OpenSky REST API

The root URL of our REST API is:

```
https://opensky-network.org/api
```

There are several functions available to retrieve state vectors, flights and tracks for the whole network, a particular sensor, or a particular aircraft. Note that the functions to retrieve state vectors of sensors other than your own are rate limited (see Limitations).

All State Vectors

The following API call can be used to retrieve any state vector of the OpenSky. Please note that rate limits apply for this call (see Limitations). For API calls without rate limitation, see Own State Vectors.

Operation

`GET /states/all`

Request

You can (optionally) request state vectors for particular airplanes or times using the following request parameters:

Property	Type	Description
time	integer	The time in seconds since epoch (Unix time stamp to retrieve states for. Current time will be used if omitted.
icao24	string	One or more ICAO24 transponder addresses represented by a hex string (e.g. *abc9f3*). To filter multiple ICAO24 append the property once for each address. If omitted, the state vectors of all aircraft are returned.

In addition to that, it is possible to query a certain area defined by a bounding box of WGS84 coordinates. For this purpose, add all of the following parameters:

Property	Type	Description
lamin	float	lower bound for the latitude in decimal degrees
lomin	float	lower bound for the longitude in decimal degrees
lamax	float	upper bound for the latitude in decimal degrees
lomax	float	upper bound for the longitude in decimal degrees

Lastly, you can request the category of aircraft by adding the following request parameter:

Property	Type	Description
extended	integer	Set to 1 if required

Example query with time and aircraft: `https://opensky-network.org/api/states/all?time=1458564121&icao24=3c6444`

Example query with bounding box covering Switzerland: `https://opensky-network.org/api/states/all?lamin=45.8389&lomin=5.9962&lamax=47.8229&lomax=10.5226`

Response

The response is a JSON object with the following properties

Property	Type	Description
time	integer	The time which the state vectors in this response are associated with. All vectors represent the state of a vehicle with the interval $[time - 1, time]$.
states	array	The state vectors.

The *states* property is a two-dimensional array. Each row represents a state vector and contains the following fields:

Index	Property	Type	Description
0	*icao24*	string	Unique ICAO 24-bit address of the transponder in hex string representation.
1	*callsign*	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.
2	*origin_country*	string	Country name inferred from the ICAO 24-bit address.
3	*time_position*	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.

Index	Property	Type	Description
4	*last_contact*	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.
5	*longitude*	float	WGS-84 longitude in decimal degrees. Can be null.
6	*latitude*	float	WGS-84 latitude in decimal degrees. Can be null.
7	*baro_altitude*	float	Barometric altitude in meters. Can be null.
8	*on_ground*	boolean	Boolean value which indicates if the position was retrieved from a surface position report.
9	*velocity*	float	Velocity over ground in m/s. Can be null.
10	*true_track*	float	True track in decimal degrees clockwise from north (north=0°). Can be null.
11	*vertical_rate*	float	Vertical rate in m/s. A positive value indicates that the airplane is climbing, a negative value indicates that it descends. Can be null.
12	*sensors*	int[]	IDs of the receivers which contributed to this state vector. Is null if no filtering for sensor was used in the request.
13	*geo_altitude*	float	Geometric altitude in meters. Can be null.
14	*squawk*	string	The transponder code aka Squawk. Can be null.
15	*spi*	boolean	Whether flight status indicates special purpose indicator.
16	*position_source*	int	Origin of this state's position. • 0 = ADS-B • 1 = ASTERIX • 2 = MLAT • 3 = FLARM

Index	Property	Type	Description
17	*category*	int	Aircraft category. • 0 = No information at all • 1 = No ADS-B Emitter Category Information • 2 = Light (< 15500 lbs) • 3 = Small (15500 to 75000 lbs) • 4 = Large (75000 to 300000 lbs) • 5 = High Vortex Large (aircraft such as B-757) • 6 = Heavy (> 300000 lbs) • 7 = High Performance (> 5g acceleration and 400 kts) • 8 = Rotorcraft • 9 = Glider / sailplane • 10 = Lighter-than-air • 11 = Parachutist / Skydiver • 12 = Ultralight / hang-glider / paraglider • 13 = Reserved • 14 = Unmanned Aerial Vehicle • 15 = Space / Trans-atmospheric vehicle • 16 = Surface Vehicle – Emergency Vehicle • 17 = Surface Vehicle – Service Vehicle • 18 = Point Obstacle (includes tethered balloons) • 19 = Cluster Obstacle • 20 = Line Obstacle

Limitations

Limitiations for anonymous (unauthenticated) users

Anonymous are those users who access the API without using credentials. The limitations for anonymous users are:

- Anonymous users can only get the most recent state vectors, i.e. the *time* parameter will be ignored.
- Anonymous users can only retrieve data with a time resolution of 10 seconds. That means, the API will return state vectors for time $now - (now \ mod \ 10)$.
- Anonymous users get 400 API credits per day (see credit usage below).

