# Enhanced Flight Search Service using Google's QPX Express API

Vinayak Pawar & Poorn Pragya

## Introduction

In recent years, there have been rapid developments in the field of airline industries. As more and more airlines join in competition, travelers have been provided with many options when they travel across the globe. Biggest concern faced by travelers is airline reservation. Apart from flight costs being major factor in deciding which flights to take, many parameters such as - airline company reviews, layover airports, layover duration, departure and arrival timings, etc come into picture. Moreover, there are various ways a traveler can perform airline reservation (online on airline's portal, by calling helpdesk, contacting various travel agents, etc). Flight ticket prices always fluctuate and it becomes difficult for traveler to finalize reservation as he is always looking for so called best offer. For this, customer searches on various websites to get best prices, and deals. In this whole process, some people become victim of phishing attacks, and scams. There have been situations in the past in which travelers booked flight tickets from travel agents and they had been denied flight boarding for various reasons.

## Problem Statement

It is important that traveler should reserve ticket from legitimate sources and experience hassle free journey. Most airlines provide platform where travelers can book their tickets. It is tedious task to go on every airline’s website and check available flights. Airlines have their own criteria and rules to open up reservation windows. It becomes difficult for end user to track each and every airline’s details.

In our term project, we are providing functionalities around flight search mechanism. In 2010, Google acquired ITA Software. ITA Software’s QPX (which has now been used as one of the google’s APIs) provides end users a better flight search experience and allow them to buy flight tickets online. We have built a small application which requests flights and gets responses back with the use of this QPX Express API. Our main objective is to fetch flight details and show them to end users. This API takes data from most of the airlines and consolidate that data into response.

