RealTraffic Developer Documentation v 1.3

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 by using the spotter location. If you use the spotter location, you only need to listen for UDP packets. If you want to integrate it into a simulator application however, you need to provide your location in regular intervals to RealTraffic.

In both cases, RealTraffic will transmit weather and traffic as UDP packets in approximately 10 second intervals, traffic is limited to approximately 100 NM of range around the center of the position provided.

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 pane of the RealTraffic User Interface). Once your plugin detects that RealTraffic has connected to it, you need to transmit the following parameters at between 1 - 5 Hz (i.e. 1 – 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, as often 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 in the map display, as well as show any traffic around you.

Reading weather and traffic information

RealTraffic broadcasts all traffic and weather information via UDP. Weather information pertains to the nearest airport, and is mostly of value to plugins who want to position traffic at the correct altitude. ADS-B altitude is always reported in reference to the standard atmospheric pressure of 1013 hPa. Therefore, if local pressure is different from 1013 hPa, you have to correct the altitude of the traffic (approximately 30ft of altitude per hPa of pressure difference). Assuming I remember my barometry correctly from flight school: If the local pressure is 1023 hPa and ground is at sea level, an airplane indicating ADS-B altitude of 10'000ft would in reality be at 10'300ft, while with a local pressure of 1003 hPa the same airplane would be flying at 9'700ft AGL (over the ground).

The weather messages are broadcast as UDP packets once every 10 seconds on port 49004 containing a JSON string with the following format:

"{"ICAO": "XXXX","QNH": 1013, "METAR": "XXXX", "NAME": "XXXX", "IATA": "XXXX" , "DISTNM": 0}";

The fields should be self-explanatory, in case they're not:

- ICAO is the ICAO code of the nearest airport,
- QNH is the reported pressure in hPa,
- METAR contains the full METAR received,
- NAME shows the airport name (usually long),
- IATA is the IATA code of the airport (YSSY = Sydney in ICAO speak, SYD = Sydney in IATA speak), and lastly,
- DISTNM is the distance to said airport in nautical miles.

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

- Foreflight (an electronic flight bag app) format which is broadcast on port 49002,
- AITraffic format, broadcast on port 49003

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.

Foreflight format

In foreflight format, each traffic in your area will be broadcast in a string with the following format:

"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 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

AITraffic format

AI Traffic format is similar to Foreflight, but has additional parameters (the last 3 parameters, marked in bold below):

"AITFC,hexid,lat,lon,alt,vs,airborne,hdg,spd,cs,type,**tail,from,to,timestamp**"

For the first 10 parameters see description above, the last three parameters are:

- Tail: The registration number of the aircraft
- From: The origin airport where known (in IATA or ICAO code)
- To: The destination airport where known (in IATA or ICAO code)
- Timestamp: The UNIX epoch timestamp when this position was valid

Using Satellite Imagery

As of version 8 (currently in public beta), global real-time satellite coverage is available. Of particular interest for developers are the RDW (Radar raw) data, the true color image, as well as the cloud top data.

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 8 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. These 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 format of the images is in png so as to provide the best quality. Resolution is 3200 x 3200 pixel for each tile (equivalent to the native resolution available at the sub-satellite point), each spanning 20 x 20 degrees in a Mercator projection. This makes it relatively easy to derive the position of each pixel in simulator space.

If you would like any products not already available, please inquire and I can make them available as all data is derived from the satellite's raw data feeds.

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 infrared. 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.

How to access the data

RealTraffic has a web interface at port 60888. Connect a web-browser to this port and you can download the data products (or any app you might write is able to talk to RT on that port):

- Launch a web browser and navigate to the host where RT is running on, usually that's localhost: http://localhost:60888
- This will show you a JSON string with the UTC timestamp (in epoch seconds) of the currently shown image, it also shows which image is being shown, as well as the lower left corner latitude and longitude of the image. Each image is 20 x 20 degrees (mercator projected).
- navigate to http://localhost:60888/data and the currently used image in the GUI is shown in the browser.
- navigate to http://localhost:60888/rdw and a satellite derived radar reflectivity map is shown. This could be used to create a realistic radar display in the cockpit. it's a grayscale image that can be used to simulate a radar return based on the whiteness of the pixels shown.