Affinity Search
Solution Document /
Interface Definition Document
for
[REDACTED]

ITA Software, Inc.
01 December, 2010

VERSION HISTORY

v1.0	2010-12-Oct	Peter Fernandez Tom Kirchman Geoff Andrew David Olliges Any Meyer Andrzej Walczak
V2.0	2010-25-Oct	Merged with IDD + response to feedback from [REDACTED]
V3.0	2010-15-Nov	Revision with multiple clarifications
V4.0	2010-01-Dec`	Multiple clarifications for inclusion in agreement

1 INTENT OF THIS DOCUMENT

The purpose of this document is to describe ITA's understanding of the requirements for the [REDACTED] Affinity Search project, as it pertains to ITA's major deliverables. This document can serve as a common source of understanding about the project, and a reference point in terms of deliverables, scope and major requirements.

2 INTRODUCTION

[REDACTED], one of the global leaders in air travel, has enlisted ITA to develop an airfare search tool that will be deployed on both the [REDACTED] and [REDACTED] websites. Dubbed Affinity Search, the ITA-hosted service will cater to travelers who have less specific travel plans, allowing searches by wide ranges of travel dates, destinations and lengths of stay. Using the latest in ITA's low fare search technology, Affinity Search will be deployed as two separate instances of a single implementation, with each instance tailored to the specific business and technical requirements of [REDACTED].

[REDACTED] is pursuing a target launch date of [REDACTED], with the release of the new search product to a single point of sale. The system will be deployed more widely as deemed appropriate, and continuing into a six month total trial deployment.

2.1 Business Context

The objective of Affinity Search is to attract new types of customers, and provide an improvement to [REDACTED] conversion ratio.

2.2 Major Milestones

ITA has been tasked with providing 3 high-level deliverables for the Affinity Search project:

1. An Interface Definition Document (IDD), that describes the mechanisms for passing searches to the system, and consuming and interpreting the results. This document will also define processes for auxiliary services of the system, such as a BIS-like feedback / solution quality assurance mechanism. This will be delivered in the form of a Microsoft Word document containing example syntax, XSD's, field definitions and special instructions.
2. A functional test system ready for integration testing. This system will support both level 1 and level 2 queries, though will be limited in some functional ways (ie, promotions will likely not be fully implemented, but will be complete in terms of interface, so that work will not have to be re-done on the [REDACTED] side)
3. A live production environment ready for service prior to the culmination of UAT.

Other relevant services by ITA will be provided in support of these basic deliverables, such as integration of critical data feeds, server and network deployments, pricing consultation, reporting, project management and work covered by a Service Level Agreement (SLA) to be agreed upon by ITA and [REDACTED].

2.3 Project Milestone Delivery Timeline

The following milestone dates for the Affinity Search project has been identified.

- [REDACTED]

These milestones will be tracked in a comprehensive project plan delivered weekly to [REDACTED] with the ITA weekly status report and issue log.

It should be noted that the above timeline dates are independent of the implementation of the [REDACTED] and [REDACTED] projects (see section 6).

2.4 Relevant Documents

Further description of business goals and general requirements can be found in the following documents, available upon request from pfernandez@itasoftware.com:

RFP_AffinitySearch.doc, dated 9 April 2010

PID Inspire – Affinity Search_v1 draft, dated 5 October 2010

User interface and ITA data v0.12.mpp, dated 29 September 2010

These documents were supplemented by in-person meetings in [REDACTED] and Cambridge to solidify requirements and make agreements on the implementation. This document is the product of these sources.

2.5 General Assumptions

The following assumptions are made by ITA in composing this solution document and planning the Affinity Search project.

- [REDACTED] are responsible for the implementation of the front end application, and collaborating with a design firm to create the user interface and all graphical elements
- [REDACTED] front end applications will directly access two separate ITA API services, one for cached queries and one for live QPX queries (see section 3.2 for details)
- [REDACTED] has decided to forego the development of an alternate level 1 interface, as ITA and [REDACTED] will make every effort to represent markets in the cache that have the most commercial value. Those not present in the cache will be of low enough commercial value to be considered unnecessary for inclusion
- The [REDACTED] solution is not a necessary element to have in place for the launch of Affinity Search. The [REDACTED], however, is considered a key dependency for launch (see section 6 for a list of key dependencies)
- Management of this document and the project as a whole will be conducted as per the governance model outlined in the project charter document

2.6 Success Criteria

[REDACTED] have identified the following criteria which will be factored into their consideration of the success of the Affinity Search project.

