

FLIGHT TRACKER

Flight/Tail #:

[▶ Track Any Flight](#)

Enter Airline Name Here

Flight Number:

[▶ Track Commercial Flight](#)

[Don't know the flight number?](#)

Airport Code:

[▶ View Airport Activity](#)

[Don't know the airport code?](#)

▶ Documentation

Index

- [Overview](#)
- [Procedures/Functions](#)
- [Code examples](#)

Overview

About

This document contains everything you should need to know to create a fully functional application using FlightAware's DirectFlight API. Every function available is documented as well as the array result for that query. Additionally, code examples in various languages are provided to make the process as easy as possible.

DirectFlight uses SOAP/WSDL to easily integrate into new or existing applications using any modern development platform. Applications should fetch the DirectFlight WSDL, parse and cache the result, and then make queries as necessary by submitting SOAP requests directly to FlightAware with your FlightAware-supplied [DirectFlight key](#).

Authentication

To access DirectFlight, all requests must include a username and DirectFlight Key ([don't have one?](#)). This data is transmitted via basic HTTP Authentication methods that encode user:key in base64 and send the data to the DirectFlight server as a "Authorization" header as part of the HTTP request.

Most programming languages and HTTP/SOAP/XML/WSDL packages allow you to specify your username and password as an argument for the request so that the authentication is transparent to your application as it makes requests.

Web Services / WSDL

View: [DirectFlight 1.0 WSDL XML](#)

Although you can read the WSDL and generate SOAP queries manually, it is recommended that you develop your applications to automatically parse the WSDL and populate your application namespace with the DirectFlight functions. Additionally, it is strongly suggested that you ensure that your applications cache the WSDL file so that it is not necessary to fetch and parse the WSDL for every request or instance of your application. This will vastly improve the performance and efficiency of your application.

FlightAware will continue to develop DirectFlight and expand the 1.0 WSDL. However, in the event of major changes or a lack of compatibility, FlightAware will increment the version number and notify you that you should consider upgrading your application to the latest version.

Procedures / Functions

[FlightInfo](#)
[InFlightInfo](#)
[GetLastTrack](#)
[Search](#)
[SearchCount](#)
[Scheduled](#)
[Departed](#)
[Enroute](#)
[FleetArrived](#)
[Arrived](#)
[AllAirports](#)
[ZipcodeInfo](#)
[AirportInfo](#)
[TailOwner](#)
[RoutesBetweenAirports](#)
[AircraftType](#)
[countAirportOperations](#)
[blockIdentCheck](#)
[METAR](#)
[TAF](#)
[NTAF](#)
[LatLongsToDistance](#)
[LatLongsToHeading](#)
[MapFlight Beta](#)

FlightInfo

FlightInfo returns information about flights for a specific tail number (e.g., **N12345**) or ICAO airline and flight number (e.g., **SWA2558**) and the maximum number of flights to be returned. Flight information will be returned from newest to old est with the oldest not being more than 3-4 days in the past.

Times are in integer seconds-since-1970 (UNIX epoch) time, except for estimated time enroute, which is in hours and minutes.

- Arguments
 - ident [string](#)
 - howMany [int](#)
- Returns
 - [FlightInfoStruct](#)

InFlightInfo

InFlightInfo looks up a specific tail number (e.g., **N12345**) or ICAO airline and flight number (e.g., **SWA2558**) and returns current position/direction/speed information. It is only useful for airborne flights.

- Arguments
 - ident [string](#)

- Returns
 - [InFlightAircraftStruct](#)

GetLastTrack

GetLastTrack looks up a flight by specific tail number (e.g., **N12345**) or ICAO airline and flight number (e.g., **SWA2558**). It returns the track log from the current IFR flight or, if the aircraft is not airborne, the most recent IFR flight. It returns an array of positions, with each including the timestamp, longitude, latitude, groundspeed, altitude, altitudestatus, updatetype, and altitudechange. Altitude is in hundreds of feet or Flight Level where appropriate, see <http://flightaware.com/about/faq.rvt#flightLevel>. Also included altitude status, update type, and altitude change

Altitude status is 'C' when the flight is more than 200 feet away from its ATC-assigned altitude. (For example, the aircraft is transitioning to its assigned altitude.) Altitude change is 'C' if the aircraft is climbing (compared to the previous position reported), 'D' for descending, and empty if it is level. This happens for VFR flights with flight following, among other things. Timestamp is integer seconds-since-1970.

