RealTraffic Application Programmable Interface (API) Documentation

Release 2.5
December 2025

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

Contents

Readme First

RealTraffic (RT) 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.

Therea are two methods to interface with RT data:

- **Direct via API** queries to the RT servers
- **Indirect via the RT application**

Each of these methods is described in this document separately.

When determining which method you want to use, consider the following points:

- Standard RT licenses can retrieve data from the RT servers with a single connection, i.e. no concurrent queries are possible.
- Professional RT licenses get two concurrent connections to the servers.

If you're simply wanting to retrieve traffic and/or weather for a given location, you're better off using the direct API calls to the RT servers.

If you're writing a plugin for MSFS, P3D, X-Plane, or PSX, that works in conjunction with the existing plugins PSXT, LiveTraffic, or the RT application, you should consider using the traffic and weather data broadcast by the RT application instead so you don't use one of the available concurrent server connections.

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/or weather injection:

Software	Supported simulators	Supported features		Platform	Link	Notes
		Traffic	Weather			
RealTraffic client	Aerowinx PSX PSXTraffic clients	Ground Flying	Global WX METARs SIGMETs	Windows MacOS Linux	Visit	
PSXT	MSFS 2020 P3D	Parked Ground Flying	-	Windows	Visit	Standalone version
PSXTraffic	MSFS 2020 P3D	Ground Flying	-	Windows	Visit	Requires RealTraffic client
LiveTraffic	X-plane 11/12	Parked Ground Flying	Global WX METARs	Windows MacOS Linux	Visit	

If you’re a developer and would like your application listed here, please reach out to me via the email on the title page.

Direct API connections

The direct API method give you access to 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. When calling the APIs, **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.

General workflow for direct API connections

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 to fetch traffic and weather:
    - 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 you receive this response, 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 (or license, in the case of auth API) was found. Did you send the correct license string? Typo?

        - 405: Too many sessions. This happens when trying to access RT data with more than one client for a standard license, or more than two clients for a professional license. Also happens if you try to authenticate too quickly after ending a session and that session didn't use deauth to clean up.
        - 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 request rate limits can change dynamically during a session if there is a sudden spike in load on the servers, make sure your code can accommodate that.
        - 407: History too far back. The maximum history supported at present is 14 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.

You can also refer to the Python scripts included with the API documentation to see how these calls are implemented.

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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/deauth

POST parameters

- GUID: the session GUID

Example call

```
curl -sH 'Accept-encoding: gzip' -d "GUID=26cb1a52-169b-425e-8df6-7713e5c34835" -X
POST https://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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,
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 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 obstsacle
- 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 the following format:


```
{
    "data": {
        "7c4920": [-33.936407, 151.169229, "YSSY_101", "A388", "VH-OQA",
1721590016.01, "QFA2", 123],
        "7c5325": [-33.936333, 151.170109, "YSSY_115", "A333", "VH-QPJ",
1721597924.81, "QFA128", 350],
        "7c765a": [-33.935635, 151.17746, "YSSY_D12", "A320", "VH-XNW",
1721650004.0, "JST825", 320]
    },
    "full_count": 13765,
    "source": "DiskDB",
    "rrl": 2000,
    "status": 200,
    "dataepoch": 1721650780
}
```


The fields are latitude, longitude, gate ID with airport ICAO ID prepended, separated by underscore, 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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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': []
}
```


