

# **RealTraffic Application Programmable Interface (API) Documentation**

Release 2.0  
July 2024

Questions or comments: Balthasar Indermühle <balt@inside.net>

# Contents

Readme First.....	4
Applications using this API.....	4
General workflow .....	5
API call examples .....	6
<b>/auth – Authentication .....</b>	<b>6</b>
POST parameters.....	6
Example call.....	6
Response .....	6
<b>/deauth - Deauthentication.....</b>	<b>7</b>
POST parameters.....	7
Example call.....	7
Response .....	7
<b>/traffic – Obtaining traffic .....</b>	<b>7</b>
POST parameters.....	7
Example call.....	8
Response .....	8
Weather system information .....	11
<b>/nearestmetar - Nearest Airports with METARs.....</b>	<b>12</b>
POST parameters.....	12
Example call.....	12
Response .....	12
<b>/weather – Obtaining weather.....</b>	<b>13</b>
POST parameters.....	13
Example call.....	13
Response .....	13
A note on implementing realistic weather in a simulator using RT weather data: .....	18
<b>/sigmet – Fetch global SIGMETs .....</b>	<b>20</b>
POST parameters.....	20
Example call.....	20
Response .....	20
<b>/airportinfo – Obtaining information about an airport.....</b>	<b>20</b>
POST parameters.....	21
Example call.....	21

Response .....	21
<b>/search – Find a flight in the system .....</b>	<b>23</b>
POST parameters.....	23
Example call.....	23
Response .....	23
<b>Getting the API example scripts to work .....</b>	<b>24</b>
Windows.....	24
Step 1: Download Python Installer .....	24
Step 2: Run the Installer .....	24
Step 3: Verify the Installation .....	24
Step 4: Upgrade pip (Optional but Recommended) .....	25
Step 5: Install required python modules .....	25
MacOS/Linux .....	25
Example: Show a map of parked aircraft at Sydney .....	25
Example: Show a map of live traffic in a 20km radius around Los Angeles.....	26
Example: Tying it all together with API_tester.py .....	27

## Readme First

RealTraffic aggregates global ADS-B data from multiple global data providers as well as weather information from over 5000 airports globally and from the 0.25-degree resolution GFS model weather data at 20 altitude layers vertically. The data is available for live, real-time consumption, and in identical resolution and fidelity up to 2 weeks into the past.

The APIs give you access to that full **traffic** data as well as **weather** and **airport** data – this is everything you need to have clear situation awareness of all aviation movements in your area of interest.

The responses to the API calls are given in JSON dictionaries. **Please make sure you use the gzip compression setting in the HTTPS requests** so as not to incur unnecessary data transfer cost at my end. Input data for the API is always expected as HTTP POST data.

Please refer to the included python example scripts that illustrate:

- The use of each API call
- API\_tester.py ties it all together and shows you a practical implementation of all the calls and provides you with a neat, terminal based, flight tracking experience!

The last chapter in this documentation gives detailed instructions on how to get the examples to work.

## Applications using this API

The main use case since RealTraffic's inception in 2017 has been to inject real air traffic into flight simulators. As of the writing of this document, the following programs are available for flight simulator traffic and weather injection:

Software	Supported simulators	Supported features		Platform	Link	Notes
		Traffic	Weather			
RealTraffic client	Aerowinx PSX	Ground Flying	Global WX METARs SIGMETs	Windows MacOS Linux	<a href="#">Visit</a>	
PSXT	MSFS 2004 MSFS 2020 P3D	Parked Ground Flying	No	Windows	<a href="#">Visit</a>	Standalone version
PSXTraffic	MSFS 2004 MSFS 2020 P3D	Parked Ground Flying	No	Windows	<a href="#">Visit</a>	Requires RealTraffic client
LiveTraffic	X-plane 11/12	Parked Ground Flying	Global WX METARs	Windows MacOS Linux	<a href="#">Visit</a>	

If you're a developer and would like your application listed here, please reach out to me: [balt@inside.net](mailto:balt@inside.net)

## General workflow

The sequence to establish a session and retrieve data works as follows:

- Authenticate using the license string, e.g. AAA-BBB-CCC-DDD
  - Authentication returns a session GUID (a globally unique identifier that identifies this license's session on the RT servers). Retrieve the GUID and use the GUID in all subsequent requests
  - It can also return a status 405: Too many sessions. In this case, wait 10 seconds and try authenticating again. Licenses are limited to a single session per standard license, and two sessions for a professional license.
- Loop:
  - Wait request rate limit (RRL) time in milliseconds since the last traffic request.
  - Fetch traffic.
  - Wait weather request rate limit (WRRL) time in milliseconds since the last weather request.
  - Fetch weather.
  - Check the status of each response. Status codes:
    - 200: Request processed ok
    - 400: Parameter fault. The message string contains details on which parameter you supplied (or didn't supply) that is causing the error.
    - 401: GUID not found, reauthenticate. If one is received, simply authenticate again and use the new session GUID for subsequent requests.
    - 402: License expired.
    - 403: No data for this license type (happens for example when trying to access historical data with a standard license).
    - 404: No data (license in the case of auth call) found. Did you send the correct license string? Typo?
    - 405: Too many sessions.
    - 406: Request rate violation. Make sure you wait for request rate limit (the number of) milliseconds specified in the traffic and weather returns. These are the "rrl" (traffic) and "wrml" (weather) entries respectively. These numbers can change during a session, make sure your code can accommodate that.
    - 407: History too far back. The maximum history supported at present is 28 days. If you request traffic or weather data that is further back, this error will trigger.
    - 500: An internal server error has occurred. The message string will contain some further information. If it's a repeatable problem, please let me know the details of that message string (i.e. you should log it)
- Deauthenticate when your client disconnects/shuts down. Please don't forget this, it helps keeping the server less cluttered with session data from the users. It also allows your client to reconnect immediately if for some reason you need to do that.

The GUID will remain in memory on the server for 120 seconds after the last use. If more than 120 seconds elapse between your calls, you will need to authenticate again and obtain a new GUID. Thus, session lifetime is 120 seconds (2 minutes).

## API call examples

Note in the cURL command line examples I'm explicitly setting the header to accept gzip compression and pipe the output into gunzip rather than using the --compressed option which does the same thing but in a simpler call.

This is purely to remind you to please use gzip compression in all of your calls! That means putting this in the HTTP header for all of your POST requests:

```
'Accept-encoding: gzip'
```

## /auth – Authentication

Address: <https://rtw.flyrealtraffic.com/v4/auth>

### POST parameters

- license
  - the user's RealTraffic license, e.g. AAA-BBB-CCC-DDD
- software
  - the name and version of your software.