- Affinity Search services must be available 99,5%
- Trip availability must match with the [REDACTED] and [REDACTED] websites at a rate of at least 90%
- Fare/Quote information (including taxes and surcharges) must be accurate up to an equivalent of 5 euro difference
- Promotion data must be 95% accurate with a backlog of max 1 day.
- System performance meets the requirements outlined in the original RFP
- Positive internal feedback by users [REDACTED] website
- Response times of services must be less than or equal to 3 seconds

2.7 Change Control

Any changes to the scope as outlined in this document after official signoff must go through a formal change request process, as described in the Project Charter, with the understanding that changes in functional requirements may impact the agreed-upon delivery schedule.

3 SOLUTION ARCHITECTURE

3.1 Overview

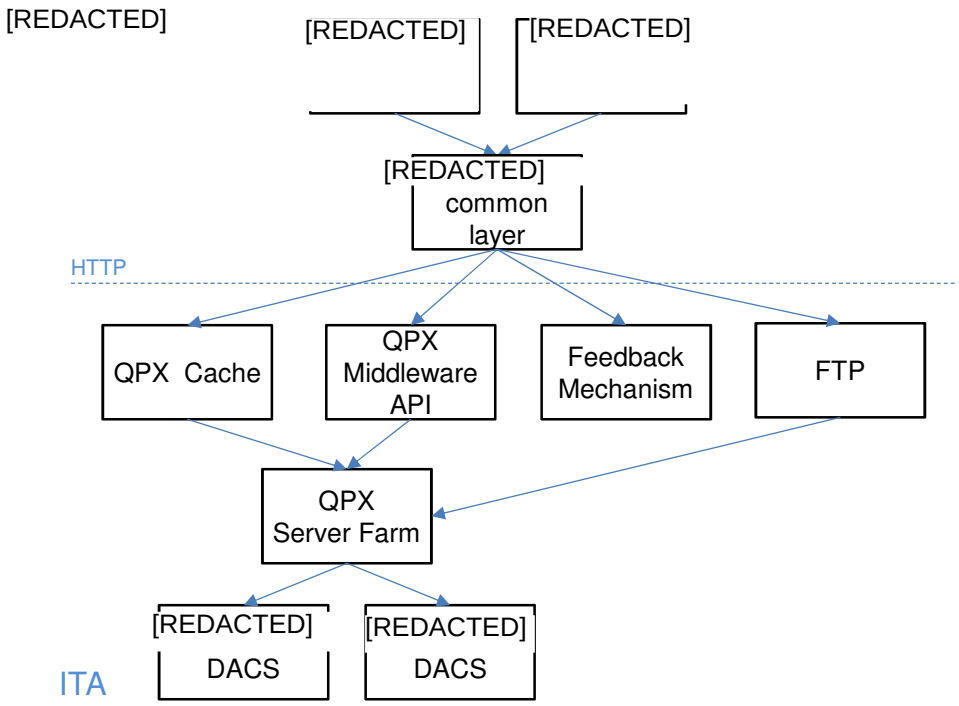

Figure 2: [REDACTED] *Affinity Search Functional Model*

The above diagram illustrates the core logical units of the Affinity Search project.

- [REDACTED] **Website** – [REDACTED] and all country sites
- [REDACTED] **Website** – [REDACTED] and all country sites
- [REDACTED] **Common Layer -** the middleware layer that both carrier websites will use to communicate to ITA systems
- **QPX middleware API** – High-level XML interface that sends live queries to the QPX Server Farm
- **QPX Cache** – the database of collected airfare pricing information populated by an automated process that queries the QPX Server Farm on a regular basis. It is accessed via an HTTP API
- **QPX Server Farm** – the collection of QPX Execution Units (EU's) that respond to airfare pricing queries from the live QPX interface and automated cache process. Accepts and returns XML
- **Feedback Mechanism** – the specialized XML API for providing ITA comparative itineraries between ITA and the [REDACTED] host system for solution quality evaluation
- **FTP** – an ITA-hosted FTP server to which [REDACTED] will transfer custom data files (see section 5)
- [REDACTED] **DACS** – provides availability information for [REDACTED] and its partners to QPX queries for use in returning pricing solutions. Similar to the [REDACTED] DACS, this is being implemented by ITA under a separate, but closely integrated project.