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.
    - Treat like NSC, there may be cloud but it's not operationally relevant.
    - **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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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-30WNW 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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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",
  "rrl": 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://rtwa.flyrealtraffic.com/v5/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://rtwa.flyrealtraffic.com/v5/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 --traffictype parkedtraffic --plot --radius 3
```

This prints the output of the API calls, and shows a plot:

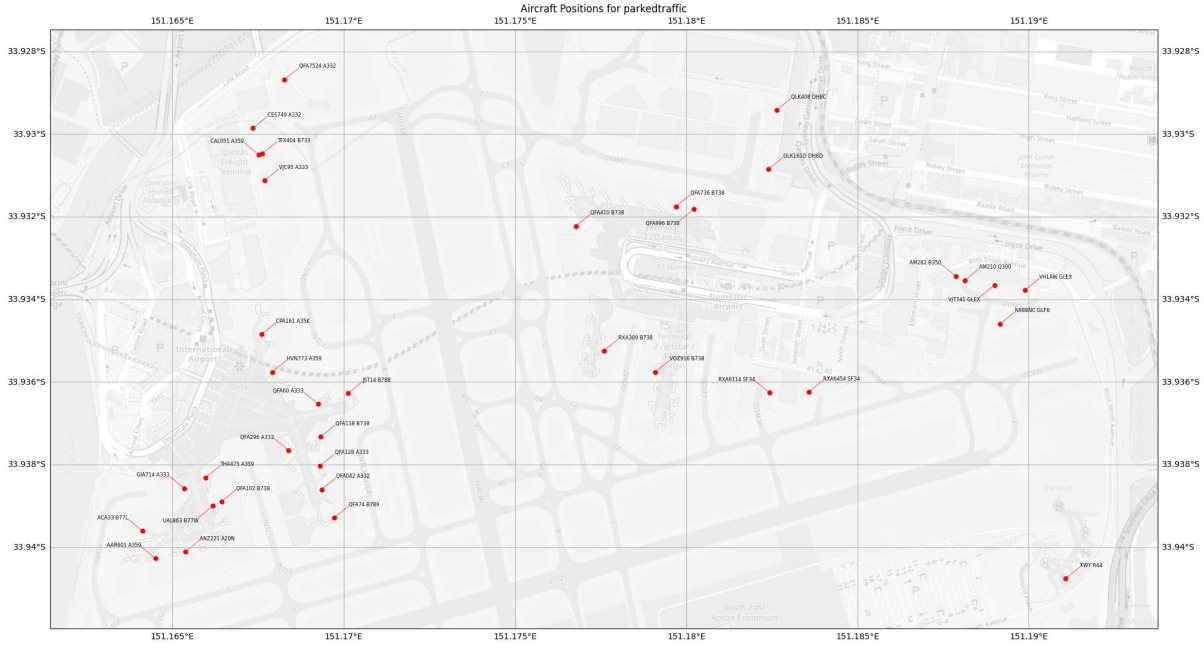


Figure 1 Parked traffic in a 3km radius from the center of YSSY

To save the plot to a file, pass the filename with the --plot parameter: --plot KLAX.jpg

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 --traffictype locationtraffic --plot plot.jpg
```

