              RESTRICTED OR PROHIBITED AIRSPACE ADJACENT TO SUCH LOCATIONS,
              THAT RESULTS IN THE INTERFERENCE, DISRUPTION, SEIZURE,
              2009011200-2109011159 END PART 1 OF 4 !FDC 0/5116 FDC PART
              2 OF 4 SECURITY..SPECIAL SECURITY INSTRUCTIONS FOR UNMANNED
              DAMAGING, OR DESTRUCTION OF UNMANNED AIRCRAFT CONSIDER(INCMPL)",
  "start_of_activity_time": "2020-09-01T12:00:00Z",
  "end_of_validity_time": "2021-09-01T11:59:00Z",
  "expiration_time": "2021-06-26T02:24:57Z"
  }
```

Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- **lat=** and **lon=** are valid query strings. If present, only those results which contain the supplied point will be returned.
- **high=** and **low=** are valid query strings. If present, only those results that fall within a certain altitude range will be returned.

NOTAM-D and NOTAM-FDC

```
(M) /notam-d
(M) /notam-d/<4 character id>
(M) /notam-fdc
(M) /notam-fdc/<4 character id>
```

NOTAM-D (Distant) and NOTAM-FDC (Flight Data Center) have identical formats other than the subtype. Both may have geojson 'Point' objects.

These objects (as well as the NOTAM-D-SUA, NOTAM-TMOA, and NOTAM-TRA objects to be described later) will have the following fields:

`"accountable"`
: Accountable entity.

`"affected"`
: Facility (usually airfield) primarily affected.

`"keyword"`
: Notam type. For non-NOTAM-FDCs, one of:

  - RWY: Runway
  - TWY: Taxiway
  - APRON: Apron/Ramp
  - AD: Aerodrome
  - OBST: Obstruction
  - NAV: Navigation Aid
  - COM: Communications
  - SVC: Services
  - AIRSPACE: Airspace
  - OPD: Obstacle Departure Procedure
  - SID: Standard Instrument Departure
  - STAR: Standard Terminal Arrival

  For NOTAM-FDCs, one of:

  - CHART: Chart
  - DATA: Data
  - IAP: Instrument Approach Procedure
  - VFP: Visual Flight Procedures
  - ROUTE: Route
  - SPECIAL: Special

- SECURITY: Security
- U: Unverified Aeronautical information
- O: Other

Example NOTAM-D:


```
{
  "type": "NOTAM",
  "unique_name": "21-12579-KHFY",
  "keyword": "OBST",
  "location": "KHFY",
  "number": "06/579",
  "start_of_activity_time": "2021-06-24T17:47:00Z",
  "accountable": "HUF",
  "affected": "HFY",
  "contents": "!HUF 06/579 HFY OBST TOWER LGT (ASR 1002451)
               393252.90N0861537.20W (9.3NM WSW HFY) 1042.7FT (294.9FT AGL)
               U/S 2106241747-2109220400",
  "end_of_validity_time": "2021-09-22T04:00:00Z",
  "expiration_time": "2021-09-22T04:00:00Z",
  "geojson": {
        "features": [
          {
              "geometry": {
                  "coordinates": [-86.087494, 39.626999],
                  "type": "Point"
              },
              "properties": {
                  "airport_id": "KHFY",
                  "altitudes": [0, "AGL", 0, "AGL"],
                  "id": "21-12579-KHFY",
                  "start_time": "2021-06-24T17:47:00Z",
                  "stop_time": "2021-09-22T04:00:00Z"
              },
              "type": "Feature"
          }
      ],
      "type": "FeatureCollection"
  }
}
```


Example NOTAM-FDC:


```
{
  "type": "NOTAM",
  "unique_name": "1-8239",
  "subtype": "FDC",
  "keyword": "IAP",
  "location": "KSBN",
  "number": "1/8239",
  "accountable": "FDC",
  "affected": "SBN",
  "contents": "!FDC 1/8239 SBN IAP SOUTH BEND INTL, SOUTH BEND, IN.\n
               RNAV (GPS) RWY 9R, AMDT 1A...\nLPV DA NA ALL CATS AND
               LNAV/VNAV DA NA ALL CATS.\n2106071415-2306071415EST",
  "start_of_activity_time": "2021-06-07T14:15:00Z",
  "end_of_validity_time": "2023-06-07T14:15:00Z",
```

