# Fareboom.com

# Flight Search API Integration

## http://api.fareboom.com

# v. 1.0

# Introduction

This document describes the Fareboom.com Flight Search JSON API and its integration with third party meta-search engines.

JSON API is a specification for how a client should request that resources be fetched or modified and how the Fareboom server shall respond to those requests.

JSON API is designed to minimize both the number of requests and the amount of data transmitted between clients and servers. This efficiency is achieved without compromising readability, flexibility, and discoverability.

JSON API requires use of the JSON API media type (application/vnd.api+json) for exchanging data.

A JSON API server supports fetching of resources through the HTTP method GET.

## API URL and Methods

The API is available at <http://api.fareboom.com>.

Two methods are currently supported:

* FlightSearch: runs full flight search per input request
* GetItinerary: retrieves detailed information about a specific flight itinerary

The FlightSearch is invoked by passing a specially prepared AirItineraryAPIRequest object.

## API Request

The API request contains the so-called Flight Search Request or “AirItinerarySearchApiRequest” parameters necessary to perform a flight search. At a minimum it must contain parameters listed in the sample below:

{

"clientIp": "192.168.100.90",

"currency": "USD",

"legs": [

{

"arrivalLocID": "MNL",

"departureDate": "2015-06-30",

"departureLocID": "LAX",

"departureTimeIgnore": true

},

{

"arrivalLocID": "LAX",

"departureDate": "2015-07-15",

"departureLocID": "MNL",

"departureTimeIgnore": true

}

],

"passengers": {

"adults": 1,

"children": 0,

"infants": 0

}

}

### Important Required Parameters Notes

* “clientIp”: end user browser IP address. This address is required to adhere to our marketing and sales restrictions that limit the distribution of certain airlines and/or fares in specific countries or regions. The IP address is evaluated at runtime to determine the user location country or region and is then discarded in further processing. ***We do not collect, share, store or analyze user IP data, therefore there are no user privacy legal implications resulting from passing of this parameter.***
* departureDate / returnDate format: we implement the ISO 8601 format as yyyy-mm-dd. Optionally, departure time may be specified in the format yyyy-mm-ddThh:mm:ss”. When time is specified only flights around that time are returned. If no flights exist at the given time then search is expanded to include a wider time frame.
* departureTimeIgnore: leave true by default for time neutral searches.

Additional descriptions of mandatory and optional parameters are included in the screenshot below. The client application design will determine which parameters will be sent at all times, based on flight search form options that are made available on the client website.

### AirItinerarySearchApiRequest Details Screenshot



### AirItinerarySearchApiRequest Structs

public struct PassengerApiRequest

{

public byte Adults;

public byte Children;

public byte Infants;

}

public struct LegApiRequest

{

public string DepartureLocID;

public int DepartureRadiusMiles;

public string ArrivalLocID;

public int ArrivalRadiusMiles;

public DateTime DepartureDate;

public bool? DepartureTimeIgnore;

public byte DepartureDateOffsetNegative;

public byte DepartureDateOffsetPositive;

}

public struct AirItinerarySearchApiRequest

{

public string ClientIp;

public int Cabin;

public string Currency;

public PassengerApiRequest Passengers;

public IList<LegApiRequest> Legs;

public IList<string> Carriers;

public override string ToString()

{

var serializer = new JavaScriptSerializer();

return serializer.Serialize(this);

}

## API Response

Each API response is assigned a unique one-time “cacheKey” that is valid only for the duration of the current search (up to 30 minutes). The cacheKey is required when redirecting the user to a specific flight itinerary landing page on the [www.fareboom.com](http://www.fareboom.com) site. More on this topic follows below.

Below screenshot highlights critical referencing between each flight itinerary offer and its flight sectors (physical flights). For data compression reasons each itinerary references a global collection of unique flight sectors via a “sectors” references array (see red arrow pointer below for sample). The client application is responsible for mapping correctly individual flight sectors to their referencing itineraries.

### AirItinerarySearchApiResponse Screenshots

Screenshot 1: Flight Itinerary and Sector Referencing, Fare Information

Note: fare breakdown is coded in the accountingLines class (see screenshot notes for details).



Screenshot 2:

Flight Sector Properties



Screenshot 3:

Flight Sector Pricing, Cabin and Baggage Allowance Information



### AirItinerarySearchApiResponse structs

AirItinerarySearchApiResponse is the parent wrapper class containing a list of unique Flight Sectors (FlightSectorApiResponse class) and separately of unique Flight Itineraries (AirItineraryApiResponse class).