- Arguments
 - ident [string](#)
- Returns
 - [ArrayOfTrackStruct](#)

Search

Search searches data on all airborne aircraft to find ones matching the search query. Query parameters include a latitude/longitude box, aircraft ident with wildcards, type with wildcards, prefix, suffix, origin airport, destination airport, origin or destination airport, groundspeed, and altitude. It takes search terms in a single string comprising "-key value" pairs and returns an array of flight structures.

Keys include:

- -prefix
- -type
- -suffix
- -idents
- -destination
- -origin
- -originOrDestination
- -aboveAltitude
- -belowAltitude
- -aboveGroundspeed
- -belowGroundspeed
- -latlong

To search for all aircraft below ten-thousand feet with a groundspeed over 200 kts:

-belowAltitude 100 -aboveGroundspeed 200

To search for all in-air Boeing 777s:

-type B77*

To search for all aircraft heading to Los Angeles International Airport (LAX) that are "heavy" aircraft:

-destination LAX -prefix H

To search for all Continental Airlines flights in Boeing 737s

-idents COA* -type B73*

- Arguments
 - query [string](#)
 - howMany [int](#)
 - offset [int](#)
- Returns
 - [InFlightStruct](#)

SearchCount

SearchCount works like *Search* but returns a count of matching flights rather than information about each flight.

- Arguments
 - query [string](#)
- Returns
 - [int](#)

Scheduled

Scheduled returns information about scheduled flights (technically, filed IFR flights) for a specified airport and a maximum number of flights to be returned. Scheduled flights are returned from soonest to furthest in the future to depart.

The **airport** argument must be the ICAO airport ID (e.g., KLAX, KSFO, KIAH, KHOU, KJFK, KEWR, KORD, KATL, etc).

The **howMany** argument must be an integer value less than or equal to 15 and determines the number of results.

The **offset** argument must be an integer value of the offset row count you want the search to start at. Most requests should be 0.

The **filter** argument can be "ga" or "airline" to only show GA or Airline traffic, respectively, or null/empty to show all traffic.

The **next_offset** value returned advises an application of the next offset to use (if more data is available).

Times returned are UNIX epoch (integer seconds since 1970) format.

- Arguments
 - airport [string](#)
 - howMany [int](#)
 - filter [string](#)
 - offset [int](#)
- Returns
 - [ScheduledStruct](#)

Departed

Departed returns information about already departed flights for a specified airport and maximum number of flights to be returned. Departed flights are returned in order from most recently to least recently departed.

The **airport** argument must be the ICAO airport ID (e.g., KLAX, KSFO, KIAH, KHOU, KJFK, KEWR, KORD, KATL, etc).

The **howMany** argument must be an integer value less than or equal to 15 and determines the number of results.

The **offset** argument must be an integer value of the offset row count you want the search to start at. Most requests should be 0.

The **filter** argument can be "ga" or "airline" to only show GA or Airline traffic, respectively, or null/empty to show all traffic.

The **next_offset** value returned advises an application of the next offset to use (if more data is available).

- Arguments
 - airport [string](#)
 - howMany [int](#)
 - filter [string](#)
 - offset [int](#)
- Returns
 - [DepartureStruct](#)

Enroute

Enroute returns information about flights already in the air for the specified airport and maximum number of flights to be returned. Enroute flights are returned from soonest estimated arrival to least soon estimated arrival.

The **airport** argument must be the ICAO airport ID (e.g., KLAX, KSFO, KIAH, KHOU, KJFK, KEWR, KORD, KATL, etc).