```
  "expiration_time": "2023-06-07T14:15:00Z"
}
```

Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- Since airports are located on the ground, the `"altitudes"` list will always show 0 AGL for high and low altitudes.
- These NOTAMs will have only ‘Point’ geojson objects, if they have any at all.

NOTAM-D-SUA

```
(M) /notam-d-sua
(M) /notam-d-sua/<4 character ARTCC location>
```

NOTAM-D SUAs are used to activate special use airspace that, in addition to regularly active times, also have the provision ‘other times by NOTAM’.

Each NOTAM-D SUA will declare an airspace it applies to, and at what altitudes it is valid for. The start and stop times are also given.

Most special use airspaces have a single geographical region. But many do not. They may have up to ten or so geographic regions that all go under a single name. And each geographic area may have different times and different altitudes associated with it. But the NOTAM will only refer to the entire SUA and will only provide one set of altitudes. To be on the safe side, we assume the NOTAM applies to all geographic portions and will store the altitudes, but not treat them as the whole truth.

The NOTAM-D SUA never provides a graphic portion, but it is possible to load the database with SUA graphic information. See the ‘fisb-decode’ documentation for more about this.

Also note that while the NOTAM usually indicates a well-known area that is documented in SUA definition files, the area may not be in the file (aerial refueling is common), or sometimes they make something up like ‘RANDOM AIR REFUELING FKL-THS’.

Fields unique to NOTAM-D SUA:

`"airspace"`
: This is the special use airspace official name.

`"altitude_text"`
: A text string that defines the altitudes to be used. See the caution about this under the `"altitudes"` item below.

`"altitudes"`
: This is in the exact same form as the altitudes field for other objects, but it isn’t quite the same. The first item is the high altitude, followed by the high altitude type, then the low altitude and low altitude type (just like all the other `"altitudes"` fields).

  However, remember that special use airspaces may have multiple geographic areas, each with their own altitudes. We really can’t be sure how that applies in a given NOTAM-D SUA. That is why, unlike other

`"altitudes"` fields, this one is placed at the top level of the object and not inside a `"geojson"` field. A good display program would point this out when displaying any FAA SUA altitude information.

Also, it is not always clear what altitudes are AGL or MSL. Flight levels are considered MSL, `SFC` is considered AGL, but the usual reference is just feet (`'FT'`). If we can't determine AGL or MSL, we are forced to use what they tell us, and that is `'FT'`.

Example:


```
{
  "type": "NOTAM",
  "unique_name": "21-12582-KZKC",
  "subtype": "D-SUA",
  "keyword": "AIRSPACE",
  "location": "KZKC",
  "number": "06/582",
  "accountable": "SUAC",
  "affected": "ZKC",
  "airspace": "R4501F",
  "altitude_text": "SFC-3200FT",
  "altitudes": [
      3200,
      "FT",
      0,
      "AGL"
  ],
  "start_of_activity_time": "2021-06-25T23:00:00Z",
  "end_of_validity_time": "2021-06-26T05:00:00Z",
  "contents": "!SUAC 06/582 ZKC AIRSPACE R4501F ACT SFC-3200FT
               2106252300-2106260500",
  "expiration_time": "2021-06-26T05:00:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [
                      [-92.151396, 37.68334],  [-92.181396, 37.68334],
                      [-92.203062, 37.717229], [-92.146118, 37.719451],
                      [-92.151396, 37.68334]
                  ],
                  "type": "Polygon"
              },
              "properties": {
                  "name": "R-4501F",
                  "remarks": "EXCLUDES R-4501A, R-4501B, AND R-4501C
                              WHEN ACTIVE",
                  "times_of_use": "0700 - 1800, DAILY; OTHER TIMES BY
                                   NOTAM 24 HOURS IN ADVANCE",
                  "type": "R"
              },
              "type": "Feature"
          }
      ],
      "type": "FeatureCollection"
  }
}
```


Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- If the NOTAM-D SUA has a geojson field, the `"properties"` field will contain the following fields which are taken from the source data file:
  - `"name"`: Name of the SUA. Note that the FAA in the NOTAM will take out dashes. This name will contain them. This is probably the better display name for users.
  - `"remarks"`: Remarks taken directly from the data file.
  - `"times_of_use"`: Times of use taken directly from the data source. This is a mix of hard to tell apart local and UTC times.
  - `"type"`: Single letter code type. `'R'` for Restricted, `'W'` for Warning, `'A'` for Alert, etc.
- The `"accountable"` field will be (for CONUS) `SUA` followed by an `'E'`, `'C'`, or `'W'` for East, Central, or West.
- The `"location"` field will be (for CONUS) the three letter code for an ARTCC with a 'K' at the front (i.e. `KZID`).
- **lat=** and **lon=** are valid query strings. If present, only those results which contain the supplied point will be returned.

NOTAM-TMOA, NOTAM-TRA