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "license=AAA-BBB-CCC-DDD&software=MyCoolSoftware 1.0" -X POST https://rtw.flyrealtraffic.com/v4/auth | gunzip -
```

### Response

```
{ "status": 200, "type": 2, "rrl": 2000, "wrrl": 2000, "expiry": 1957899487 "GUID": "c0245e11-8840-46f9-9cf9-985183612495", "server": "rtw03a" }
```

- type
  - 1 = standard, 2 = professional
- rrl
  - traffic request rate limit in ms. Make sure you sleep that amount of time before your next traffic request or you will get a request rate violation.
- wrrl
  - weather request rate limit in ms. Make sure you sleep that amount of time before your next weather request or you will get a request rate violation.
- expiry
  - UNIX epoch second of the expiration date of the license.
- GUID
  - the session GUID you need to store and use in any communications going forward.
- server
  - the internal name of the server handling the request. Please indicate this name in case you run into repeatable problems using the API.

## /deauth - Deauthentication

Address: <https://rtw.flyrealtraffic.com/v4/deauth>

### POST parameters

- GUID: the session GUID

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=26cb1a52-169b-425e-8df6-7713e5c34835" -X POST https://rtw.flyrealtraffic.com/v4/deauth | gunzip -
```

### Response

```
{ "status": 200, "message": "Deauth success", "Session length": 1572, "server": "rtw01a" }
```

Note that the session length contains the duration of the session you just closed in number of seconds.

## /traffic – Obtaining traffic

Address: <https://rtw.flyrealtraffic.com/v4/traffic>

Traffic can be retrieved by giving a latitude/longitude box and a time offset in minutes from real-time.

Optionally, if you want to speed up the buffering internally, you can request up to 10 buffers of maximum 10s spacing (maximum 100s of wallclock time). This will return `buffer_0` to `buffer_n` in the data field of the response, with `buffer_0` being the youngest in time, and `buffer_n` the oldest. The `endepoch` parameter returned contains the unix epoch second of the oldest buffer that was returned.

### POST parameters

- GUID
  - the GUID obtained from the auth call.
- querytype
  - For standard traffic queries, pass the string “locationtraffic”  
For a secondary query that won’t run afoul of request rate violations, use “destinationtraffic”  
For parked traffic, use “parkedtraffic”. See below for the format of parked traffic (it is different from the standard format).
- bottom
  - southern latitude
- top
  - northern latitude
- left
  - western longitude
- right

- eastern longitude
- toffset
  - time offset into the past in minutes
- [optional] buffercount
  - The number of buffers to retrieve. Maximum 10.
- [optional] buffertime
  - The number of seconds between buffers (must be an even number). Maximum 10

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&top=-16&bottom=-17&left=145&right=146&querytype=locationtraffic&toffset=0" -X POST https://rtw.flyrealtraffic.com/v4/traffic | gunzip -
```

### Response

```
{
  "data": {
    "7c1522": ["7c1522", -16.878799, 145.66705, 103.16, 3025, 127.3, 1752,
    "X2", "A139", "VH-EGK", 1658644400.67, "MRG", "", "RSCU510", 0, -768, "RSCU510",
    "adsb_icao", 3350, 140, 148, 0.224, 1.06, 7.38, 93.87, 100.46, -800, "none", "A7",
    1014.4, 1024, 1008, , "tcas", 8, 186, 1, 9, 2, 0.1, -9.3, 0, 0, 91, 20, , , 1],
    "7c68a1": ["7c68a1", -16.754288, 145.693311, 156.07, 2325, 165.2, 6042,
    "X2", "E190", "VH-UYB", 1658644401.01, "BNE", "CNS", "QFA1926", 0, -960, "QF1926",
    "adsb_icao", 2625, 173, 182, 0.272, -0.09, -1.41, 146.6, 153.18, -928, "none",
    "A3", 1014.4, 3712, 2896, , "autopilot|approach|tcas", 8, 186, 1, 9, 2, 0.1, -20.2,
    0, 0, 123, 19, , , 1],
    "7c49c2": ["7c49c2", -16.866531, 145.743906, 0, 75, 0, , "X2", "C208", "VH-
    OUS", 1658644389.19, "", "", "OUS", 0, 0, "", "adsb_icao", , , , , , , , , , "",
    "A1", , , , , "", 8, 186, , , , 11.9, -24.9, 0, 0, , , , , 1],
    "7c7aa3": ["7c7aa3", -16.8705, 145.747, 247, 0, 0, , "F", "B738", "VH-YID",
    1658644396, "CNS", "SYD", "VOZ1424", 1, 0, "VA1424", "adsb_other", null, null,
    null, null, null, null, null, null, null, "null", "null", null, null, null, null,
    "null", null, null, null, null, null, null, null, null, null, null, null,
    null, 1]
  },
  "full_count": 6839,
  "source": "MemoryDB ",
  "rrl": 2000,
  "status": 200,
  "dataepoch": 16738882
}
```

Each aircraft has a state vector entry, with the transponder's hex ID as the key. Additional fields returned include:

- data
  - contains the aircraft data as a dictionary
- full\_count
  - The number of aircraft being tracked in the system at the requested point in time
- source
  - MemoryDB or DiskDB. Generally, data within 9h of real-time is kept in-memory on the server. Older data comes from disk storage.
- rrl



- The number of milliseconds you should wait before sending the next traffic request. During times of heavy load, you may be requested to poll less frequently. If you don't honour this request, you will receive request rate violation errors.
- status:
  - 200 for success, any other status indicates an error
- dataepoch
  - the epoch seconds in UTC of the data delivered

The data fields for each aircraft entry are as follows:

Example:

```
[ "7c68a1", -16.754288, 145.693311, 156.07, 2325, 165.2, 6042, "X", "E190", "VH-UYB", 1658644401.01, "BNE", "CNS", "QFA1926", 0, -960, "QF1926", "X_adsb_icao", 2625, 173, 182, 0.272, -0.09, -1.41, 146.6, 153.18, -928, "none", "A3", 1014.4, 3712, 2896, "autopilot|approach|tcas", 8, 186, 1, 9, 2, 0.1, -20.2, 0, 0, 123, 19, ,, 1]
```

- hexid (7c68a1)
- latitude (-16.754288)
- longitude (145.693311)
- track in degrees (156.07)
- barometric altitude in ft (std pressure) (2325)
- Ground speed in kts (165.2)
- Squawk / transponder code (6042)
- Data source: "X" (the provider code where the data came from)
- Type (E190)
- Registration (VH-UYB)
- Epoch timestamp of last position update (1658644401.01)
- IATA origin (BNE)
- IATA destination (CNS)
- ATC Callsign (QFA1926)
- On ground (0)
- Barometric vertical rate in fpm (-928)
- Flight number
- Message source type (X\_adsb\_icao) – see below for all possible types
- Geometric altitude in ft (=GPS altitude) (2625)
- Indicated air speed / IAS in kts (173)
- True air speed / TAS in kts (182)
- Mach number (0.272)
- Track rate of turn (-0.09) negative = left
- Roll / Bank (-1.41) – negative = left
- Magnetic heading (146.6)
- True heading (153.18)
- Geometric vertical rate in fpm (-928)
- Emergency (none)

- Category (A3) – see below for all possible categories
- QNH set by crew in hPa (1014.4)
- MCP selected altitude in ft (3712)
- Autopilot target altitude in ft (2896)
- Selected heading (empty)
- Selected autopilot modes (AP on, approach mode, TCAS active)
- Navigation integrity category (8) – see below for categories
- Radius of containment in meters (186)
- Navigation integrity category for barometric altimeter (1)
- Navigation accuracy for Position (9)
- Navigation accuracy for velocity (2)
- Age of position in seconds (0.1)
- Signal strength reported by receiver (-20.2 dbFS, -49.5 indicates a source that doesn't provide signal strength, e.g. ADS-C positions)
- Flight status alert bit (0)
- Flight status special position identification bit (0)
- Wind direction (123)
- Wind speed (19)
- SAT/OAT in C (none)
- TAT (none)
- Is this an ICAO valid hex ID (1)

The “message source type” field is preceded by ?\_ where ? contains any alphabetical code to indicate the data provider the data came from, and optionally is preceded by ID\_ if the data is interpolated data. The remainder can contain the following values:

- est: an estimated position
- adsb: a simplified ADS-B position only providing position, speed, altitude and track.
- adsb\_icao: messages from a Mode S or ADS-B transponder.
- adsr\_icao: rebroadcast of an ADS-B messages originally sent via another data link
- adsc: ADS-C (received by satellite downlink) – usually old positions, check tstamp.
- mlat: MLAT, position calculated by multilateration. Usually somewhat inaccurate.
- other: quality/source undisclosed.
- mode\_s: ModeS data only, no position.
- adsb\_other: using an anonymised ICAO address. Rare.
- adsr\_other: rebroadcast of ‘adsb\_other’ ADS-B messages.

Transmitter categories:

- A0 – No information
- A1 – Light ( < 15,500 lbs)
- A2 – Small (15,500 – 75,000 lbs)
- A3 – Large (75,000 – 300,000 lbs)
- A4 – High vortex generating Acft (e.g. B757)
- A5 – Heavy ( > 300,000 lbs)
- A6 – High Performance ( > 5G accel, > 400 kts)
- A7 – Rotorcraft

- B0 – No information
  - B1 – Glider/Sailplane
  - B2 – Lighter-than-air
  - B3 – Parachutist/Skydiver
  - B4 – Ultralight/Hangglider/Paraglider
  - B5 – Reserved
  - B6 – Unmanned Aerial Vehicle / Drone
  - B7 – Space/Trans-atmospheric vehicle
- 
- C0 – No Information
  - C1 – Surface vehicles: Emergency
  - C2 – Surface vehicles: Service
  - C3 – Point obstacles (e.g. tethered balloon)
  - C4 – Cluster obstacle
  - C5 – Line obstacle
  - C6-7 – Reserved
- 
- D0 – No information
  - D1-7 – Reserved

Queries for parkedtraffic will return any traffic whose last groundspeed was zero, and whose position timestamp is at least 10 minutes old, but less than 24h old, in following format:

```
{
  "data": {
    "7c4920": [-33.936407, 151.169229, 0.0, "A388", "VH-OQA", 1721590016.01,
"QFA2", 123],
    "7c5325": [-33.936333, 151.170109, 0.0, "A333", "VH-QPJ", 1721597924.81,
"QFA128", 350],
    "7c765a": [-33.935635, 151.17746, 0, "A320", "VH-XNW", 1721650004.0,
"JST825", 320]
  },
  "full_count": 13765,
  "source": "DiskDB",
  "rrl": 2000,
  "status": 200,
  "dataepoch": 1721650780
}
```

The fields are latitude, longitude, last groundspeed, type, tail, last timestamp when it was moving into place, ATC callsign, last track (for orientation).

## Weather system information

RealTraffic provides access to the highest resolution GFS model data available. This allows a realistic weather environment to be simulated, including historical time offsets.

As a first step, you should obtain the airport ICAO code of the nearest airport with METAR information in the RT system. This can be achieved by calling the /nearestmetarcode

endpoint. The retrieved airport code should then be fed into the /weather endpoint query so that the actual METAR can be retrieved. For best results, you should always use the cloud and wind information from the METARs rather than the global forecasting model when flying near the ground and near an airport. For conditions aloft while enroute (wind, temperature, cloud, and turbulence), you can rely on the altitude profile data returned by the /weather endpoint.

## /nearestmetar - Nearest Airports with METARs

Address: <https://rtw.flyrealtraffic.com/v4/nearestmetar>

### POST parameters

- GUID
  - the GUID obtained from the auth call
- toffset
  - time offset into the past in minutes
- lat
  - latitude in degrees
- lon
  - longitude in degrees (west = negative)
- maxcount
  - the number of nearest airports to find. Maximum = 20. If omitted, returns the nearest only the nearest airport.

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&lat=47.5&&lon=5.5&toffset=0" -X POST https://rtw.flyrealtraffic.com/v4/nearestmetarcode | gunzip -
```

### Response

```
{'ICAO': 'KNSI', 'Dist': 10.3, 'BrgTo': 42.3, 'METAR': "KNSI 130053Z AUTO 28010KT 10SM FEW009 00/ A2986 RMK A02 SLP114 T0000 $"}
```

- ICAO
  - The ICAO code of the nearest Airport that has a METAR. Use this to feed the airports parameter in the weather API call.
- Dist
  - The distance to the airport in NM
- BrgTo
  - The true bearing to the airport
- METAR
  - The current METAR for that airport

If multiple airports are returned, they show up as a list (only showing the "data" part of the response object), sorted by distance (closest first):

```
{ ... "data":
```

```
[{"ICAO": "KNSI", "Dist": 764.3, "BrgTo": 32.1, "METAR": "KNSI 130053Z AUTO 28010KT 10SM FEW009 00\ A2986 RMK A02 SLP114 T0000 $"}, {"ICAO": "KAVX", "Dist": 804, "BrgTo": 34.7, "METAR": "KAVX 130051Z AUTO VRB06KT 10SM CLR 20\14 A2990 RMK A02 SLP111 T02000144"}, {"ICAO": "KLPC", "Dist": 809.6, "BrgTo": 25.6, "METAR": "KLPC 130056Z AUTO 27010KT 10SM CLR 18\15 A2986 RMK A02 SLP110 T01830150"}]
```

## /weather – Obtaining weather

Address: <https://rtw.flyrealtraffic.com/v4/weather>

Weather (METARs and TAFs) and the local GFS derived weather can be retrieved using this call.

### POST parameters

- GUID
  - the GUID obtained from the auth call
- querytype
  - should be set to the string "locwx"
- toffset
  - time offset into the past in minutes
- lat
  - latitude in degrees
- lon
  - longitude in degrees (west = negative)
- alt
  - geometric altitude in ft
- airports
  - a list of airports for which to retrieve METARs, delimited by pipe (|)

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&lat=47.5&lon=5.5&alt=37000&airports=LSZB|LSZG|LFSB&querytype=locwx&toffset=0" -X POST https://rtw.flyrealtraffic.com/v4/weather | gunzip -
```

### Response

**Note** that the locWX field in the response only returns data when sufficient difference in distance (10NM) or altitude (2000ft) has elapsed between the last call and the current call, or if more than 60 seconds have elapsed since the last call, or if it is the initial call.