The **howMany** argument must be an integer value less than or equal to 15 and determines the number of results.

The **offset** argument must be an integer value of the offset row count you want the search to start at. Most requests should be 0.

The **filter** argument can be "ga" or "airline" to only show GA or Airline traffic, respectively, or null/empty to show all traffic.

The **next_offset** value returned advises an application of the next offset to use (if more data is available).

- Arguments
 - airport [string](#)
 - howMany [int](#)
 - filter [string](#)
 - offset [int](#)
- Returns
 - [EnrouteStruct](#)

FleetArrived

The **fleet** argument must be an ICAO prefix (e.g., COA, DAL, UAL, OPT, etc.)

The **howMany** argument must be an integer value less than or equal to 15 and determines the number of results.

The **offset** argument must be an integer value of the offset row count you want the search to start at. Most requests should be 0.

The **next_offset** value returned advises an application of the next offset to use (if more data is available).

- Arguments
 - fleet [string](#)
 - howMany [int](#)
 - offset [int](#)
- Returns
 - [ArrivalStruct](#)

Arrived

Arrived returns information about flights that have recently arrived for the specified airport and maximum number of flights to be returned. Flights are returned from most to least recent.

The **airport** argument must be the ICAO airport ID (e.g., KLAX, KSFO, KIAH, KHOU, KJFK, KEWR, KORD, KATL, etc).

The **howMany** argument must be an integer value less than or equal to 15 and determines the number of results.

The **offset** argument must be an integer value of the offset row count you want the search to start at. Most requests should be 0.

The **filter** argument can be "ga" or "airline" to only show GA or Airline traffic, respectively, or null/empty to show all traffic.

The **next_offset** value returned advises an application of the next offset to use (if more data is available).

- Arguments

- [airport](#) *string*
 - [howMany](#) *int*
 - [filter](#) *string*
 - [offset](#) *int*
 - Returns
 - [ArrivalStruct](#)
-

AllAirports

AllAirports returns the ICAO airport IDs of all known airports.

- Arguments
 - None
 - Returns
 - [ArrayOfString](#)
-

ZipcodeInfo

ZipcodeInfo returns information about a five-digit zipcode. Of particular importance is latitude and longitude.

- Arguments
 - [zipcode](#) *string*
 - Returns
 - [ZipcodeInfoStruct](#)
-

AirportInfo

AirportInfo returns information about an airport given an ICAO airport code such as KLAX, KSFO, KORD, KIAH, O07, etc. Data returned includes name (Houston Intercontinental Airport), location (typically city and state), latitude and longitude.

- Arguments
 - [airportCode](#) *string*
 - Returns
 - [AirportInfoStruct](#)
-

TailOwner

TailOwner returns information about the owner of an aircraft, given a flight number or N-number. Data returned includes owner's name, location (typically city and state), and website, if any.

- Arguments
 - [ident](#) *string*
 - Returns
 - [TailOwnerStruct](#)
-

RoutesBetweenAirports

RoutesBetweenAirports returns information about assigned IFR routings between two airports. For each known routing, the route, number of times that route has been assigned and the filed altitude are returned.

- Arguments
 - [origin](#) *string*
 - [destination](#) *string*
 - Returns
 - [ArrayOfRoutesBetweenAirportsStruct](#)
-

AircraftType

Given an aircraft type string such as GALX, *AircraftType* returns information about that type, comprising the manufacturer (for instance, "IAI"), type (for instance, "Gulfstream G200"), and description (like "twin-jet").

- Arguments
 - [type](#) *string*
 - Returns
 - [AircraftTypeStruct](#)
-

countAirportOperations

Given an airport, returns integer values on the number of aircraft scheduled or actually en route or departing from the airport. Scheduled arrival is a non-airborne flight that is scheduled to the airport in question.

- Arguments
 - airport [string](#)
 - Returns
 - [countAirportOperationsStruct](#)
-

blockIdentCheck

Given an aircraft identification, returns 1 if the aircraft is blocked from public tracking, 0 if it is not.