```
(M) /notam-tmoa
(M) /notam-tmoa/<4 letter SUAx location>
(M) /notam-tra
(M) /notam-tra/<4 letter SUAx location>
```

These NOTAMs are basically NOTAM-D-SUAs, but the NOTAM itself provides the geometry.

The location for TMOA and TRA NOTAMs are the SUA sites (SUAE, SUAC, SUAW) as opposed to D-SUA NOTAMs which use ARTCC site names (KZID, etc).

Example:


```
{
  "type": "NOTAM",
  "unique_name": "8-142",
  "subtype": "TMOA",
  "keyword": "AIRSPACE",
  "location": "SUAC",
  "number": "08/142",
  "accountable": "SUAC",
  "affected": "ZOB",
  "airspace": "STINGER TEMPORARY MOA",
  "altitude_text": "6000FT UP TO BUT NOT INCLUDING FL180",
  "contents": "!SUAC 08/142 ZOB AIRSPACE STINGER TEMPORARY MOA ACT 6000FT UP
               TO BUT NOT INCLUDING FL180 2106281700-2106291800",
  "start_of_activity_time": "2021-06-28T17:00:00Z",
  "end_of_validity_time": "2021-06-29T18:00:00Z",
  "expiration_time": "2021-06-29T18:00:00Z",
  "geojson": {
      "features": [
          {
              "geometry": {
                  "coordinates": [
                      [-84.246597, 41.191864], [-84.37294,  40.870514],
                      [-83.937607, 40.810776], [-83.823624, 41.176758],
```

```
                    [-84.246597, 41.191864]
                ],
                "type": "Polygon"
            },
            "properties": {
                "airport_id": "KZOB",
                "altitudes": [18000, "MSL", 6000, "MSL"],
                "id": "8-142",
                "start_time": "2021-06-28T17:00:00Z",
                "stop_time": "2021-06-29T18:00:00Z"
            },
            "type": "Feature"
        }
    ],
    "type": "FeatureCollection"
  }
}
```


Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required.
- These will not have a top-level `"altitudes"` field. The `"altitudes"` field will be inside the geojson object(s). The only altitude types will be AGL and MSL (they may be mixed).
- **lat=** and **lon=** are valid query strings. If present, only those results which contain the supplied point will be returned.
- **high=** and **low=** are valid query strings. If present, only those results that fall within a certain altitude range will be returned.

FIS-B Unavailable

```
(M) /fis-b-unavailable
```

Returns FIS-B Unavailable reports. Each report will be in a separate object. Per the standard, these must be made available to the pilot. Experience has shown that most of these are triggered by some long absence of the actual data, so you will probably notice the missing data long before FIS-B tells you about it. It will send these messages for Guam, San Juan, Alaska, and Hawaii, even if you are in the continental U.S.

Fields:

`"product"`
: Short coded text description of the unavailable product. Will be one of:

  - ALASKA NEXRAD
  - CLOUD TOPS
  - CWA
  - D-NOTAM
  - FDC-NOTAM
  - G-AIRMET
  - GUAM NEXRAD
  - HAWAII NEXRAD
  - ICING
  - LIGHTNING

- METAR
- NEXRAD IMAGERY
- NOTAM-D-CANCEL
- NOTAM-FDC-CANCEL
- PIREP ICING
- PIREP TURBULENCE
- PIREP URGENT
- PIREP WIND SHEAR
- ROUTINE PIREP
- SAN JUAN NEXRAD
- SIGMET/CONVECTIVE SIGMET
- SUA
- TAF
- TFR NOTAM
- TRA-NOTAM/TMOA-NOTAM
- TURBULENCE
- WINDS AND TEMPERATURE ALOFT

`"centers"`
: Locations affected by the outage.

Example:

```
{
  "type": "FIS_B_UNAVAILABLE",
  "unique_name": "21-10582",
  "product": "GUAM NEXRAD",
  "contents": "GUAM NEXRAD PRODUCT UPDATES UNAVAILABLE",
  "expiration_time": "2021-06-25T09:13:59Z",
  "issued_time": "2021-06-22T07:56:00Z",
  "centers": [
      "ZAB", "ZAU", "ZFW", "ZHU", "ZID", "ZKC", "ZMP", "ZOB"
  ]
}
```

Another example:

```
{
  "type": "FIS_B_UNAVAILABLE",
  "unique_name": "21-10589",
  "product": "ICING",
  "contents": "ICING PRODUCT UPDATES UNAVAILABLE AT 16000FT AND
               18000FT AND 24000FT",
  "expiration_time": "2021-06-27T05:14:15Z",
  "issued_time": "2021-06-27T04:17:00Z",
  "centers": [
      "ZAB", "ZAU", "ZFW", "ZHU", "ZID", "ZKC", "ZMP", "ZOB"
  ]
}
```

Notes:

- `"cancel"`: Present only when cancelled. Always check for this first and delete the report. No other processing required. This is specified in the standard, but I have never seen one of these cancelled. They

are left to expire.

Cancelled Messages

```
(M) /cancel
(M) /cancel-notam
(M) /cancel-g-airmet
(M) /cancel-cwa
(M) /cancel-sigmet
(M) /cancel-airmet
```

Will show recently cancelled messages. Note that these are messages specifically cancelled by FIS-B before their usual expiration time. '/cancel' will show all recently cancelled messages. More specific types will only show those message types. Only TWGO messages can be cancelled. '/cancel/sigmet' will include SIGMET and WST messages.

See the earlier discussion about the meaning of the `"cancel"` field.

Example:

```
{
  "type": "NOTAM",
  "unique_name": "21-12120",
  "cancel": "21-12120",
  "expiration_time": "2021-06-27T10:16:31Z"
}
```

Images

```
(M) /image
(1) /image/<image-name>
```

Return image metadata and link(s) to image file(s). The following are the available image names:

NEXRAD-REGIONAL
: NEXRAD regional radar image.

NEXRAD-CONUS
: NEXRAD CONUS radar image.

CLOUD-TOPS
: Cloud Tops image.

LIGHTNING
: Lightning image. This will produce two images in the `"urls"` field. One called LIGHTNING_ALL for all lightning and the other called LIGHTNING_POS containing only positive lightning strikes.

ICING-02000 through **ICING-24000**
: Icing image for each altitude. Icing produces three images in the `"urls"` field: ICING_xxxxx_PRB, ICING_xxxxx_SEV, and ICING_xxxxx_SLD for icing probability, severity, and super large droplet probability, respectively.

TURBULENCE-02000 through **TURBULENCE-24000**

Turbulence image for each altitude.

Fields with particular meaning for images:

`"bbox"`
: Bounding box. Used by slippy map javascript libraries to place the 'png' file in the image. The contents is a list containing 2 elements, each a two item list. The first element is the coordinate of the NW corner of the image, and the second is the coordinates of the SE corner of the image. Unlike almost all other FIS-B Rest coordinates, these are in latitude first, then longitude order.

`"valid_time"`
: Valid time of a forecast image in ISO-8601 format. Images that are forecasts will have a `"valid_time"` field, and images that are observations will have an `"observation_time"` field. Only one or the other will appear.

`"observation_time"`
: Observation time of an observation in ISO-8601 format. Only radar images and lightning are observations. All other images are forecasts.

  All of the observation images can have a 10 minute variance. That means newer observations can blend with older observations until the oldest observation is 10 minutes or older than the newest image. Then its data must disappear. The oldest data will be considered the observation time.

  Why this is useful is often seen in NEXRAD REGIONAL radar. When you get a new image, you overwrite all the old data. But if some of the new data is missing, but present in an older image, the older data will remain and fill-in the newer missing data.

  In practice, this really only applies to NEXRAD regional radar and lightning images. NEXRAD CONUS is sent every 15 minutes, so it doesn't apply. Lighting is sent every 5 minutes, so a single image can have data from 2 images. NEXRAD regional can have data from 5 images since it is sent every 2 minutes. This is why, in poor reception conditions, the NEXRAD regional data will appear perfect, the lightning image will have an error or two, and the NEXRAD CONUS image will be a total mess.

`"urls"`
: This is an dictionary whose key is the name of an image, and whose value is a URL pointing to that image. Icing and lightning images can have multiple images per product. All others only have a single image.

  You may note that the name of the '.png' file associated with an image is a set of random characters that changes for each image update. This is so there is no chance of an image changing during the time an image object is in use. These images will be deleted after their maximum possible life (135 minutes).

Example of an image with a single link:


```
{
  "type": "IMAGE",
  "unique_name": "NEXRAD_REGIONAL",
  "observation_time": "2021-06-26T08:16:00Z",
  "bbox": [
      [43.333333, -90.4],
      [36.666667, -81.6]
  ],
  "expiration_time": "2021-06-26T09:31:00Z",
  "urls": {
      "NEXRAD_REGIONAL": "http://127.0.0.1:5000/png/qXQisU08RoXp.png"
```

```
  }
}
```

Example of an image with multiple links:

```
{
  "type": "IMAGE",
  "unique_name": "ICING_10000",
  "valid_time": "2021-06-26T07:00:00Z"
  "bbox": [
      [43.333333, -92.0],
      [36.666667, -80.0]
  ],
  "expiration_time": "2021-06-26T08:45:00Z",
  "urls": {
      "ICING_10000_PRB": "http://127.0.0.1:5000/png/niojQdPQKRWs.png",
      "ICING_10000_SEV": "http://127.0.0.1:5000/png/4luk4TWJSDd3.png",
      "ICING_10000_SLD": "http://127.0.0.1:5000/png/47JNF24Cwwzu.png"
  },
}
```

Current Report List (CRL)

```
(M) /crl-notam-tfr
(1) /crl-notam-tfr/<station>
(M) /crl-airmet
(1) /crl-airmet/<station>
(M) /crl-sigmet
(1) /crl-sigmet/<station>
(M) /crl-g-airmet
(1) /crl-g-airmet/<station>
(M) /crl-cwa
(1) /crl-cwa/<station>
(M) /crl-notam-tra
(1) /crl-notam-tra/<station>
(M) /crl-notam-tmoa
(1) /crl-notam-tmoa/<station>
```

Current Report Lists contain the reports sent by a particular ground station for a certain subset of messages. If you wish to get a CRL only for a particular ‘<station>’, you may do so. See the example for the form of a ‘<station>’.

A CRL is tied to a specific ground station. Each ground station will have its own set of CRLs. If you are receiving three ground stations, you should be getting a CRL object of each type from all three ground stations.

CRLs have the concept of ‘completeness’. If you have a copy of each message in the CRL, that CRL is said to be ‘complete’. If the CRL doesn’t show any messages for that type, the CRL is also considered complete. If there is no CRL for a particular type, that CRL is considered ‘incomplete’. If a particular CRL type has messages with both a text and a graphics portion, both must be present in order to be considered complete.

CRLs can have a maximum number of 138 reports. If there are more than 138 items, the overflow field will be set to 1. The standard calls the state where the overflow field is set, and all the other CRLs are present, as ‘Indeterminate’.

Fields with importance to CRL objects:

`"complete"`
: Set to ‘1’ if all messages have been received, otherwise ‘0’. If `"overflow"` is set to ‘1’, complete will be set to ‘0’.

`"overflow"`
: Will be ‘0’ is there is not overflow (more than 138 reports) or ‘1’ is there is overflow.

`"station"`
: Station name of the ground station.

`"product_type"`
: Type of product this applies to. Will have one of the following values:

    - `NOTAM/TFR`
    - `AIRMET`
    - `SIGMET`
    - `G-AIRMET`
    - `CWA`
    - `NOTAM/TRA`
    - `NOTAM/TMOA`

    Each string contains the message `"type"` that it represents. SIGMET includes WSTs. If you need both a message type and a subtype, they will be separated by a forward slash (`'/'`). For example, `AIRMET` refers to a message of type `AIRMET`. `NOTAM/TMOA` refers to a message of `"type"` `NOTAM` with a `"subtype"` of `TMOA`. Using this string and the ‘unique_name’ in the `"reports"` field, you could lookup any message.

`"range_nm"`
: Look-ahead range of the product in nautical miles. Different CRL types have different look-ahead ranges. Look-ahead range is also determined by the type of ground station (surface, low, medium, or high).

`"reports"`
: List of all possible reports. If the list is empty, there are no reports available for this CRL type. There will always be a `"reports"` field.

    Each line will have a format like:

    ```
    0-5116/TO*
    ```

    The first set of characters up to the slash (‘`/`’) is the ‘unique_name’ of the product. After the forward slash will be `'TG'` (text and graphics) or `'TO'` (text only). If there is an asterisk (`'*'`) at the end of the string, this indicates that the message has been received (in the case of `'TG'` items, it implies both text *and* graphic parts have been received).

Example:


```
{
  "type": "CRL_NOTAM_TFR",
  "unique_name": "40.0383~-86.255593",
  "station": "40.0383~-86.255593",
  "complete": 0,
  "overflow": 0,
  "product_id": 8,
  "range_nm": 100,
  "product_type": "NOTAM/TFR"
```

```
  "reports": [
      "0-5116/TO",
      "0-367/TO*",
      "0-9801/TO*",
      "0-229/TO*",
      "0-230/TO*",
      "1-5318/TO*"
   ],
  "expiration_time": "2021-06-26T20:08:20Z"
}
```

Service Status

```
(M) /service-status
```

Service status lists all aircraft receiving TIS-B or ADS-R services.

There is one of these objects for each ground station. The `"unique_name"` is the station name compatible with other FIS-B Rest `"station"` fields.

If there is no current traffic for a station, there will be no traffic message (i.e. there is no such thing as a empty traffic message).

The only new field is `"traffic"`:

`"traffic"`
: List of all aircraft being provided services by the ground station. The list contains the aircraft's ICAO address. Each address may optionally be followed by either a forward slash followed by a single digit, and/or a forward slash followed by one to three characters from the set '`T`', '`R`', and '`S`'. The number indicates the *address qualifier* whose value can be found in DO-282C. The normal address qualifier is '`0`', and no '`/0`' is appended if this is the case. The letters indicate the type of services the ground station is providing for the aircraft. '`T`' is TIS-B, '`R`' is ADS-R, and '`S`' is ADS-SLR. ADS-SLR stands for *Same Link Rebroadcast*. This is for aircraft on the ground when structures may block aircraft from receiving signals from other aircraft.

Example:

```
{
  "type": "SERVICE_STATUS",
  "unique_name": "40.0383~-86.255593",
  "expiration_time": "2021-06-26T22:40:24Z",
  "traffic": [
      "a8e069",
      "a8eb8e",
      "aa8cf4/1",
      "aba852/R",
      "a20c5c",
      "a20885/5/RS",
      "a03af6"
  ]
}
```

Reception Success Rate (RSR)

```
(1) /rsr
```

The Reception Success Rate (RSR) is the percentage of messages you are receiving vs the maximum number of messages you could have received. This calculation can be set up in different ways. See the 'fisb-decode' documentation for more details.

There is only one (or zero) RSR messages available at any given time. Each message contains a list of all stations being received and their RSR value.

The `"stations"` field is unique to RSR:

`"stations"`
: Dictionary whose keys are the names of ground stations being received in the usual `"station"` field format. The value is the percentage of possible packets being received. In the example below, 91% of messages from ground station `"40.0383~-86.255593"` are being received.

Example:

```
{
  "type": "RSR",
  "unique_name": "RSR",
  "stations": {
      "40.0383~-86.255593": 91
  },
  "expiration_time": "2021-06-26T22:43:55.130000Z"
}
```

Notes:

- If no stations are being received, any RSR message will expire and will not be created again until more messages arrive.

SUA (replaced by NOTAM-D SUA)

```
(M) /sua
```

These should no longer be used. They have been functionally replaced by NOTAM-D SUAs. As of 2020 the product range has been reduced to 5 NM.

Example:

```
{
  "type": "SUA",
  "unique_name": "21-6934",
  "start_time": "2021-06-25T15:00:00Z",
  "end_time": "2021-06-25T21:00:00Z",
  "schedule_id": "5988401",
  "airspace_id": "23941",
  "status": "P",
  "airspace_type": "B",
  "airspace_name": "AR113(W)",
  "expiration_time": "2021-06-25T21:00:00Z",
  "high_altitude": 23000,
  "low_altitude": 19000,
```

```
  "separation_rule": "A",
  "shape_defined": "Y"
}
```

Notes:

- Detailed description of the fields will not be described, because you should not use this. If you desire historical information, a good place to look is *Surveillance and Broadcast Services Description Document SRT-047 Revision 02* (2013). Revision 01 (2011) also has this information. Revision 05 (2020) makes note of the reduced product range and future elimination of this product.

Image Legends (static)

```
(1) /static/legend
```

Returns single object containing colors, legend text, and units of measurement for all images.

Consists of a dictionary whose keys represent the image type and will be one of the following:

- `"CLOUDTOP"`: Cloudtop image.
- `"ICING_PRB"`: Icing probability image.
- `"ICING_SEV"`: Icing severity image.
- `"ICING_SLD"`: Icing super large droplets image.
- `"LIGHTNING"`: Lighting image (both all lightning and positive lightning).
- `"RADAR"`: All RADAR images.
- `"TURBULENCE"`: Turbulence image.

Each image type holds a dictionary with these fields:

`"units"`
: String containing the text of the units represented by the values in `"colors"`.

`"colors"`
: Ordered list containing a set of two element lists consisting of:

    1. RGB (integer) color
    2. Text description for that color.

    The `"colors"` list is ordered in the form it should be displayed.

    There is a quirk in the standard where the ‘Severe’ value for the `"ICING_SEV"` image has a lower value than ‘Heavy’. This is corrected by ‘fisb-decode’ and ‘fisb-rest’ (and is a moot point since the source FIS-B product doesn’t actually have a ‘Severe’ value– but if they ever change this, you are set!).

Example:

```
{
  "CLOUDTOP": {
      "colors": [
          [14867152, "< 1500"],
          [14733760, "1500-3000"],
          [14731693, "3000-4500"],
```

```
            [14860697, "4500-6000"],
            [15120770, "6000-7500"],
            [15577449, "7500-9000"],
            [16753202, "9000-10500"],
            [15373608, "10500-12000"],
            [13928478, "12000-13500"],
            [12548886, "13500-15000"],
            [11103503, "15000-18000"],
            [9723913, "18000-21000"],
            [8278532, "21000-24000"],
            [6898689, ">24000"],
            [15522454, "No Data"]
        ],
        "units": "ft MSL"
    },
    "ICING_PRB": {
        "colors": [
            [7787519, "5-20"],
            [65280, "20-30"],
            [16776960, "30-40"],
            [15828533, "40-60"],
            [16711680, "60-80"],
            [16711935, ">80"],
            [15522454, "No Data"]
        ],
        "units": "%"
    },
    "ICING_SEV": {
        "colors": [
            [13303807, "Trace"],
            [9752061, "Light"],
            [6003708, "Moderate"],
            [670714, "Heavy"],
            [14299098, "Severe"],
            [15522454, "No Data"]
        ],
        "units": "Type"
    },
    "ICING_SLD": {
        "colors": [
            [16776960, "5-50"],
            [16711680, ">50"],
            [15522454, "No Data"]
        ],
        "units": "SLD %"
    },
    "LIGHTNING": {
        "colors": [
            [46321, "1"],
            [12704239, "2"],
            [5933115, "3-5"],
            [13230776, "6-10"],
            [16776960, "11-15"],
            [13197076, ">15"],
            [15522454, "No Data"]
        ],
        "units": "Strike Density"
    },
    "RADAR": {
        "colors": [
```