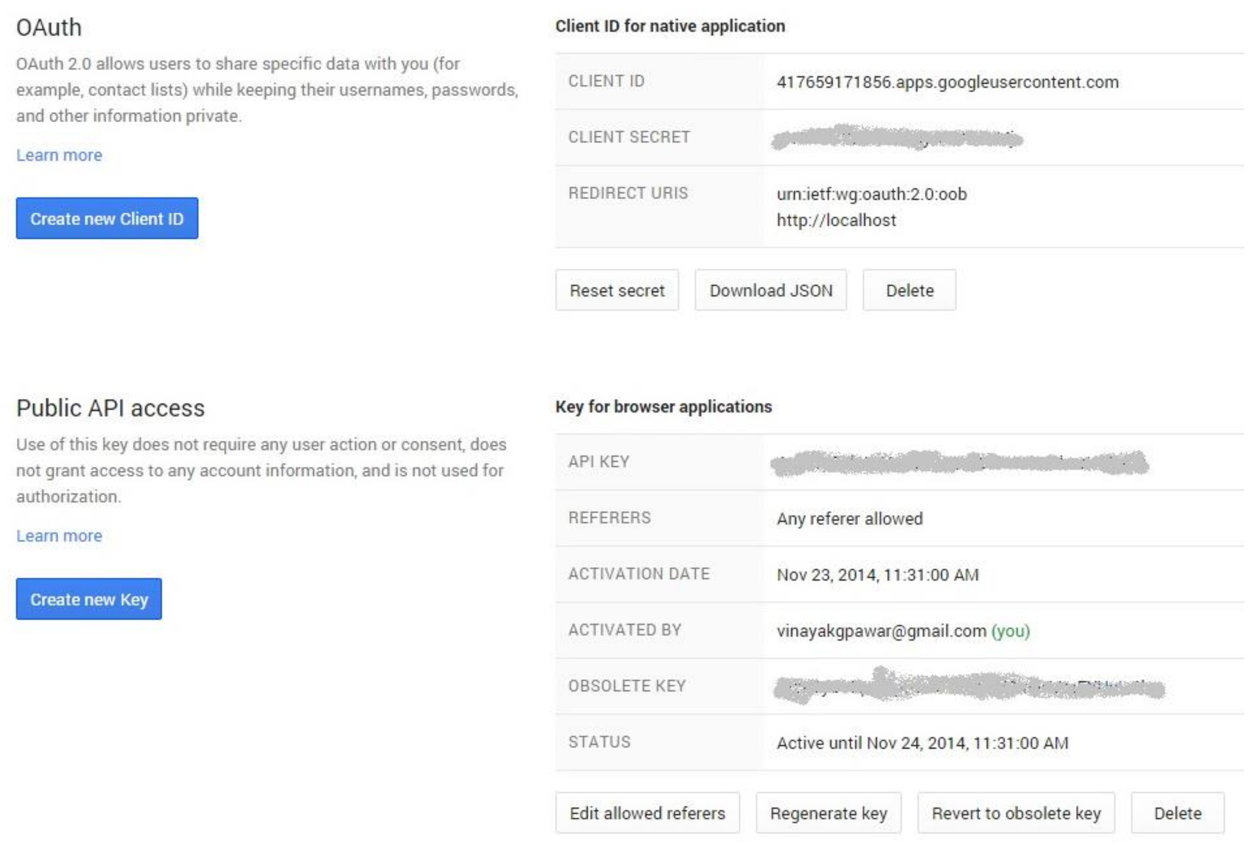
## Methodology

Our implementation is based on following sequential steps –

1. **Creating a Google API Project**

   To utilize Google’s APIs, google wants users to create projects online and generate API key. This unique API key is used for any request sent by user from standalone application. This particular API monitors activities performed by end user with the API and maintains usage details. Daily limit for number of request has been set to 50. If exceeded, free user cannot send more requests. This was one of the limitation for our project. For business purpose, there is a charge of $0.035 per request.

   Following screenshot shows how project API credential tab looks like. API accepts requests in JSON format. End user needs to send it with HTTP POST request.



OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

Learn more

**Client ID for native application**

| | |
|---|---|
| CLIENT ID | 417659171856.apps.googleusercontent.com |
| CLIENT SECRET | [redacted] |
| REDIRECT URIS | urn:ietf:wg:oauth:2.0:oob<br>http://localhost |

Public API access

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

Learn more

**Key for browser applications**

| | |
|---|---|
| API KEY | [redacted] |
| REFERERS | Any referer allowed |
| ACTIVATION DATE | Nov 23, 2014, 11:31:00 AM |
| ACTIVATED BY | vinayakgpawar@gmail.com (you) |
| OBSOLETE KEY | [redacted] |
| STATUS | Active until Nov 24, 2014, 11:31:00 AM |

2. **Creating a JSON request as per user's input and sending it to API**
   Our application asks user for inputs and forms a request.json file which can then be sent to QPX Express. Standard JSON request looks like below. Once, user inputs are received, they will be plugged into this standard file.

Standard JSON request format -

```
{
 "request": {
  "passengers": {
   "kind": "qpxexpress#passengerCounts",
   "adultCount": integer,
   "childCount": integer,
   "infantInLapCount": integer,
   "infantInSeatCount": integer,
   "seniorCount": integer
  },
  "slice": [
   {
    "kind": "qpxexpress#sliceInput",
    "origin": string,
    "destination": string,
    "date": string,
    "maxStops": integer,
    "maxConnectionDuration": integer,
    "preferredCabin": string,
    "permittedDepartureTime": {
      "kind": "qpxexpress#timeOfDayRange",
      "earliestTime": string,
      "latestTime": string
    },
    "permittedCarrier": [
      string
    ],
    "alliance": string,
    "prohibitedCarrier": [
      string
    ]
   }
  ],
  "maxPrice": string,
  "saleCountry": string,
  "refundable": boolean,
  "solutions": integer
 }
}
```

3. **Reading & parsing JSON response in csv file**
   If the request is valid, application gets a response back in following format. For invalid requests, an error response is sent in JSON. Our application then parses this JSON response file using JSON module which is available in CPAN.