- Arguments
 - ident [string](#)
 - Returns
 - [int](#)
-

METAR

Given an airport, return the METAR weather info, if available.

- Arguments
 - airport [string](#)
 - Returns
 - [string](#)
-

TAF

Given an airport, return the terminal area forecast, if available.

- Arguments
 - airport [string](#)
 - Returns
 - [string](#)
-

NTAF

Given an airport, return the terminal area forecast, if available.

- Arguments
 - airport [string](#)
 - Returns
 - [TafStruct](#)
-

LatLongsToDistance

Given two latitudes and longitudes, lat1 lon1 lat2 and lon2, respectively, determine the great circle distance between those positions in miles.

- Arguments
 - lat1 [float](#)
 - lon1 [float](#)
 - lat2 [float](#)
 - lon2 [float](#)
 - Returns
 - [int](#)
-

LatLongsToHeading

Given two latitudes and longitudes, lat1 lon1 lat2 and lon2, respectively, calculate and return the initial compass heading (where 360 is North) from position one to position two. Quite accurate for relatively short distances but since it assumes the earth is a sphere rather than an irregular oblate spheroid may be inaccurate for flights around a good chunk of the world, etc.

- Arguments
 - lat1 [float](#)
 - lon1 [float](#)
 - lat2 [float](#)
 - lon2 [float](#)
 - Returns
 - [int](#)
-

MapFlight_Beta

This function will return a base64 encoded GIF (with the height and width as specified in pixels) of the most recent (past or current) flight of a specified ident. This service is in beta. Future versions will allow viewing of past flights, configuring nexrad, and configuring zoom.

- Arguments
 - ident [string](#)
 - mapHeight [int](#)
 - mapWidth [int](#)
- Returns
 - [string](#)

[Classes/Structs](#) | [Enumerations](#) | [Array Types](#) | [Simple Types](#)

Classes/Structs

[DepartureStruct](#)
[TafStruct](#)
[FlightStruct](#)
[ScheduledFlightStruct](#)
[EnrouteStruct](#)
[AirportInfoStruct](#)
[TrackStruct](#)
[ArrivalStruct](#)
[ZipcodeInfoStruct](#)
[AircraftSuffixDescriptionStruct](#)
[RoutesBetweenAirportsStruct](#)
[MetarStruct](#)
[InFlightAircraftStruct](#)
[EnrouteFlightStruct](#)
[ScheduledStruct](#)
[InFlightStruct](#)
[countAirportOperationsStruct](#)
[TailOwnerStruct](#)
[DepartureFlightStruct](#)
[AircraftTypeStruct](#)
[ArrivalFlightStruct](#)
[FlightInfoStruct](#)

DepartureStruct

- next_offset [int](#)
- departures [ArrayOfDepartureFlightStruct](#)

TafStruct

- airport [string](#)
- timeString [string](#)
- forecast [ArrayOfString](#)

FlightStruct

- ident [string](#)
- aircrafttype [string](#)
- filed_ete [string](#)
- filed_time [int](#)
- filed_departuretime [int](#)
- filed_airspeed_kts [int](#)
- filed_airspeed_mach [string](#)
- filed_altitude [int](#)
- route [string](#)
- actualdeparturetime [int](#)
- estimatedarrivaltime [int](#)
- actualarrivaltime [int](#)
- diverted [string](#)
- origin [string](#)
- destination [string](#)
- originName [string](#)
- originCity [string](#)
- destinationName [string](#)
- destinationCity [string](#)

ScheduledFlightStruct

- ident [string](#)
- aircrafttype [string](#)
- filed_departuretime [int](#)
- estimatedarrivaltime [int](#)
- origin [string](#)
- destination [string](#)
- originName [string](#)
- originCity [string](#)
- destinationName [string](#)
- destinationCity [string](#)

EnrouteStruct

- next_offset [int](#)
- enroute [ArrayOfEnrouteFlightStruct](#)