```
            [60977, "20-30"],
            [762654, "30-40"],
            [16776762, "40-45"],
            [16750398, "45-50"],
            [16711697, "50-55"],
            [16711931, ">55"],
            [15522454, "Not Incl"]
        ],
        "units": "dBZ"
    },
    "TURBULENCE": {
        "colors": [
            [13434481, "14-21"],
            [15588917, "21-28"],
            [16757806, "28-35"],
            [16749864, "35-42"],
            [16741923, "42-49"],
            [16731165, "49-56"],
            [16711704, "56-63"],
            [14876693, "63-70"],
            [12124177, "70-77"],
            [9371661, "77-84"],
            [8060940, "84-91"],
            [5439496, "91-98"],
            [4259846, ">98"],
            [15522454, "No Data"]
        ],
        "units": "EDR*100"
    }
}
```

Automation using systemd

Once your system is properly configured, you can automate everything using `systemd`. This will start-up fis-b rest at boot time. This assumes you are running a Linux system that uses systemd for scheduling system tasks.

The most important thing is to make sure your fis-b rest is properly configured. The `systemd` scripts use `gunicorn` as the HTTP server (this was installed when you installed the requirements). In the `fisb-rest` directory is the `gunicorn.conf.py` configuration file. Alter as you see fit. The stock file will bind the IP address to your current external IP address and port 7214. Two workers are configured. The rest of the file should not need any changes.

Note: If you installed `gunicorn` using the `requirements.txt` file, it will install it in your home directory as `/home/<username>/.local/bin/gunicorn`. When started as a service, `gunicorn` must be available as a general command. It doesn't count if it's in your path via `.bashrc` or something similar. The best way is to make a symbolic link such as:

```
sudo ln -s /home/<username>/.local/bin/gunicorn /usr/local/bin/gunicorn
```

Note: Don't forget to update the value for `URL_PREFIX` in `harvestConfig.py` (from the harvest distribution from fisb-decode. This determines the URL for web-images. It will not have the correct value for an external website.

Next, determine the non-root username you wish to run under, and the path to *fis-b rest* on your system. Then, from the `bin` directory, type:

```
./systemd-create username path-to-fisb-rest
```

If your account name is `fred` and the `fisb-rest` directory is located at `/home/fred/fisb-rest` you would type:

```
./systemd-create fred /home/fred/fisb-rest
```

This will create the files `fisb-rest_service` in the top-level `fisb-rest` directory and `fisb-rest.service` file in the `fisb-rest/misc` directory.

To install `fisb-rest` as a service (this will also start the web-server immediately and at boot time), type (from the `fisb-rest` directory):

```
sudo cp misc/fisb-rest.service /etc/systemd/system
sudo systemctl enable --now fisb-rest.service
sudo systemctl status fisb-rest.service
```

Check the `status` to make sure it is running. It should look similar to:

```
● fisb-rest.service - FIS-B Rest Web service
   Loaded: loaded (/etc/systemd/system/fisb-rest.service; enabled; vendor preset: ena
   Active: active (running) since Tue 2021-08-17 11:53:14 UTC; 2s ago
 Main PID: 31286 (bash)
    Tasks: 10 (limit: 9448)
   Memory: 97.5M
   CGroup: /system.slice/fisb-rest.service
           ├─31286 /bin/bash /home/mbarnes/fisb-rest/fisb-rest_service
           ├─31287 /usr/bin/python3 /usr/local/bin/gunicorn
           ├─31303 /usr/bin/python3 /usr/local/bin/gunicorn
           └─31304 /usr/bin/python3 /usr/local/bin/gunicorn
```

In general, if you wish to start, stop, restart, or disable (make it not run at boot), the service, issue the following commands:

```
sudo systemctl start fisb-rest.service
sudo systemctl stop fisb-rest.service
sudo systemctl restart fisb-rest.service
sudo systemctl disable fisb-rest.service
```