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

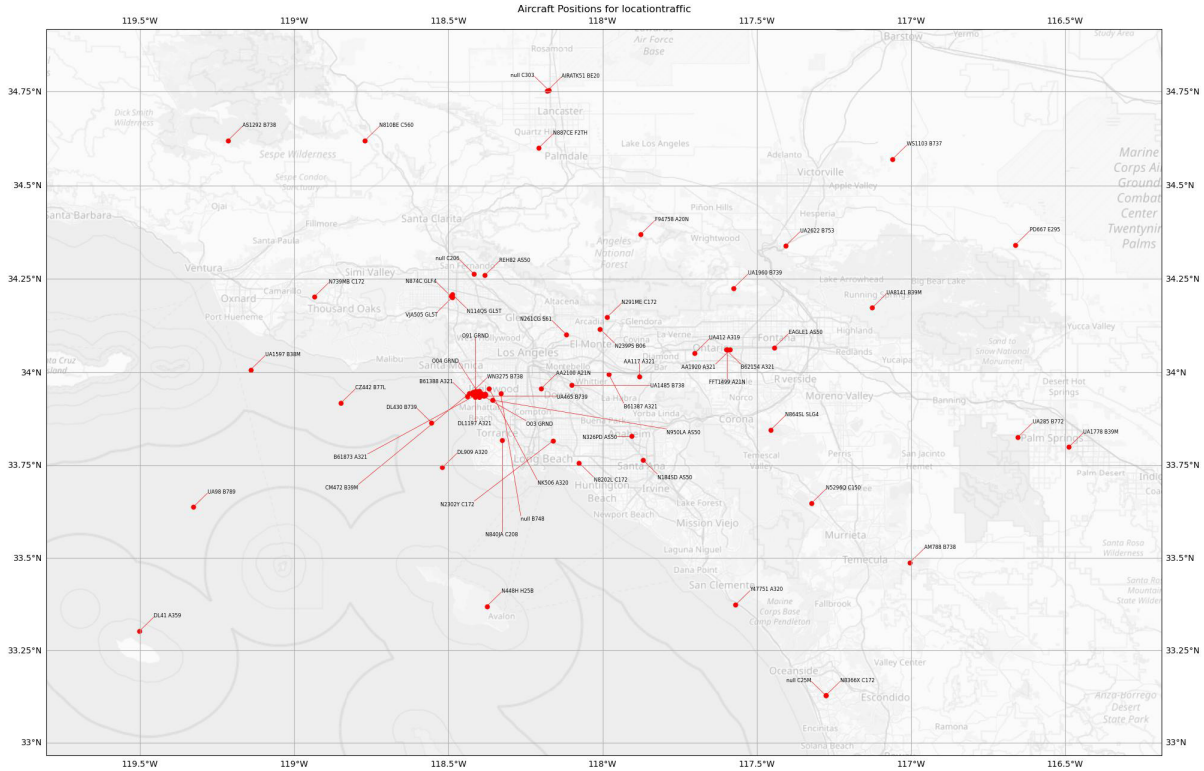


Figure 2 Live airtraffic in a 100km radius around KLAX

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.