AirportInfoStruct

- name [string](#)
- location [string](#)
- longitude [float](#)
- latitude [float](#)
- timezone [string](#)

TrackStruct

- timestamp [int](#)
- latitude [float](#)
- longitude [float](#)
- groundspeed [int](#)
- altitude [int](#)
- altitudeStatus [string](#)
- updateType [string](#)
- altitudeChange [string](#)

ArrivalStruct

- next_offset [int](#)
- arrivals [ArrayOfArrivalFlightStruct](#)

ZipcodeInfoStruct

- latitude [float](#)
- longitude [float](#)
- city [string](#)
- state [string](#)
- county [string](#)

AircraftSuffixDescriptionStruct

- description [string](#)

RoutesBetweenAirportsStruct

- count [int](#)
- route [string](#)
- filedAltitude [int](#)

MetarStruct

- name [string](#)
- timeString [string](#)
- reading [string](#)

InFlightAircraftStruct

- ident [string](#)
- prefix [string](#)
- type [string](#)
- suffix [string](#)
- origin [string](#)
- destination [string](#)
- timeout [string](#)
- timestamp [int](#)
- firstPositionTime [int](#)
- longitude [float](#)
- latitude [float](#)
- groundspeed [int](#)
- altitude [int](#)
- heading [int](#)
- altitudeStatus [string](#)
- updateType [string](#)
- altitudeChange [string](#)

EnrouteFlightStruct

- ident [string](#)
- aircrafttype [string](#)
- actualdeparturetime [int](#)
- estimatedarrivaltime [int](#)
- origin [string](#)
- destination [string](#)
- originName [string](#)
- originCity [string](#)
- destinationName [string](#)
- destinationCity [string](#)

ScheduledStruct

- next_offset [int](#)
- scheduled [ArrayOfScheduledFlightStruct](#)

InFlightStruct

- next_offset [int](#)
- aircraft [ArrayOfInFlightAircraftStruct](#)

countAirportOperationsStruct

- enroute [int](#)
- departed [int](#)
- scheduled_departures [int](#)
- scheduled_arrivals [int](#)

TailOwnerStruct

- owner [string](#)
- location [string](#)
- location2 [string](#)
- website [string](#)

DepartureFlightStruct

- o ident [string](#)
- o aircrafttype [string](#)
- o actualdeparturetime [int](#)
- o estimatedarrivaltime [int](#)
- o actualarrivaltime [int](#)
- o origin [string](#)
- o destination [string](#)
- o originName [string](#)
- o originCity [string](#)
- o destinationName [string](#)
- o destinationCity [string](#)

AircraftTypeStruct

- o manufacturer [string](#)
- o type [string](#)
- o description [string](#)

ArrivalFlightStruct

- o ident [string](#)
- o aircrafttype [string](#)
- o actualdeparturetime [int](#)
- o actualarrivaltime [int](#)
- o origin [string](#)
- o destination [string](#)
- o originName [string](#)
- o originCity [string](#)
- o destinationName [string](#)
- o destinationCity [string](#)

FlightInfoStruct

- o next_offset [int](#)
- o flights [ArrayOfFlightStruct](#)

Enumerations

Array Types

- [ArrayOfScheduledFlightStruct ScheduledFlightStruct\(\)](#)
- [ArrayOfTrackStruct TrackStruct\(\)](#)
- [ArrayOfRoutesBetweenAirportsStruct RoutesBetweenAirportsStruct\(\)](#)
- [ArrayOfEnrouteFlightStruct EnrouteFlightStruct\(\)](#)
- [ArrayOfFlightStruct FlightStruct\(\)](#)
- [ArrayOfDepartureFlightStruct DepartureFlightStruct\(\)](#)
- [ArrayOfArrivalFlightStruct ArrivalFlightStruct\(\)](#)
- [ArrayOfInFlightAircraftStruct InFlightAircraftStruct\(\)](#)
- [ArrayOfString string\(\)](#)