```
{
  'ICAO': 'KLBX',
  'QNH': 2958,
  'METAR': 'KLBX 080721Z AUTO 09023G50KT 2SM +RA BR FEW012 OVC023 26/26 A2958 RMK A02 PK WND 12050/0712 PRESFR P0016 T02560256',
  'locWX': {
    'Info': '2024-07-08_0740Z',
    'SLP': 1001.23,
```

```

'WSPD': 99.07,
'WDIR': 181.09,
'T': -24.42,
'ST': 27.87,
'SVis': 4342,
'SWSPD': 85.24,
'SWDIR': 127.7,
'DZDT': 0.524,
'LLC': {
  'cover': 100.0,
  'base': 111,
  'tops': 5180,
  'type': 1.77,
  'confidence': 0.61
},
'MLC': {
  'cover': 100.0,
  'base': 5196,
  'tops': 10359,
  'type': 1.77,
  'confidence': 0.61
},
'HLC': {
  'cover': 100.0,
  'base': 10375,
  'tops': 15193,
  'type': 1.65,
  'confidence': 0.44
},
'TPP': 15559.67,
'PRR': 4.33,
'CAPE': 1854.0,
'DPs': [25.5, 23.77, 21.67, 20.43, 17.47, 14.54, 11.97, 9.43, 6.5, 3.04, -
0.16, -2.9, -6.57, -11.0, -16.54, -24.54, -34.73, -48.57, -64.27, -80.04],
'TEMPs': [27.53, 25.07, 21.73, 20.43, 17.53, 15.0, 12.37, 9.63, 6.77, 3.27,
-0.1, -2.67, -6.17, -10.57, -16.44, -24.54, -34.4, -47.97, -64.27, -74.04],
'WDIRs': [127.7, 130.87, 136.95, 141.37, 151.15, 152.4, 154.78, 154.59,
151.98, 153.85, 159.76, 162.24, 165.67, 169.51, 174.98, 181.18, 185.82, 168.36,
197.06, 167.62],
'WSPDs': [85.24, 114.85, 126.5, 129.37, 128.72, 128.36, 128.12, 124.07,
117.05, 115.72, 111.24, 104.53, 102.28, 101.74, 100.29, 99.06, 89.58, 46.88, 62.02,
31.7],
'DZDTs': [0.0, 0.02, 0.05, 0.06, 0.04, 0.0, -0.01, 0.01, 0.05, 0.06, 0.09,
0.18, 0.33, 0.52, 0.62, 0.52, 0.37, 0.29, 0.15, -0.03],
'Profiles': 'RTFX1          ^N2900.0          ^W09500.0
^FL339 186/048 -34 ^FL319 183/051 -29 ^FL299 181/053 -24 ^FL279 177/054 -20
^FL259 174/054 -15 ^^'
},
'AM': []
}

```

An explanation of the fields in the response:

- ICAO
  - the ICAO code of the nearest airport. This is the first airport from the list of ICAO codes you supplied in the API call to get the METARs for. See the /nearestmetarcode API call to retrieve this before calling weather.

- QNH
  - the reported pressure in hPa, often to within 0.1 hPa precision. For numbers greater than 2000, the unit is inHg, not hPa.
- METAR
  - contains the full METAR received,
- locWX
  - the GFS derived location weather at the present position and time, and contains the following information:
    - info
      - contains the timestamp if data is present, if no data is present it contains the reason for no data. Valid reasons are:
        - TinyDelta: Means that less than one minute has elapsed since the last query, or the lateral/vertical distance to the last query is less than 10NM / 2000ft.
        - error: Means there was an error on the server, the description after error contains more information. When asking for older historical data that is not in the cache, you may see the message “File requested” in this text, which indicates that it is being retrieved from storage. This takes about 30 seconds to complete, so it would be a good idea to wait 30 seconds before launching the next weather request for that given timestamp.
        - If no data is provided (because not enough time or distance has elapsed to the last call), all fields are set to -1
- ST
  - surface temperature in C
- SWSPD
  - surface wind speed in km/h
- SWDIR
  - surface wind direction in km/h
- SVis
  - surface visibility in meters
- CAPE
  - the convective available potential energy in J/kg. This is an indicator for the vertical stability of the atmospheric column at this location.
- PRR
  - precipitation rate on ground in mm/h.
    - < 0.5: none - or drizzle if not zero
    - < 2.5: light
    - < 7.5: moderate
    - > 7.5: heavy
- LLCC
  - low level (lowest third of troposphere) cloud details:
    - cover: cloud cover in percent
    - base: the cloud base in meters

- tops: the cloud tops in meters
  - type: Type of cloud on a sliding scale from 0-3, see cloud types below.
  - confidence: The confidence with which the cloud type has been estimated (0=low, 1=very high).
- MLCC
  - medium level (middle third of troposphere):
    - cover: cloud cover in percent
    - base: the cloud base in meters
    - tops: the cloud tops in meters
    - type: Type of cloud on a sliding scale from 0-3, see cloud types below.
    - confidence: The confidence with which the cloud type has been estimated (0=low, 1=very high).
- HLCC
  - high level (top third of troposphere) cloud cover in percent:
    - cover: cloud cover in percent
    - base: the cloud base in meters
    - tops: the cloud tops in meters
    - type: Type of cloud on a sliding scale from 0-3, see cloud types below.
    - confidence: The confidence with which the cloud type has been estimated (0=low, 1=very high).
- DZDT
  - vorticity of atmospheric layer. Larger values are indicative of turbulence. As a rule of thumb:
    - < 0.05: still air
    - < 0.5: light
    - < 1: medium (spills coffee)
    - > 1: strong (unattached objects and people go flying)
    - > 2: severe (unable to retain positive aircraft control at all times – block altitude needed)
- T
  - OAT/SAT in C at the supplied altitude
- WDIR
  - wind direction in degrees at the supplied altitude
- WSPD
  - wind speed in km/h at the supplied altitude
- TPP
  - tropopause height in meters
- SLP
  - sea level pressure in hPa
- DPs
  - An array of dew point temperatures in C in a vertical cross section of the atmosphere. See below for the altitudes.
- TEMPs



- An array of still air temperatures in C in a vertical cross section of the atmosphere. See below for the altitudes.
- WDIRs
  - An array of wind directions (true north oriented) in a vertical cross section of the atmosphere. See below for the altitudes.
- WSPDs
  - An array of wind speeds in km/h in a vertical cross section of the atmosphere. See below for the altitudes.
- DZDTs
  - An array of turbulence measurements in m/s in a vertical cross section of the atmosphere. See below for the altitudes.
- Profiles
  - contains a vertical cross section / profile at the current location, formatted as “Aerowinx Format D”. This is a common output format used in flight planning and dispatch software. The format contains a line with caret symbols as newline placeholders. In the example above, the line reads as follows:

```
RTFX1
N3737.2
W12034.8
FL381 265/071 -51
FL361 267/075 -52
FL341 269/078 -53
FL321 269/082 -51
FL301 268/086 -48
```

The first line contains the waypoint name (RT Fix 1)

The second and third lines contain the coordinates of the fix

Lines 4 – 8 contain the altitude (or FL) in 2000ft increments above and below the current FL, and the wind direction / wind speed and OAT in C for each of those altitudes.

- AM
  - additional METARs. Contains a list of an additional up to 6 METARs (they must be requested). You can use these METARs to gain additional understanding of the weather surrounding the aircraft.

The atmospheric cross section parameters listed above (TEMPs, DPs, WDIRs, WSPDs, DZDTs) correspond to the following altitude levels (in meters), in the same order:

[111, 323, 762, 988, 1457, 1948, 2465, 3011, 3589, 4205, 4863, 5572, 6341, 7182, 8114, 9160, 10359, 11770, 13503, 15790]

The cloud types in the LLC/MLC/HLC fields correspond to the following types, on a sliding scale from 0 – 3, where 0 = Cirrus, 1 = Stratus, 2 = Cumulus, and 3 = Cumulonimbus.

A note on implementing realistic weather in a simulator using RT weather data:

The locWX data is always model derived, while METARs are always observation derived. Therefore, METARs are **always** more accurate than the locWX derived data, keep that in mind.

So for locations where you have METAR, you should always use the METAR for the information that METARs provide (see below for limitations!). In other places and for information not supplied by METARs, use locWX.

METARs normally are updated every 30 minutes by the airport systems or on-duty meteorologist. If weather is rapidly changing (e.g. thunderstorms developing, a frontal system coming through with rapid wind changes), they are issued more frequently, at most every 10 minutes.

RT uses a 10 minute cadence to refresh METAR data. That means you won't see updated METARs for locations with slowly changing weather conditions except for twice an hour usually, or sometimes even just once an hour. This is not a malfunction of the METAR system in RT, but a function of the METAR update frequency at the airports.

locWX doesn't do a great job depicting thunderstorms but works great with other severe phenomena like hurricanes. Why is this? Thunderstorms are small and localised atmospheric phenomena that can only be forecast statistically. The CAPE parameter in locWX gives some indication as to the probability of cumulonimbus (CB) developing, and when it does, as to the severity.

CAPE values from 100 upwards indicate atmospheric instability. With a CAPE of 100, if CBs develop, they're unlikely to be severe. A CAPE of 3000 however would be indicative of a high likelihood for CBs, and severe ones at that.

A hurricane by contrast is a weather system that can be modelled quite accurately and can be predicted. This is why a hurricane will look nearly perfect in the locWX data, but thunderstorms do not.

The same goes for weather frontal systems such as cold- or warm fronts, or occlusions, they are modelled really well in locWX data because they span a large geographic area.

METARs by contrast only give you a spot measurement on the ground and provide the best information only for landing and taking off.

For good enroute weather, and at altitude, you should use thunderstorm indicators in METARs only to modify the cloud type. Use other cloud, pressure, wind, and temperatures from locWX.

It helps to think of these data points in terms of geographic extent:

- METARs are relevant to ~10NM radius around the location of the observation. Vertically you can only derive cloud bases from them - to some extent - as METARs generally only report operationally relevant clouds (see the list below

for details). So higher level clouds shown in locWX may in fact be there even if METAR reports no clouds.

- locWX parameters are relevant on scales of 15NM extent and larger (15NM being the resolution used in the model data the locWX data is derived from). locWX provides good accuracy for vertical temperature profiles up to and past the tropopause, and wind directions are generally quite accurate once outside the ground effect (above 1000m AGL in most locations, higher in mountainous regions).

If you find any of the following cloud keywords in the METAR, follow this guidance on what to do:

- SKC (Sky Clear)
  - This indicates that no clouds are present at all.
  - **Action:**  
You can use this to override any cloud shown in locWX.
- CLR (Clear)
  - Used in automated reports to indicate no clouds detected below 12,000 feet (3,700 m).
  - **Action:**  
If locWX has cloud lower than 12,000ft, you can discard it. Keep the higher cloud if present in locWX.
- CAVOK (Ceiling and Visibility OK)
  - This indicates simultaneously that:
    - Visibility is 10 km (6 statute miles) or more
    - No clouds below 5,000 feet (1,500 m) or below the highest MSA (minimum sector altitude), whichever is greater
    - No cumulonimbus (CB) or towering cumulus (TCU) at any level
    - No significant weather phenomena
  - **Action:**  
If locWX has cloud lower than 6,000ft above the local terrain (5,000ft + 1,000ft MSA safety buffer), you can discard it.
  - Keep any higher altitude cloud.
  - Note you can look up the MSAs for an airport using the /airportinfo API.
- NSC (No Significant Cloud)
  - Used when there are no clouds below 5,000 feet (1,500 m) or below the highest MSA, whichever is greater, and no cumulonimbus or towering cumulus at any level.
  - **Action:**  
If locWX has cloud lower than 6,000ft above the local terrain (5,000ft + 1,000ft MSA safety buffer), you can discard locWX cloud.
  - Note you can look up the MSAs for an airport using the /airportinfo API.
- NCD (No Cloud Detected)

- Used in some automated reports when the ceilometer detects no clouds.
- **Action:**  
Use locWX cloud. NCD are automated systems, they can and do malfunction.
- VV /// (Vertical Visibility undefined)
  - In some cases, when the sky is obscured and the vertical visibility can't be assessed, this might indicate a lack of detectable cloud base.
  - **Action:**  
Use locWX derived cloud.

## /sigmet – Fetch global SIGMETs

Address: <https://rtw.flyrealtraffic.com/v4/sigmet>

Obtains the global SIGMET data for the requested time period.

### POST parameters

- GUID
  - the GUID obtained from the auth call
- toffset
  - The time offset into the past in minutes

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&toffset=0" -X POST https://rtw.flyrealtraffic.com/v4/sigmet | gunzip -
```

### Response

```
{
  "source": "MemoryDB",
  "status": 200,
  "data": {
    "DATE": "2024-07-22 04:00:00",
    "DATA": "WSCI35 ZGGG 212345^ZGZU SIGMET 6
    VALID 220015/220415 ZGGG-^ZGZU GUANGZHOU FIR EMBD TS FCST WI N2336 E11658 - N2242
    E10819 -^N2140 E10659 - N2030 E10804 - N2030 E11131 - N2255 E11659 - N2336^E11658
    TOP FL450 MOV NW 30KMH NC=^^WSMX31 MMMX 220021
    ...
    ^BOUNDED BY 20S UIN-50SSW STL-40NNW ARG-50N LIT-40ESE MLC-60E TTT- ^40SSW TTT-50S
    SPS-30WNNW SPS-70SSE MMB-40E OSW-50SSW IRK-20S UIN ^CIG BLW 010/VIS BLW 3SM BR.
    CONDS ENDG 12-15Z.^^"
  }
}
```

You will need to parse the SIGMET data on your own. This is a complex format, but some help is available online on how to parse it successfully.

## /airportinfo – Obtaining information about an airport

Address: <https://rtw.flyrealtraffic.com/v4/airportinfo>

Provides information on an airport. This includes airport general information, minimum sector altitudes, and runways, and any available ILS systems for the runways.