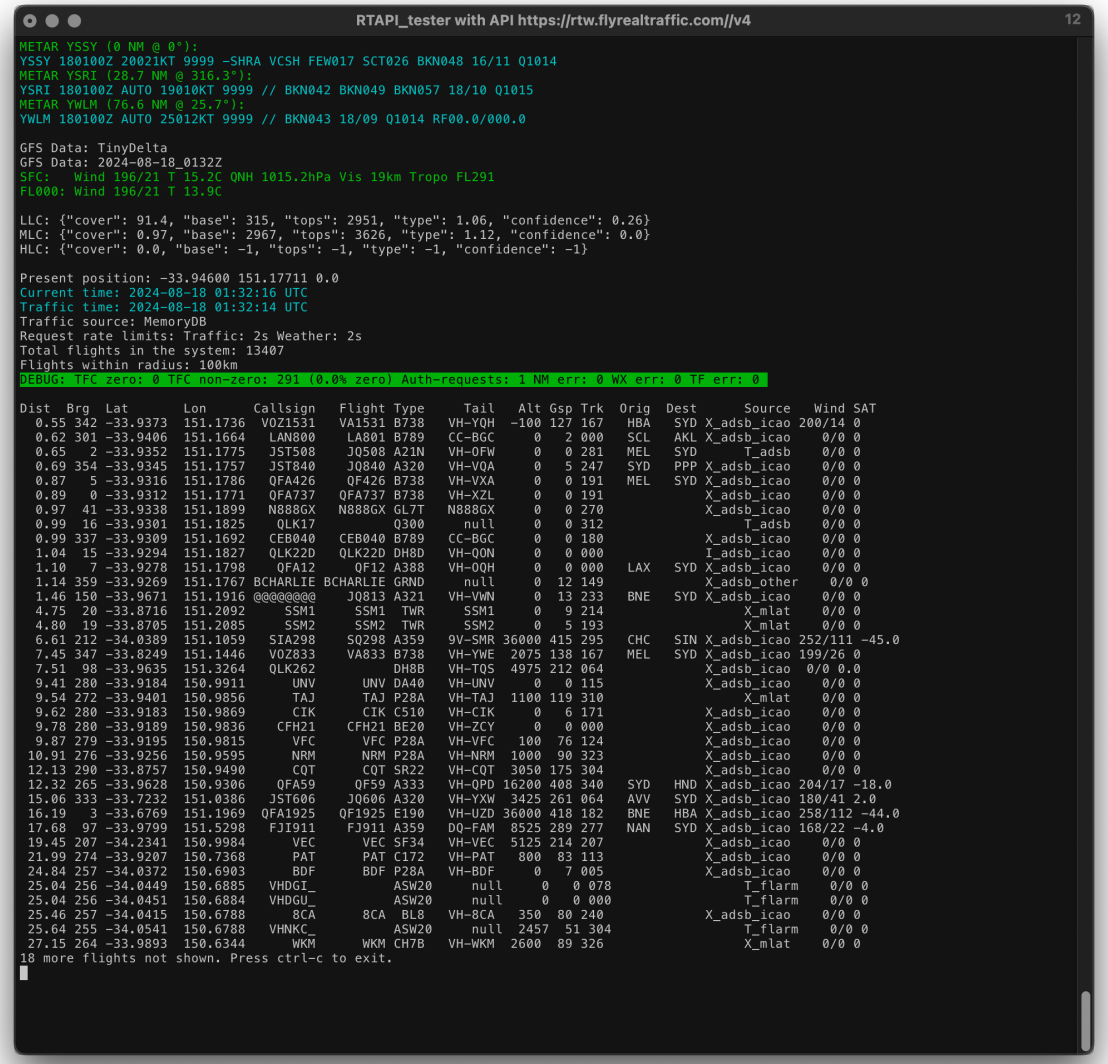


Figure 3 Terminal view of live traffic at YSSY running on MacOS

Indirect API connections via RT application

Since version 10, RT has evolved into both a traffic and weather source. This is very important because flying with real air traffic in the vicinity is only realistic if the weather experienced by the traffic around you is mirrored in the simulator you are flying.

While this has always been supported by RT for landing and departing traffic by providing the correct air pressures and nearest METARs, RT now provides global weather coverage from the crowded skies of Europe to the remotest corners in the polar and oceanic regions.

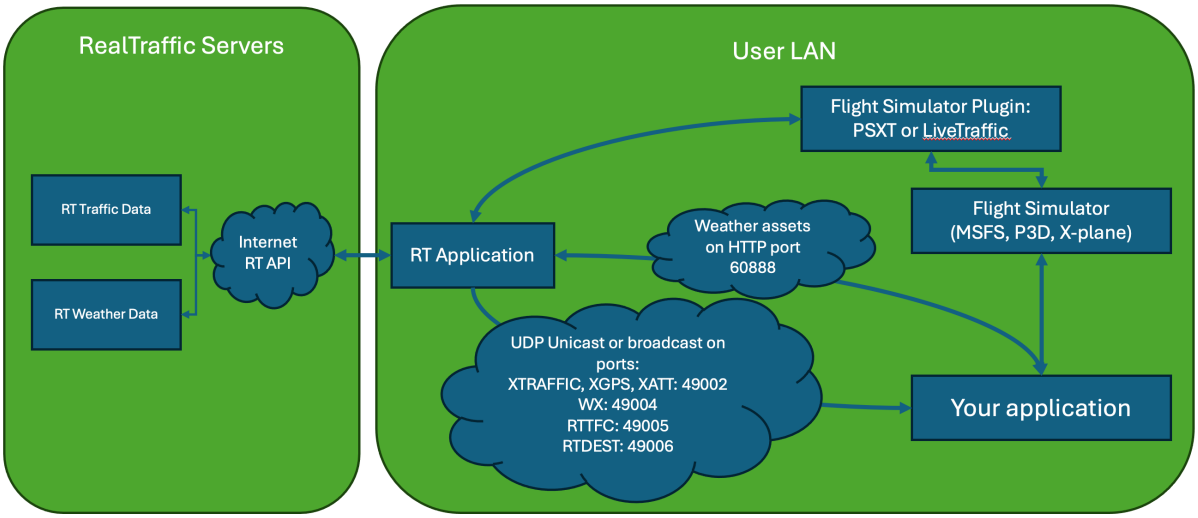
MSFS 2024 unfortunately does not allow weather injection, but they are using global real-time weather from Meteoblue, so when flying in real-time, the weather will match the real-world weather available in RT. When flying with historical offsets however, the weather may not match.

Both X-Plane and PSX fully support weather injection including historical weather. The LiveTraffic plugin used for X-Plane supports the weather messages, and PSX runs with the RT application natively which also fully supports real-time and historical weather.

With satellite derived air traffic now available in RT v10, you no longer are flying on your own during a long-haul flight, and it is even more important that your aircraft experiences the same head- and tail winds as the other traffic, as well as the same air temperatures, so that the Mach number you're flying on the oceanic track system matches that of traffic ahead and behind you – so you don't catch up to them, and they don't catch up to you.

RealTraffic obtains traffic and weather information for the location desired, either by tracking the flight simulator's location (information which you have to provide or is already provided by the plugin in use), or by using the spotter location. If you use the spotter location, you only need to listen for UDP packets.