Examples

Microsoft .NET

Save file as test.cs:

```
using System.Net;
class test {
    public static void Main(string[] args)
    {
        DirectFlight df = new DirectFlight();
        df.Credentials = new NetworkCredential("sampleUser", "abc123abc123abc123abc123abc123");
        df.PreAuthenticate = true;
        EnrouteStruct r = df.Enroute("KAUS", 10, "", 0);

        foreach (EnrouteFlightStruct e in r.enroute) {
            System.Console.WriteLine(e.ident);
        }

        System.Console.WriteLine(df.METAR("KAUS"));
    }
}
```

Run commands:

```
wSDL http://flightaware.com/commercial/directflight/data/wsd11.xml

csc test.cs DirectFlight.cs
test.exe
```

PHP

Requirements

- PHP4 or PHP5
- pear-HTTP
- pear-SOAP

```

<?php

require_once('SOAP/Client.php');

$DirectFlight_Authentication = array(
    'user'      => 'sampleUser',
    'pass'      => 'abc123abc123abc123abc123abc123abc123',
);

$wsdl_url = 'http://flightaware.com/commercial/directflight/data/wsdl1.xml';
$WSDL = new SOAP_WSDL($wsdl_url,$DirectFlight_Authentication);
$soap = $WSDL->getProxy();

$result = $soap->Enroute('KIAH',10,'airline',0);

php?>

```

Perl

Requirements

- Perl5 or above
- SOAP::Lite

Save as test.pl:

```

# API Configuration

my $username = "sampleUser";
my $apiKey = "abc123abc123abc123abc123abc123abc123";

# Example code

use strict;
use DirectFlight;
use SOAP::Lite;

sub SOAP::Transport::HTTP::Client::get_basic_credentials {
    return $username => $apiKey;
}

my $DF = new DirectFlight;

my $enroute = $DF->Enroute('KSMO',10,'',0);

print "Aircraft en route to KSMO:\n";

my $flights = $enroute->(enroute);

foreach my $flight (@$flights) {
    print $flight->{'ident'} . " (" . $flight->{'aircrafttype'} . ") \t" .
    $flight->{'originName'} . " (" . $flight->{'origin'} . ")\n";
}

```

Run commands:

```

perl /usr/local/bin/stubmaker.pl -v http://flightaware.com/commercial/directflight/data/wsdl1.xml
./test.pl

```

Java

Requirements

- Java 1.5
- [Apache AXIS](#)

test.java:

```

import com.flightaware.directflight.soap.DirectFlight.*;

class test {
    public static void main(String[] args)
    {
        try {
            DirectFlightLocator locator = new DirectFlightLocator();
            DirectFlightSoap df = locator.getDirectFlightSoap();
            DirectFlightSoapStub stub = (DirectFlightSoapStub)df;
            stub.setUsername("sampleUser");
            stub.setPassword("abc123abc123abc123abc123");
            EnrouteStruct r = df.enroute("KAUS", 10, "", 0);
            for (EnrouteFlightStruct e: r.getEnroute()) {
                System.out.println(e.getIdent());
            }
            System.out.println(df.METAR("KAUS"));
        } catch (javax.xml.rpc.ServiceException x) {
            System.err.println(x);
        } catch (java.rmi.RemoteException x) {
            System.err.println(x);
        }
    }
}

```

```
}
```

Windows-specific build (requires [wsdl1.xml](#)):

```
set CLASSPATH=axis-1_4\lib\axis-ant.jar;axis-1_4\lib\axis.jar;axis-1_4\lib\commons-discovery-0.2.jar;  
axis-1_4\lib\commons-logging-1.0.4.jar;axis-1_4\lib\jaxrpc.jar;axis-1_4\lib\log4j-1.2.8.jar;  
axis-1_4\lib\saaaj.jar;axis-1_4\lib\wsdl4j-1.5.1.jar;.

java org.apache.axis.wsdl.WSDL2Java wsdl1.xml

javac test.java

java test
```

Tcl

Coming soon

Python

Coming soon