QPX Express API Response :

```
{
 "kind": "qpxExpress#tripsSearch",
 "trips": {
   "kind": "qpxexpress#tripOptions",
   "requestId": string,
   "data": {
     "kind": "qpxexpress#data",
     "airport": [
       {
         "kind": "qpxexpress#airportData",
         "code": string,
         "city": string,
         "name": string
       }
```

```
    ],
    "city": [
      {
        "kind": "qpxexpress#cityData",
        "code": string,
        "country": string,
        "name": string
      }
    ],
    "aircraft": [
      {
        "kind": "qpxexpress#aircraftData",
        "code": string,
        "name": string
      }
    ],
    "tax": [
      {
        "kind": "qpxexpress#taxData",
        "id": string,
        "name": string
      }
    ],
    "carrier": [
      {
        "kind": "qpxexpress#carrierData",
        "code": string,
        "name": string
      }
    ]
  },
  "tripOption": [
    {
      "kind": "qpxexpress#tripOption",
      "saleTotal": string,
      "id": string,
      "slice": [
        {
          "kind": "qpxexpress#sliceInfo",
          "duration": integer,
          "segment": [
            {
              "kind": "qpxexpress#segmentInfo",
              "duration": integer,
              "flight": {
                "carrier": string,
                "number": string
              },
              "id": string,
              "cabin": string,
        "bookingCode": string,
        "bookingCodeCount": integer,
        "marriedSegmentGroup": string,
        "subjectToGovernmentApproval": boolean,
        "leg": [
         {
          "kind": "qpxexpress#legInfo",
          "id": string,
          "aircraft": string,
          "arrivalTime": string,
          "departureTime": string,
          "origin": string,
          "destination": string,
          "originTerminal": string,
          "destinationTerminal": string,
          "duration": integer,
          "operatingDisclosure": string,
          "onTimePerformance": integer,
          "mileage": integer,
          "meal": string,
          "secure": boolean,
          "connectionDuration": integer,
          "changePlane": boolean
         }
        ],
        "connectionDuration": integer
       }
      ]
     }
    ],
    "pricing": [
     {
      "kind": "qpxexpress#pricingInfo",
      "fare": [
       {
        "kind": "qpxexpress#fareInfo",
        "id": string,
        "carrier": string,
        "origin": string,
        "destination": string,
        "basisCode": string,
        "private": boolean
       }
      ],
      "segmentPricing": [
       {
        "kind": "qpxexpress#segmentPricing",
        "fareId": string,
        "segmentId": string,
        "freeBaggageOption": [
            {
              "kind": "qpxexpress#freeBaggageAllowance",
              "bagDescriptor": [
                {
                  "kind": "qpxexpress#bagDescriptor",
                  "commercialName": string,
                  "count": integer,
                  "description": [
                    string
                  ],
                  "subcode": string
                }
              ],
              "kilos": integer,
              "kilosPerPiece": integer,
              "pieces": integer,
              "pounds": integer
            }
          ]
        }
      ],
      "baseFareTotal": string,
      "saleFareTotal": string,
      "saleTaxTotal": string,
      "saleTotal": string,
      "passengers": {
        "kind": "qpxexpress#passengerCounts",
        "adultCount": integer,
        "childCount": integer,
        "infantInLapCount": integer,
        "infantInSeatCount": integer,
        "seniorCount": integer
      },
      "tax": [
        {
          "kind": "qpxexpress#taxInfo",
          "id": string,
          "chargeType": string,
          "code": string,
          "country": string,
          "salePrice": string
        }
      ],
      "fareCalculation": string,
      "latestTicketingTime": string,
      "ptc": string,
      "refundable": boolean
    }
  ]
}
```

```
   ]
  }
 }
```

4. **Creating lookup files for future processing**
   Data component in this response represents various types of code-value pairs which can be useful while retrieving actual flight details. For each request-response pair, set of lookup files are created. These lookup files are utilized for displaying data in human readable format.

5. **Using csv file and lookup files to generate data in human readable format**
   Our application then uses this newly generated csv file and lookup files to generate flight details in easily readable format.

6. **Providing user list of nearby source and destination airports**
   We have collected a file – airport.dat from http://openflights.org/data.html . This file contains information of all airports around the world. It contains FAA codes, City Names, Longitude, and latitude values. We use longitude and latitude values to determine nearby airports. At the end of search results, we provide end user this list of airports as suggestion for source and destination values.

7. **Allow end user to sort/group results on various parameters**
   We allow end user to sort results on total price of flight, and connection time. We have an option to provide end user only those flights which depart and arrive during specific times, and those flights which have layover time less than user defined value.

**Conclusion**

We were able to create a small working application which can fetch flight details per user needs and show them on terminal. We were also able to provide users different options to see results in various ways.

**Future Scope**

1. We were much interested in building an end to end application. Scope of such application can be extended from small standalone script to a dynamic web application where user can search flights and he can be navigated to airline's website for reservation.

2. One more enhancement, we were interested in was – listing flights flying between nearby source and destination airports. In our implementation, we were successful in listing such airports. We had limitations regarding number of requests hitting QPX and we were unable to perform end to end testing. We then decided to stick with basic functionalities.

**Reference**

1. https://developers.google.com/qpx-express/
2. http://openflights.org/data.html