The following diagram illustrates the data flow when using an existing simulator plugin alongside your own simulator enhancements:



Integrating your own flight simulator plugin

You need to provide your location, and ideally attitude, in regular intervals to the RT application. In both cases, RT will transmit weather and traffic as UDP packets in approximately 2 second intervals, traffic is limited to approximately 100 NM of range around the center of the position provided, and weather is given for the 7 nearest airports providing METARs as well as actual conditions at your present altitude, to include winds and temperatures aloft, turbulence, and cloud information.

Providing your simulator position to RealTraffic

Your simulator plugin needs to provide a TCP socket server connection. By default, RealTraffic expects port 10747 (but this can be changed and is configurable in the Settings panel of the RealTraffic User Interface). Once your plugin detects that RealTraffic has connected to it, you should transmit the following parameters ideally at 5 Hz (i.e. 5 times per second):

- Pitch in radians * 100000
- Bank in radians * 100000
- Heading (or Track) in radians
- Altitude in feet * 1000
- True air speed in meters per second
- Latitude in radians
- Longitude in radians
- All of that preceded by the characters “Qs121=”

An example string would look like this:
“Qs121=6747;289;5.449771266137578;37988724;501908;0.6564195830703577;-2.1443275933742236”

If your simulator is moving slowly, you need to be careful to inject something useful such as true heading instead of track, because track calculations can get a bit iffy with limited precision latitude/longitude points.

For a correct formatting example, look at the following Java code, which translates almost as is to C, just use %0.15f for the float formatting rather than %.15f:

Example code for a correctly formatted string in Java:

```
message = String.format(Locale.US, "Qs121=%d;%d;%.15f;%d;%d;%.15f;%.15f",
(int)(pitch_in_degs * 100000 * d2r), (int)(bank_in_degs * 100000*d2r)*-1,
track_in_degs * d2r, (int)(altitude_in_m * 3028), TAS_in_mps, latitude * d2r,
longitude * d2r);
```

Once you are injecting this data correctly, RealTraffic will spring to life and indicate your position on the map display, as well as show any traffic around you.

If you want the Foreflight and/or Garmin Pilot apps to be supported correctly, you must inject position and attitude updates at 5 Hz.

Using the Destination Traffic/Weather feature

To receive destination traffic and weather on ports 49005/49006, send the following message to RT:

```
Qs376=YSSY;KLAX
```

This would set the destination to KLAX and start broadcasting destination traffic and weather. This feature is only available for professional license holders of RT. The first airport (origin airport usually) is disregarded.

How to find out license and version information

Send the string “Qs999=version” to RT and you’ll receive a JSON string back with the following information:

```
{ "version":"10.0.240","level": 2 }
```

Where version is the current version being run in format major.minor.build, and the

subscription level with 0 = pre-v9-release standard license, 1 = post-v9-release standard license, 2 = professional license.

Controlling the time offset from the simulator

To create a more coherent simulation experience, the time offset for RealTraffic can be controlled from the simulator. Simulator time offset control can be enabled/disabled both in the RT GUI and from the simulator itself.

To inquire who's in control, send this string: “Qs999=timeoffsetcontrol=?”

you'll either receive
```
{ "timeoffsetcontrol": "Simulator" }
```
or
```
{ "timeoffsetcontrol": "RealTraffic" }
```

and you can send this to control it “Qs999=timeoffsetcontrol=Simulator” or “Qs999=timeoffsetcontrol=RealTraffic”.
To set the time to be used in RT, send the UTC time formatted as epoch milliseconds with a Qs123 message as follows “Qs123=1674956315329”.

You can verify in the RT GUI that the correct time offset is running. Note that this only works for professional licenses, you can enable/disable the time offset feature in your plugin by first checking what type of license the user has, see “How to find out license and version information” above for that.

Reading weather and traffic information

RealTraffic broadcasts all traffic and weather information via UDP on the local network, but can provide much more detailed weather via the TCP connection you already are connected to. The broadcast weather information pertains to the nearest 7 airports. ADS-B altitudes for traffic are already pressure corrected. You can use the altitudes as they are reported by RT and inject the traffic into your simulator using those altitudes.