## POST parameters

- GUID
  - the GUID obtained from the auth call
- ICAO
  - The ICAO code of the airport you want the information for

## Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&ICAO=LSZB" -X POST https://rtw.flyrealtraffic.com/v4/airportinfo | gunzip -
```

## Response

```
{
  "data": {
    "MSA": {
      "MSA_center": "LSZB",
      "MSA_center_lat": 46.91222222,
      "MSA_center_lon": 7.49944444,
      "MSA_radius_limit": 25,
      "MSA_sector1_alt": 15800,
      "MSA_sector1_brg": 250,
      "MSA_sector2_alt": 10700,
      "MSA_sector2_brg": 5,
      "MSA_sector3_alt": 7200,
      "MSA_sector3_brg": 65
    },
    "airport": {
      "elevation": 1675,
      "name": "BELP",
      "ref_lat": 46.91222222,
      "ref_lon": 7.49944444,
      "transition_altitude": 6000,
      "transition_level": -1
    },
    "runways": {
      "RW14": {
        "declination": 2.0,
        "displaced_threshold_distance": 656,
        "gradient": 0.123,
        "gs_angle": 4.0,
        "ils_cat": 1,
        "landing_threshold_elevation": 1668,
        "lat": 46.91793889,
        "length": 5676,
        "llz_brg": 138.0,
        "llz_freq": 110.1,
        "llz_ident": "IBE",
        "lon": 7.49249444,
        "mag_brg": 138.0,
        "surface": "Asphalt",
        "threshold_crossing_height": 43,
        "true_brg": 140.197,
        "width": 98
      },
      "RW32": {
```

```

    "declination": -1,
    "displaced_threshold_distance": 0,
    "gradient": -0.123,
    "gs_angle": -1,
    "ils_cat": -1,
    "landing_threshold_elevation": 1675,
    "lat": 46.90738889,
    "length": 5676,
    "llz_brg": -1,
    "llz_freq": -1,
    "llz_ident": "",
    "lon": 7.50536111,
    "mag_brg": 318.0,
    "surface": "Asphalt",
    "threshold_crossing_height": 50,
    "true_brg": 320.206,
    "width": 98
  }
},
"message": "OK",
"rrtl": 2000,
"status": 200
}

```

Notes on selected items of the returned data:

- MSA (Minimum Sector Altitude) can contain up to 5 sectors. Note the bearings are TO the airport (not a radial FROM the airport) – imagine flying towards the airport to find out which sector you are in. Interpretation goes clockwise as follows in the shown example:
  - Sector 1 goes from 250 to 005 degrees and the MSA is 15800ft
  - Sector 2 goes from 005 to 065 degrees and the MSA is 10700ft
  - Sector 3 goes from 065 to 250 degrees and the MSA is 7200ft
- Transition (TL) level is returned as -1 if it is not applicable. Many airports don't have a fixed TL, but make this dependent on the atmospheric conditions. The TL will be given by ATC for these airports (i.e. is not available for RealTraffic use).
- Runways contains information on all the runways at the airport and any ILS that might be available for this runway. The values are empty or -1 for runways that do not have an ILS. Specifically:
  - Declination contains the magnetic declination at the location of the localizer (LLZ) transmitter
  - gs\_angle is the glideslope (GS) angle in degrees
  - ils\_cat is the certified ILS category
  - llz\_brg is the localiser bearing in degrees
  - llz\_freq is the localizer frequency in MHz (aka the "ILS Frequency"). LLZ and GS are transmitted on separate frequencies, but to keep operations simple, every LLZ frequency is matched to a GS frequency on a fixed relationship. Pilots therefore only need to set the LLZ frequency (which in aviation is called the ILS frequency).
  - llz\_ident is the identifier for the ILS

## /search – Find a flight in the system

Address: <https://rtw.flyrealtraffic.com/v4/search>

This lets you search the RealTraffic data for a flight.

### POST parameters

- GUID
  - the GUID obtained from the auth call
- toffset
  - The time offset into the past in minutes
- searchParam
  - The parameter to search. Options are:
    - Callsign
      - Search the ATC callsign. Matches any substring. E.g. CPA will match CPA123, CPA345 etc.
    - CallsignExact
      - Search the ATC callsign but only return an exact match.
    - FlightNumber
      - Search the IATA flight code. Matches any substring. E.g. CX will match CX123, CX345 etc.
    - FlightNumberExact
      - Search the IATA flight code but only return the exact match.
    - From
      - Return flights originating from this IATA airport code. E.g. HND will return all flight originating at Tokyo Haneda.
    - To
      - Return flights with this destination IATA airport code. E.g. NRT will return all flights with destination Narita.
    - Type
      - Return all aircraft of this given ICAO aircraft type. E.g. B77W will return all Boeing 777
    - Tail
      - Return all flights with this tail number. E.g. N1234 will find N1234 as well as N12345 etc.
- search
  - The term to search for, e.g. CPA123 if you're looking for a callsign of that description

### Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=76ff411b-d481-470f-9ce5-5c3cbc71a276&toffset=0&searchParam=Callsign&search=CPA1" -X POST https://rtw.flyrealtraffic.com/v4/search | gunzip -
```

### Response

```
{
  "status": 200,
  "message": "OK",
  "rrl": 2000,
  "data": {
    "780a37": ["780a37", 17.709503, 119.143596, 327.99, 38000, 490.6, "1165",
"H", "B77W", "B-KQE", 1721602623.48, "BNE", "HKG", "CPA156", 0, 0, "CX156"],
    "780a5c": ["780a5c", -22.671359, 142.890015, 137.29, 35000, 566.1, "5334",
"H", "B77W", "B-KQL", 1721602623.5, "HKG", "SYD", "CPA101", 0, 0, "CX101"],
    "789232": ["789232", -18.29892, 136.87906, 163, 39000, 499, "", "H",
"A35K", "B-LXL", 1721602613.0, "HKG", "MEL", "CPA105", 0, 0, "CX105"],
    "780abb": ["780abb", -30.371052, 145.94897, 318, 40000, 453, "", "H",
"A359", "B-LRP", 1721602613.0, "SYD", "HKG", "CPA110", 0, 0, "CX110"],
    "780237": ["780237", -22.652506, 142.8712, 137, 35000, 566, "", "H",
"A333", "B-LAL", 1721602613.0, "HKG", "SYD", "CPA101", 0, 0, "CX101"],
    "789246": ["789246", -30.21781, 162.6366, 131, 37000, 596, "", "H", "A35K",
"B-LXN", 1721602611.0, "HKG", "AKL", "CPA113", 0, 0, "CX113"],
    "780a61": ["780a61", 18.049524, 116.35497, 332, 38000, 489, "", "H",
"B77W", "B-KQM", 1721602619.0, "PER", "HKG", "CPA170", 0, 0, "CX170"]
  }
}
```

## Getting the API example scripts to work

The API documentation download includes an example call for each of the APIs. In order to get the python-based examples to work, you will need to install some packages. Following is a non-exhaustive installation instruction by platform.

### Windows

#### Step 1: Download Python Installer

1. Open your web browser and go to the official Python website:  
<https://www.python.org/downloads/windows/>
2. Click on the button that says "Download Python X.X.X" (where X.X.X is the latest version number).
3. The download should start automatically. If not, click on the download link provided.

#### Step 2: Run the Installer

1. Once the download is complete, locate the installer file (usually in your Downloads folder) and double-click it to run.
2. Important: On the first screen of the installer, check the box that says "Add Python X.X to PATH". This will make it easier to run Python from the command line.
3. Click "Install Now" to start the installation with default settings (recommended for most users).

#### Step 3: Verify the Installation

1. After the installation completes, open the Command Prompt:
  - Press Win + R, type "cmd", and press Enter.
2. In the Command Prompt, type the following commands to check if Python and pip are installed correctly:



```
python --version  
pip --version
```

#### Step 4: Upgrade pip (Optional but Recommended)

It's a good practice to upgrade pip to the latest version:

1. In the Command Prompt, type:

```
python -m pip install --upgrade pip
```

#### Step 5: Install required python modules

1. Run the following command on the command prompt:

```
pip install matplotlib requests textalloc cartopy
```

You should now be ready to run the python API example scripts

#### MacOS/Linux

Python comes preinstalled for these platforms, or is available as a standard package from the distribution channels.