Below is a list of structs that implement the necessary data structures encapsulating the full API Flight Search Response.

public struct FlightSectorApiResponse

{

public int Id;

public string DepartureLocation;

public string ArrivalLocation;

public DateTime DepartureDateTime;

public DateTime ArrivalDateTime;

public string MarketingCarrierId;

public string OperatingCarrierId;

public int MarketingFlightNumber;

public string AircraftType;

public int NumStops;

public string DepartureTerminal;

public string ArrivalTerminal;

public bool ChangeOfGauge;

public double FlightDurationHrs;

public string Stop1Location;

public string Stop2Location;

public double Stop1DurationHrs;

public double Stop2DurationHrs;

}

public struct AirItineraryApiResponse

{

public string Id;

public IList<LegApiResponse> Legs;

public IList<PassengerApiResponse> Passengers;

}

public struct PassengerApiResponse

{

public PersonApiResponse Person;

public IList<AccountingLineApiResponse> AccountingLines;

public IList<AirPassengerSectorApiResponse> BookingInfo;

}

public struct PersonApiResponse

{

public PtcApiResponse DefaultPtc;

}

public struct PtcApiResponse

{

public string Code;

public int Type;

}

public struct LegApiResponse

{

public IList<int> Sectors;

public string PrimaryCarrierId;

public double LegDurationHrs;

}

public struct AccountingLineApiResponse

{

public int AccountTypeId;

public decimal Amount;

public string Currency;

}

public struct AirPassengerSectorApiResponse

{

public string FareBasis;

public int Cabin;

public int BaggageAllowType;

public int BaggageAllowUnits;

}

public struct AirItinerarySearchApiResponse

{

public string CacheKey;

public IList<FlightSectorApiResponse> FlightSectors;

public IList<AirItineraryApiResponse> Itineraries;

}

### Test Search Code

Below C# code should be used in testing to ensure that valid results are returned. Trip departure dates included in the AirItinerarySearchApiRequest may require modification in case they are in the past.

using System;

using System.Dynamic;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using BTS.Web.ApiMessaging.Messages;

using BTS.Web.ApiMessaging.Models;

using Newtonsoft.Json;

async static void TestSearch()

{

using (var httpclient = CreateHttpClient())

{

var req = @"{

""cabin"": 1,

""carriers"": [""YY""],

""clientIp"": ""192.168.100.25"",

""currency"": ""USD"",

""legs"": [

{

""arrivalLocID"": ""MNL"",

""arrivalRadiusMiles"": 0,

""departureDate"": ""2015-01-07"",

""departureDateOffsetNegative"": 0,

""departureDateOffsetPositive"": 0,

""departureLocID"": ""LAX"",

""departureRadiusMiles"": 0,

""departureTimeIgnore"": true

},

{

""arrivalLocID"": ""LAX"",

""arrivalRadiusMiles"": 0,

""departureDate"": ""2015-01-14"",

""departureDateOffsetNegative"": 0,

""departureDateOffsetPositive"": 0,

""departureLocID"": ""MNL"",

""departureRadiusMiles"": 0,

""departureTimeIgnore"": true

}

],

""passengers"": {

""adults"": 1,

""children"": 0,

""infants"": 0

}

}";

Console.Write("Searching...");

var resp = await httpclient.PostAsync("api/Search/FlightSearch", new StringContent(req, Encoding.Default, "application/json"));

var content = await resp.Content.ReadAsStringAsync();

var searchResults = JsonConvert.DeserializeObject<AirItinerarySearchApiResponse>(content);

Console.WriteLine("OK. Found " + searchResults.Itineraries.Mo(r => r.Count) + " itineraries");

## Reservation Redirect Landing Page

The Reservation Redirect landing page is the Fareboom.com page where the user is redirected after clicking on a specific flight itinerary on the meta-search site. This page redisplays the selected itinerary to the user and it performs a background price check to ensure that the quoted fare is up-to-date. The page handles possible availability and fare changes via appropriate messaging and optionally by offering to the user a new flight search in case of unexpected seat availability changes.

### URL Path

The Reservation Redirect landing page is located at: AirItny/{cacheKey}/{itnyGuid}

The URL requires two dynamic elements as follows:

* cacheKey: top element denoting the current search in the AirItinerarySearchApiResponse. It is valid for 30 minutes (validity duration is subject to change based on Fareboom caching logic)
* itnyGuid: selected flight itinerary unique identifier passed in the AirItineraryApiResponse/Id property.