UDP Weather Broadcasts

The weather messages are broadcast as UDP packets once every 10 seconds on port 49004 containing a JSON string with the nearest airport data along with data that pertains to the **current location and altitude** of the simulated aircraft. This allows you to inject the real weather experienced by your aircraft in the simulator, so that you fly in the same winds aloft and with the same engine performance as the real traffic surrounding you.
```
{
"ICAO":"KMCE",
"QNH":1019.7,
"METAR": "KMCE 130453Z AUTO 14005KT 6SM -RA BR OVC043 13/12
A3011 RMK AO2 RAB0354E08B11E21B37 SLP197 P0003 T01330117",
"TA":18000,
"locWX": {
"Info": "2023-03-13_0558Z",
"ST":11.4,
```

"SWSPD":9.59,
"SWDIR":146.29,
"SVis":18224,
"LLCC":17,
"MLCC":49.7,
"HLCC":0,
"DZDT":-0.0302,
"PRR":0.55,
"T":-52.57,
"WDIR":268.4
"WSPD":137.27,
"TPP":10084.52,
"SLP":1019.73,
"Profiles":"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 ^^",
},
"AM": [
"KMER 130535Z AUTO 17007KT 10SM SCT028 OVC035 13/13 A3012 RMK
AO1",
"KCVH 130535Z AUTO 00000KT 10SM OVC012 13/12 A3013 RMK A01",
"KNLC 130456Z AUTO 13007KT 10SM FEW090 13/11 A3011 RMK AO2
SLP196 T01330111",
"KFAT 130453Z 10008KT 6SM RA BKN060 OVC070 13/10 A3012 RMK AO2
SLP196 P0009 T01330100",
"KSNS 130453Z AUTO 01008KT 10SM SCT045 14/12 A3010 RMK AO2
SLP202 T01390117",
"KE16 130535Z AUTO 17003KT 2 1/2SM BR FEW008 OVC015 13/12
A3013 RMK AO2 PWINO PNO"]
}

The fields are:

- **ICAO:** the ICAO code of the nearest airport,
- **QNH:** the reported pressure in hPa, often to within 0.1 hPa precision,
- **METAR:** contains the full METAR received,
- **TA:** the transition altitude in ft
- **locWX:** is the location weather at the present position, 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.
    - If no data is present, 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
- **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 cloud cover in percent (lowest third of troposphere)
- **MLCC**: medium level cloud cover in percent (medium third of troposphere)
- **HLCC**: high level cloud cover in percent (top third of troposphere)
- **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
- **WDIR**: wind direction in degrees
- **WSPD**: wind speed in km/h
- **TPP**: tropopause height in meters
- **SLP**: sea level pressure in hPa
- **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 Winddirection/Windspeed and OAT in C for each of those altitudes.
- **AM**: additional metars. Contains a list of an additional 6 nearest METARs in a forward looking direction (if available). You can use these METARs to gain additional understanding of the weather surrounding the aircraft.

The traffic data is broadcast as UDP packets, and these come in several formats:

- XTRAFFICPSX (Foreflight format) which is broadcast on port 49002,

- RTTFC (RT Traffic format), broadcast on port 49005
- RTDEST (destination Traffic), broadcast on port 49006

Further to this, the simulator position is re-distributed (for the benefit of other third party apps) as **XGPS** and **XATT** messages on the same port as Foreflight messages, 49002.

XTRAFFIC format

XTRAFFIC is the native Foreflight format. Each traffic in your area will be broadcast in a string as follows:

"XTRAFFICPSX,hexid,lat,lon,alt,vs,airborne,hdg,spd,cs,type"

where the parameters are:

- Hexid: the hexadecimal ID of the transponder of the aircraft. This is a unique ID, and you can use this ID to track individual aircraft.
- Lat: latitude in degrees
- Lon: longitude in degrees
- Alt: altitude in feet
- Vs: vertical speed in ft/min
- Airborne: 1 or 0
- Hdg: The heading of the aircraft (it's actually the true track, strictly speaking.)
- Spd: The speed of the aircraft in knots
- Cs: the ICAO callsign (Emirates 413 = UAE413 in ICAO speak, = EK413 in IATA speak)
- Type: the ICAO type of the aircraft, e.g. A388 for Airbus 380-800. B789 for Boeing 787-9 etc.

