RealTraffic Application Programmable Interface (API)

Readme First 1
API call examples 3
/auth – Authentication 3
POST parameters 3
Example call 3
Response 3
/deauth - Deauthentication 4
POST parameters 4
Example call 4
Response 4
/traffic – Obtaining traffic 4
POST parameters 4
Example call 5
Response 5
Weather system information 8
/nearestmetar - Nearest Airports with METARs 8
POST parameters 8
Example call 9
Response 9
/weather – Obtaining weather 9
POST parameters 9
Example call 10
Response 10

Readme First

You can access traffic and weather data via the RealTraffic API. The responses are given in JSON strings. 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 python example script RTAPI_tester.py for a practical implementation of this.

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 license 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 "wrrl" (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/v3/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/v3/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/v3/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/v3/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/v3/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
  - set this to the string “locationtraffic”
- 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/v3/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, 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 kits (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 unknown.
- 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 obstsacle
- C5 – Line obstacle
- C6-7 – Reserved

- D0 – No information
- D1-7 – Reserved

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/v3/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/v3/nearestmetarcode | gunzip –
```

Response

```
{'ICAO': 'KNSI', 'Dist': 10.3, 'BrgTo': 42.3, 'METAR': "KNSI 130053Z AUTO 28010KT
10SM FEW009 00/ A2986 RMK AO2 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 AO2 SLP114 T0000 $"},
{"ICAO":"KAVX","Dist":804,"BrgTo":34.7,"METAR":"KAVX 130051Z AUTO VRB06KT 10SM CLR
20\/14 A2990 RMK AO2 SLP111 T02000144"},
{"ICAO":"KLPC","Dist":809.6,"BrgTo":25.6,"METAR":"KLPC 130056Z AUTO 27010KT 10SM
CLR 18\/15 A2986 RMK AO2 SLP110 T01830150"}]
}
```

/weather – Obtaining weather

Address: https://rtw.flyrealtraffic.com/v3/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&to
ffset=0" -X POST https://rtw.flyrealtraffic.com/v3/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
AO2 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': []
}
```


The fields are:

- 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.