Limitations for OpenSky users

An OpenSky user is anybody who uses a valid OpenSky account (see below) to access the API. The rate limitations for OpenSky users are:

- OpenSky users can retrieve data of up to 1 hour in the past. If the *time* parameter has a value $t < now - 3600$ the API will return *400 Bad Request*.

- OpenSky users can retrieve data with a time resolution of 5 seconds. That means, if the *time* parameter was set to t, the API will return state vectors for time $t - (t \ mod \ 5)$.
- OpenSky users get 4000 API credits per day. For higher request loads please contact OpenSky.
- Active contributing OpenSky users get a total of 8000 API credits per day. An active user is a user which has an ADS-B receiver that is at least 30% online (measured over the current month).

Note

You can retrieve all state vectors received by your receivers without any restrictions. See Own State Vectors. Before the request limit is reached the header *X-Rate-Limit-Remaining* indicates the amount of remaining credits. After the rate limit is reached the status code *429 - Too Many Requests* is returned and the header *X-Rate-Limit-Retry-After-Seconds* indicates how many seconds until credits/request become available again.

API credit usage

API credits are only used for the /states/all API endpoint. Credit usage is lower in general for restricted/smaller areas. The area can be restricted by using the *lamin*, *lamax*, *lomin*, *lomax* query parameters. The *area square deg* column in the table below, indicates the square degree limit- e.g. a box extending over latitude 10 degress and longitude 5 degrees, would equal 50 square degrees:

Area square deg	Credits	Example
0 - 25 (<500x500km)	1	/api/states/all? lamin=49.7&lamax=50.5&lomin=3.2&lomax=4.6
25 - 100 (<1000x1000km)	2	/api/states/all? lamin=46.5&lamax=49.9&lomin=-1.4&lomax=6.8
100 - 400 (<2000x2000km)	3	/api/states/all? lamin=42.2&lamax=49.8&lomin=-4.7&lomax=10.9
over 400 or all (>2000x2000km)	4	/api/states/all

Examples

Retrieve all states as an anonymous user:

```
$ curl -s "https://opensky-network.org/api/states/all" | python -m json.tool
```

Retrieve all states as an authenticated OpenSky user:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/states/all" | python -m
json.tool
```

Retrieve states of two particular airplanes:

```
$ curl -s "https://opensky-network.org/api/states/all?icao24=3c6444&icao24=3e1bf9" | python -m
json.tool
```

Own State Vectors

The following API call can be used to retrieve state vectors for your own sensors without rate limitations. Note that authentication is required for this operation, otherwise you will get a *403 - Forbidden*.

Operation

`GET /states/own`

Request

Pass one of the following (optional) properties as request parameters to the *GET* request.

Property	Type	Description
time	integer	The time in seconds since epoch (Unix timestamp to retrieve states for. Current time will be used if omitted.
icao24	string	One or more ICAO24 transponder addresses represented by a hex string (e.g. *abc9f3*). To filter multiple ICAO24 append the property once for each address. If omitted, the state vectors of all aircraft are returned.
serials	integer	Retrieve only states of a subset of your receivers. You can pass this argument several time to filter state of more than one of your receivers. In this case, the API returns all states of aircraft that are visible to at least one of the given receivers.

Response

The response is a JSON object with the following properties

Property	Type	Description
time	integer	The time which the state vectors in this response are associated with. All vectors represent the state of a vehicle with the interval $[time - 1, time]$.
states	array	The state vectors.

The *states* property is a two-dimensional array. Each row represents a state vector and contains the following fields:

Index	Property	Type	Description
0	*icao24*	string	Unique ICAO 24-bit address of the transponder in hex string representation.
1	*callsign*	string	Callsign of the vehicle (8 chars). Can be null if no callsign has been received.
2	*origin_country*	string	Country name inferred from the ICAO 24-bit address.
3	*time_position*	int	Unix timestamp (seconds) for the last position update. Can be null if no position report was received by OpenSky within the past 15s.
4	*last_contact*	int	Unix timestamp (seconds) for the last update in general. This field is updated for any new, valid message received from the transponder.
5	*longitude*	float	WGS-84 longitude in decimal degrees. Can be null.
6	*latitude*	float	WGS-84 latitude in decimal degrees. Can be null.
7	*baro_altitude*	float	Barometric altitude in meters. Can be null.
8	*on_ground*	boolean	Boolean value which indicates if the position was retrieved from a surface position report.
9	*velocity*	float	Velocity over ground in m/s. Can be null.
10	*true_track*	float	True track in decimal degrees clockwise from north (north=0°). Can be null.
11	*vertical_rate*	float	Vertical rate in m/s. A positive value indicates that the airplane is climbing, a negative value indicates that it descends. Can be null.
12	*sensors*	int[]	IDs of the receivers which contributed to this state vector. Is null if no filtering for sensor was used in the request.
13	*geo_altitude*	float	Geometric altitude in meters. Can be null.
14	*squawk*	string	The transponder code aka Squawk. Can be null.
15	*spi*	boolean	Whether flight status indicates special purpose indicator.

Index	Property	Type	Description
16	*position_source*	int	Origin of this state's position. • 0 = ADS-B • 1 = ASTERIX • 2 = MLAT • 3 = FLARM
17	*category*	int	Aircraft category. • 0 = No information at all • 1 = No ADS-B Emitter Category Information • 2 = Light (< 15500 lbs) • 3 = Small (15500 to 75000 lbs) • 4 = Large (75000 to 300000 lbs) • 5 = High Vortex Large (aircraft such as B-757) • 6 = Heavy (> 300000 lbs) • 7 = High Performance (> 5g acceleration and 400 kts) • 8 = Rotorcraft • 9 = Glider / sailplane • 10 = Lighter-than-air • 11 = Parachutist / Skydiver • 12 = Ultralight / hang-glider / paraglider • 13 = Reserved • 14 = Unmanned Aerial Vehicle • 15 = Space / Trans-atmospheric vehicle • 16 = Surface Vehicle – Emergency Vehicle • 17 = Surface Vehicle – Service Vehicle • 18 = Point Obstacle (includes tethered balloons) • 19 = Cluster Obstacle • 20 = Line Obstacle

Examples

Retrieve states for all sensors that belong to you:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/states/own" | python -m
json.tool
```

Retrieve states as seen by a specific sensor with serial *123456*

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/states/own?serials=123456" |
python -m json.tool
```

Retrieve states for several receivers:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/states/own?
serials=123456&serials=98765" | python -m json.tool
```

Flights in Time Interval

This API call retrieves flights for a certain time interval [begin, end]. If no flights are found for the given time period, HTTP status *404 - Not found* is returned with an empty response body.

Operation

`GET /flights/all`

Request

These are the required request parameters:

Property	Type	Description
begin	integer	Start of time interval to retrieve flights for as Unix time (seconds since epoch)
end	integer	End of time interval to retrieve flights for as Unix time (seconds since epoch)

The given time interval must not be larger than two hours!

Response

The response is a JSON array of flights where each flight is an object with the following properties:

Examples

Get flights from 12pm to 1pm on Jan 29 2018:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/flights/all?
begin=1517227200&end=1517230800" | python -m json.tool
```

Flights by Aircraft

This API call retrieves flights for a particular aircraft within a certain time interval. Resulting flights departed and arrived within [begin, end]. If no flights are found for the given period, HTTP stats *404 - Not found* is returned with an empty response body.

> **Note**
>
> Flights are updated by a batch process at night, i.e., only flights from the previous day or earlier are available using this endpoint.

Operation

`GET /flights/aircraft`

Request

These are the required request parameters:

Property	Type	Description
icao24	string	Unique ICAO 24-bit address of the transponder in hex string representation. All letters need to be lower case
begin	integer	Start of time interval to retrieve flights for as Unix time (seconds since epoch)
end	integer	End of time interval to retrieve flights for as Unix time (seconds since epoch)

The given time interval must not be larger than 30 days!

Response

The response is a JSON array of flights where each flight is an object with the following properties:

Examples

Get flights for D-AIZZ (3c675a) on Jan 29 2018:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/flights/aircraft?
icao24=3c675a&begin=1517184000&end=1517270400" | python -m json.tool
```

Arrivals by Airport

Retrieve flights for a certain airport which arrived within a given time interval [begin, end]. If no flights are found for the given period, HTTP stats *404 - Not found* is returned with an empty response body.

> **Note**
>
> Similar to flights, arrivals are updated by a batch process at night, i.e., only arrivals from the previous day or earlier are available using this endpoint.