The GPS and Attitude messages are formatted as follows:

"XGPSPSX,lon,lat,alt,track,gsp"

Where the parameters are:

- Lon: Longitude in degrees (remember west = negative)
- Lat: Latitude in degrees (north = positive)
- Alt: Altitude in meters
- Track: True track
- Gsp: Ground speed in meters per second

"XATTPSX,hdg,pitch,roll"

Where the parameters are:

- Hdg: The heading of the aircraft (true heading)
- Pitch: The pitch in degrees, positive = pitch up
- Roll: The roll angle in degrees, positive = right roll

RTTFC/RTDEST format

This format contains almost all information we have available from our data sources. The

format is:

"RTTFC,hexid, lat, lon, baro_alt, baro_rate, gnd, track, gsp, cs_icao, ac_type, ac_tailno, from_iata, to_iata, timestamp, source, cs_iata, msg_type, alt_geom, IAS, TAS, Mach, track_rate, roll, mag_heading, true_heading, geom_rate, emergency, category, nav_qnh, nav_altitude_mcp, nav_altitude_fms, nav_heading, nav_modes, seen, rssi, winddir, windspd, OAT, TAT, isICAOhex,augmentation_status,authentication"

If you split this string, the following 0 based indices apply:

Index	Data content
0	RTTFC = data format descriptor
1	hexid = the transponder's unique hexadecimal ID
2	lat = latitude
3	lon = longitude
4	baro_alt = barometric altitude
5	baro_rate = barometric vertical rate
6	gnd = ground flag
7	track = track
8	gsp = ground speed
9	cs_icao = ICAO call sign, the actual callsign a flight is known as to air traffic services
10	ac_type = aircraft type
11	ac_tailno = aircraft registration
12	from_iata = origin IATA code
13	to_iata = destination IATA code
14	timestamp = unix epoch timestamp when data was last updated
15	source = data source
16	cs_iata = IATA call sign (the flight number you would see on an airport arrival/departure announcement board
17	msg_type = type of message
18	alt_geom = geometric altitude (WGS84 GPS altitude)
19	IAS = indicated air speed
20	TAS = true air speed
21	Mach = Mach number
22	track_rate = rate of change for track
23	roll = roll in degrees, negative = left
24	mag_heading = magnetic heading
25	true_heading = true heading
26	geom_rate = geometric vertical rate
27	emergency = emergency status
28	category = category of the aircraft
29	nav_qnh = QNH setting in MCP/FCU
30	nav_altitude_mcp = altitude dialled into the MCP/FCU in the flight deck
31	nav_altitude_fms = altitude set by the flight management system (FMS)
32	nav_heading = heading set by the MCP
33	nav_modes = which modes the autopilot is currently in

34	seen = seconds since any message updated this aircraft state vector
35	rssi = signal strength at the ADS-B receiver that provided the data
36	winddir = wind direction in degrees true north
37	windspd = wind speed in kts
38	OAT = outside air temperature / static air temperature
39	TAT = total air temperature
40	isICAOhex = is this hexid an ICAO assigned ID
41	Augmentation_status = has this record been augmented from multiple sources
42	Authentication = authentication status of the license, safe to ignore

The “source” field can contain any of the following values:

- adsb: a reduced data ADS-B field, only providing location, altitude, track, and speed.
- adsb_icao: messages from a Mode S or ADS-B transponder often (but not always) with the full complement of data fields.
- adsb_icao_nt: messages from an ADS-B equipped "non-transponder" emitter e.g. a ground vehicle.
- adsr_icao: rebroadcast of an ADS-B messages originally sent via another data link
- tisb_icao: traffic information about a non-ADS-B target identified by a 24-bit ICAO address, e.g. a Mode S target tracked by SSR.
- 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. Use caution.
- mode_s: ModeS data only, no position.
- adsb_other: using an anonymised ICAO address. Rare.
- adsr_other: rebroadcast of ‘adsb_other’ ADS-B messages.
- est: These are estimated positions. Some of our data providers create estimates based on historical flights, and add that to their data feed.

All the source fields are preceded by ?_ where ? can be any letter of the alphabet, identifying the data source that provided the data in use. This helps debugging traffic issues, should they arise.

If the fields are preceded by ID_, that means they are based on interpolated data (only available in historical data feeds). Interpolated data is generated on the servers when satellite positions are received, or if an aircraft has flown out and back into ADS-B coverage within a timespan of maximum 2h. The server backend then interpolates the intermediate positions in 2s intervals to backfill the positions, allowing users a seamless flying experience in oceanic and off-shore places. Note that these are NOT estimated positions, but positions between known positions of the aircraft. Data is interpolated all the time, but only up to a maximum of 2h in the past. This is because the satellite derived data can sometimes have considerable delay and depends on the time it takes for the satellite to reach a downlink station. You can expect to see about 87% of air traffic recovered within a 2h timespan.