Simply install the modules required to run the scripts:

```
pip install matplotlib requests textalloc cartopy
```

#### Example: Show a map of parked aircraft at Sydney

To make a plot of all parked traffic in a 3km radius around Sydney International Airport (YSSY), enter the following on your command line / terminal:

```
python API_traffic.py --airport YSSY -t parkedtraffic --plot -r 3
```

This prints the output of the API calls, and creates a plot saved as plot.jpg:

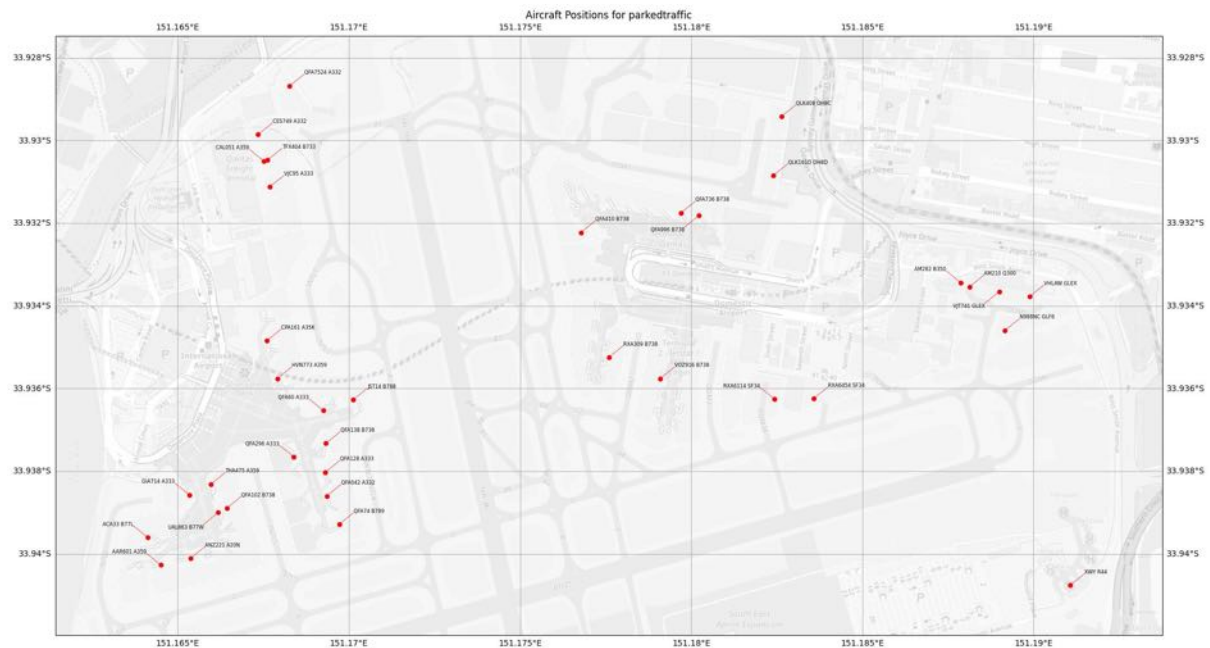


Figure 1 Parked traffic in a 3km radius from the center of YSSY

Example: Show a map of live traffic in a 20km radius around Los Angeles

To make a plot of all live air traffic in a 20km radius around Los Angeles International Airport (KLAX), enter the following on your command line / terminal:

```
python API_traffic.py --airport KLAX -t locationtraffic --plot
```

This prints the output of the API calls, and creates a plot saved as plot.jpg:

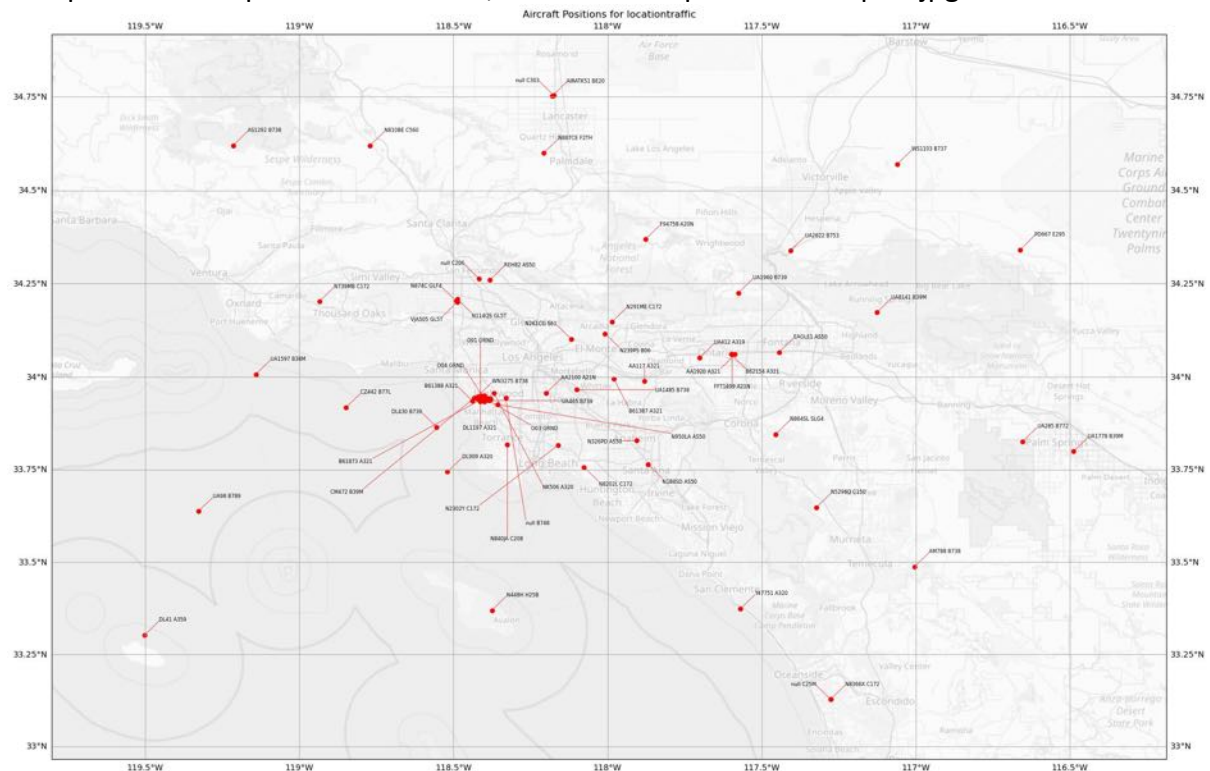


Figure 2 Live airtraffic in a 100km radius around KLAX

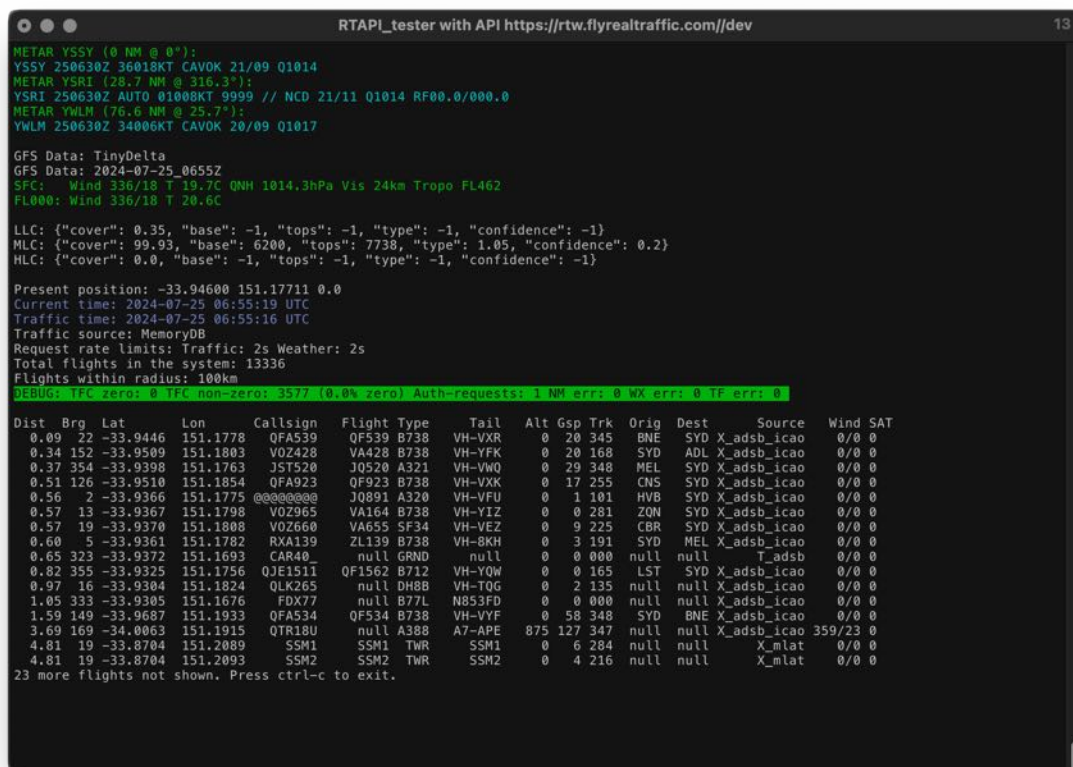
### Example: Tying it all together with API\_tester.py

The script API\_tester.py combines all of the API calls into a single interface, continuously showing weather and traffic for a specified location, or following a flight of your choice.

Simply start by running this command:

```
./API_tester.py --nummetars 3 -a YSSY
```

This places you at YSSY, shows the 3 nearest METAR stations, and keeps refreshing the data.

A terminal window titled "RTAPI\_tester with API https://rtw.flyrealtraffic.com/dev" showing live traffic data for YSSY. The output includes METAR reports for YSSY, YSRI, and YWLM, followed by GFS weather data and a list of nearby flights. The flight list is a table with columns: Dist, Brg, Lat, Lon, Callsign, Flight Type, Tail, Alt, Gsp, Trk, Orig, Dest, Source, Wind, SAT. The first 23 flights are shown, and the rest are omitted.

```
RTAPI_tester with API https://rtw.flyrealtraffic.com/dev 13
METAR YSSY (0 NM @ 0°):
YSSY 250630Z 36018KT CAVOK 21/09 Q1014
METAR YSRI (28.7 NM @ 316.3°):
YSRI 250630Z AUTO 01008KT 9999 // NCD 21/11 Q1014 RF00.0/000.0
METAR YWLM (76.6 NM @ 25.7°):
YWLM 250630Z 34006KT CAVOK 20/09 Q1017

GFS Data: TinyDelta
GFS Data: 2024-07-25_0655Z
SFC: Wind 336/18 T 19.7C QNH 1014.3hPa Vis 24km Tropo FL462
FL000: Wind 336/18 T 20.6C

LLC: {"cover": 0.35, "base": -1, "tops": -1, "type": -1, "confidence": -1}
MLC: {"cover": 99.93, "base": 6200, "tops": 7738, "type": 1.05, "confidence": 0.2}
HLC: {"cover": 0.0, "base": -1, "tops": -1, "type": -1, "confidence": -1}

Present position: -33.94600 151.17711 0.0
Current time: 2024-07-25 06:55:19 UTC
Traffic time: 2024-07-25 06:55:16 UTC
Traffic source: MemoryDB
Request rate limits: Traffic: 2s Weather: 2s
Total flights in the system: 13336
Flights within radius: 100km
DEBUG: TFC zero: 0 TFC non-zero: 3577 (0.0% zero) Auth-requests: 1 NM err: 0 WX err: 0 TF err: 0

Dist Brg Lat Lon Callsign Flight Type Tail Alt Gsp Trk Orig Dest Source Wind SAT
0.09 22 -33.9446 151.1778 QFA539 QF539 B738 VH-VXR 0 20 345 BNE SYD X_adsb_icao 0/0 0
0.34 152 -33.9509 151.1803 VOZ428 VA428 B738 VH-YFK 0 20 168 SYD ADL X_adsb_icao 0/0 0
0.37 354 -33.9398 151.1763 JST520 J0520 A321 VH-VWQ 0 29 348 MEL SYD X_adsb_icao 0/0 0
0.51 126 -33.9510 151.1854 QFA923 QF923 B738 VH-VXK 0 17 255 CNS SYD X_adsb_icao 0/0 0
0.56 2 -33.9366 151.1775 @@@@@@@@ JQ891 A320 VH-VFU 0 1 101 HVB SYD X_adsb_icao 0/0 0
0.57 13 -33.9367 151.1798 VOZ965 VA164 B738 VH-YIZ 0 0 281 ZQN SYD X_adsb_icao 0/0 0
0.57 19 -33.9370 151.1808 VOZ660 VA655 SF34 VH-VEZ 0 9 225 CBR SYD X_adsb_icao 0/0 0
0.60 5 -33.9361 151.1782 RXA139 ZL139 B738 VH-BKH 0 3 191 SYD MEL X_adsb_icao 0/0 0
0.65 323 -33.9372 151.1693 CAR40 null GRND null 0 0 000 null null T_adsb 0/0 0
0.82 355 -33.9325 151.1756 QJE1511 QF1562 B712 VH-YQW 0 0 165 LST SYD X_adsb_icao 0/0 0
0.97 16 -33.9304 151.1824 QLK265 null DH8B VH-TQG 0 2 135 null null X_adsb_icao 0/0 0
1.05 333 -33.9305 151.1676 FOX77 null B77L N853FD 0 0 000 null null X_adsb_icao 0/0 0
1.59 149 -33.9687 151.1933 QFA534 QF534 B738 VH-YYF 0 50 348 SYD BNE X_adsb_icao 0/0 0
3.69 169 -34.0063 151.1915 QTR18U null A388 A7-APE 875 127 347 null null X_adsb_icao 359/23 0
4.81 19 -33.8704 151.2089 SSM1 SSM1 TWR SSM1 0 6 284 null null X_mlat 0/0 0
4.81 19 -33.8704 151.2093 SSM2 SSM2 TWR SSM2 0 4 216 null null X_mlat 0/0 0
23 more flights not shown. Press ctrl-c to exit.
```

Figure 3 Terminal view of live traffic at YSSY