Operation

`GET /flights/arrival`

Request

These are the required request parameters:

Property	Type	Description
airport	string	ICAO identier for the airport
begin	integer	Start of time interval to retrieve flights for as Unix time (seconds since epoch)
end	integer	End of time interval to retrieve flights for as Unix time (seconds since epoch)

The given time interval must not be larger than seven days!

Response

The response is a JSON array of flights where each flight is an object with the following properties:

Examples

Get all flights arriving at Frankfurt International Airport (EDDF) from 12pm to 1pm on Jan 29 2018:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/flights/arrival?
airport=EDDF&begin=1517227200&end=1517230800" | python -m json.tool
```

Departures by Airport

Retrieve flights for a certain airport which departed within a given time interval [begin, end]. If no flights are found for the given period, HTTP stats *404 - Not found* is returned with an empty response body.

Operation

`GET /flights/departure`

Request

These are the required request parameters:

Property	Type	Description
airport	string	ICAO identier for the airport (usually upper case)
begin	integer	Start of time interval to retrieve flights for as Unix time (seconds since epoch)
end	integer	End of time interval to retrieve flights for as Unix time (seconds since epoch)

The given time interval must not be larger than seven days!

Response

The response is a JSON array of flights where each flight is an object with the following properties

Examples

Get all flights departing at Frankfurt International Airport (EDDF) from 12pm to 1pm on Jan 29 2018:

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/flights/departure?
airport=EDDF&begin=1517227200&end=1517230800" | python -m json.tool
```

Track by Aircraft

> **Note**
>
> The tracks endpoint is purely **experimental**. You can use the flights endpoint for historical data: Flights in Time Interval.

Retrieve the trajectory for a certain aircraft at a given time. The trajectory is a list of waypoints containing position, barometric altitude, true track and an on-ground flag.

In contrast to state vectors, trajectories do not contain all information we have about the flight, but rather show the aircraft's general movement pattern. For this reason, waypoints are selected among available state vectors given the following set of rules:

- The first point is set immediately after the the aircraft's expected departure, or after the network received the first poisition when the aircraft entered its reception range.
- The last point is set right before the aircraft's expected arrival, or the aircraft left the networks reception range.
- There is a waypoint at least every 15 minutes when the aircraft is in-flight.
- A waypoint is added if the aircraft changes its track more than 2.5°.
- A waypoint is added if the aircraft changes altitude by more than 100m (~330ft).
- A waypoint is added if the on-ground state changes.

Tracks are strongly related to flights. Internally, we compute flights and tracks within the same processing step. As such, it may be benificial to retrieve a list of flights with the API methods from above, and use these results with the given time stamps to retrieve detailed track information.

Operation

```
GET /tracks
```

Request

Property	Type	Description
icao24	string	Unique ICAO 24-bit address of the transponder in hex string representation. All letters need to be lower case
time	integer	Unix time in seconds since epoch. It can be any time betwee start and end of a known flight. If time = 0, get the live track if there is any flight ongoing for the given aircraft.

Response

This endpoint is experimental and can be out of order at any time.

The response is a JSON object with the following properties:

Property	Type	Description
icao24	string	Unique ICAO 24-bit address of the transponder in lower case hex string representation.
startTime	integer	Time of the first waypoint in seconds since epoch (Unix time).
endTime	integer	Time of the last waypoint in seconds since epoch (Unix time).
calllsign	string	Callsign (8 characters) that holds for the whole track. Can be null.
path	array	Waypoints of the trajectory (description below).

Waypoints are represented as JSON arrays to save bandwidth. Each point contains the following information:

Index	Property	Type	Description
0	*time*	integer	Time which the given waypoint is associated with in seconds since epoch (Unix time).
1	*latitude*	float	WGS-84 latitude in decimal degrees. Can be null.
2	*longitude*	float	WGS-84 longitude in decimal degrees. Can be null.
3	*baro_altitude*	float	Barometric altitude in meters. Can be null.
4	*true_track*	float	True track in decimal degrees clockwise from north (north=0°). Can be null.
5	*on_ground*	boolean	Boolean value which indicates if the position was retrieved from a surface position report.

Limitations

It is not possible to access flight tracks from more than 30 days in the past.

Examples

Get the live track for aircraft with transponder address *3c4b26* (D-ABYF)

```
$ curl -u "USERNAME:PASSWORD" -s "https://opensky-network.org/api/tracks/all?icao24=3c4b26&time=0"
```