Here is an example RTTFC record:
RTTFC,11234042,-33.9107,152.9902,26400,1248,0,90.12,490.00,AAL72,B789,
N835AN,SYD,LAX,1645144774.2,X2,AA72,X_adsb_icao,27575,320,474,0.780,

0.0,0.0,78.93,92.27,1280,none,A5,1012.8,35008,-1,71.02,
autopilot|vnav|lnav|tcas,0.0,-21.9,223,24,-30,0,1,170124
RTTFC,10750303,-33.7964,152.3938,20375,1376,0,66.77,484.30,UAL842,B789,
N35953,SYD,LAX,1645144889.8,X2,UA842,F_adsb_icao,21350,343,466,0.744,-0.0,
0.5,54.49,67.59,1280,none,A5,1012.8,35008,-1,54.84,
autopilot|vnav|lnav|tcas,0.0,-20.8,227,19,-15,14,1,268697

Using Satellite Imagery

As of RT version 8, global real-time satellite coverage is available. These data can be accessed by creating an HTTP connection to the host where RT is running on port 60888.

Connecting to RT on port 60888 will give you the following JSON object:

```
{ "timestamp": 1766530200, "lower_left_latitude": -40, "lower_left_longitude": 140,
"product": "TC" }
```

This gives you the timestamp of when the image was taken, as well as the lower left coordinates of the image, and the image type. Images types are:

- TC: True color (with IR merged for nighttime)
- B13: Channel 13 (thermal infrared)
- PWV: precipitable water vapour
- RDR: an artificial radar image

Launch another HTTP request to the /data endpoint and you'll receive the 1600x1600 pixels jpg image.

RT will serve whichever image the user has selected. But the one image always available is at the endpoint /RDW which provides an estimated radar reflectivity map.

The geostationary satellites used to create the imagery are

- GOES17 at 137.2 degrees West
- GOES16 at 75.2 degrees West
- Meteosat 10 at 0 degrees
- Meteosat 8 at 45 degrees East
- Himawari 9 at 140.7 degrees East

Meteosat updates every 15 minutes, the others every 10 minutes. Best resolution for Meteosat is 1km per pixel, for the others 500m per pixel.

Processing time for each satellite is approximately 8 minutes, therefore by the time the data is available, the lag behind real-time is between 8 and 18 minutes depending on the scan time. GOES16 and 17 as well as the Himawari 8 satellites don't take a single picture like a camera, but rather scan the earth in 10 horizontal scan lines, each taking about 45 seconds to complete. The Meteosat satellites are a fair bit older and use a different method of scanning the earth. The entire satellite is rotating, and multiple scan lines are taking data as the earth zips by the apertures.

The format of the images is in high quality jpg encoding. Resolution is 1600 x 1600 pixel for

each tile (equivalent to the native resolution available at the sub-satellite point), each spanning 10 x 10 degrees in a Mercator projection. This makes it relatively easy to derive the position of each pixel in simulator space.

True color image

This is a color enhanced rendition of the earth with all clouds. Native resolution is 500m per pixel at the sub-satellite point (less the further away one goes). This image is available from RT if it is selected as the currently visible image.

False colour infrared

This is the 10.4 micrometer channel in the thermal infrared band. It is coloured such that cloud top temperatures of -40C and less are rendered in colour, making it particularly useful to identify regions of convective activity, e.g. thunderstorms.

PWV – Precipitable Water Vapour

This image is a false colour composite comprised of three water vapour channels: Blue represents water vapour at about 9km altitude, green at 5km altitude, and red at 3km altitude. This image is useful to identify Jetstream locations, as well as areas of clear air turbulence (where Jetstream collide or make sharp turns).

RDR – Radar image

This is a product created to estimate radar returns as they would be shown on an on-board radar system.

RDW – Raw radar image

This is the image the radar image is based on, but is in grayscale, with full white indicating strongest returns. This allows a plugin developer to implement gain control in the on-board radar. This image is not available for display in the GUI, but it is available from the RT web interface regardless of which satellite overlay is selected by the user.
10How to access the data