- [REDACTED] **DACS** – provides availability information for [REDACTED] and its partners [REDACTED] to QPX queries for use in returning pricing solutions

The architecture and functions of each unit are described in detail in the following sections.

3.2 Service Model

All servers and data needed to return airfare pricing solutions for Affinity Search will be hosted by ITA and accessed by [REDACTED] via internet-based connectivity. Additional data specific to [REDACTED] will be provided to ITA and integrated into the functions of the pricing engine and related applications. Uptime and performance of all ITA services is subject to the Service Level Agreement (SLA), described in the Master Services Agreement (MSA) between ITA and [REDACTED].

3.2.1 QPX middleware API

[REDACTED] will use the QPX middleware API, also known as “Boombox”, to send queries to live QPX servers through a specified custom XML interface. The API will run one or more QPX queries and return aggregated results directly to the calling application.

3.2.2 QPX Cache

The QPX Cache is designed to return highly compact airfare pricing information very quickly, to fulfill [REDACTED] requirement to be able to retrieve results in less than three seconds. The QPX Cache will maintain a collection of availability-checked fare market data pulled from QPX that will be used to populate the responsive, interactive interface for Affinity Search. The cache will be tailored for [REDACTED] to best advance their business goals, and be updated on a schedule that optimizes server usage, solution quality and the amount of data that can be stored.

The QPX Cache will be populated based on the following basic parameters:

- A list of points of sale & markets (origins & destinations) specified by [REDACTED] to be included in the cache, which will be shared by the two carriers.
- A total of 300,000 solution sets, across both brands and all points of sale, will be represented in the cache. In this context, a solution set is defined as a unique combination of point of sale, origin and destination and carrier. This number is based on a consistent cache refresh rate of 48 hours for every solution set in the cache, and is subject to reduction if [REDACTED] elects to refresh some markets more frequently than every 48 hours.
- For each solution set included in the cache, all trips departing between the day the query takes place and the following 180 days will be included in the cache
- For each solution set included in the cache, and each departure date, all trips that have a length of stay between 1 and 31 days will be included in the cache
- Only round-trip, single adult passenger solutions are considered

For each of the 300,000 specified solution sets, the cache will contain a price for each departure date and each trip duration. The price may be simply the ticket price, or it may include taxes or taxes and booking fees, depending on the POS and market. It is up to [REDACTED] to indicate which markets / points of sale require which kind of price data. In addition, each market, POS, departure date and trip duration cell will include a Boolean indicator whether the price corresponds to a promotional fare.

3.2.3 QPX Server Farm

Affinity Search airfare pricing solutions will be launched into production running the current release version of QPX, running on servers hosted by ITA. As determined necessary, ITA will integrate new data sources for information required to properly return [REDACTED] solutions, such as schedules, taxes, and private fares. When possible, ITA will integrate custom [REDACTED] data, such as business rules that affect what schedules and fares may be included in results. See section 5 for descriptions of all required custom data.

ITA will also endeavor to fulfill [REDACTED] requirements to match the pricing of [REDACTED] host system. ITA will implement a mechanism to monitor solution quality to aid in this effort, described in section 4.3.

3.2.4 [REDACTED] and [REDACTED] DACS

To supplement the Affinity Search pricing solutions with optimal availability data, [REDACTED] and [REDACTED] are implementing dynamic inventory calculation solutions with ITA. The [REDACTED] DACS is a key dependency for the Affinity Search project. The availability systems for [REDACTED] and [REDACTED] are being implemented by ITA under separate, but closely-integrated projects, with their own timelines and deliverables.

4 SOLUTION REQUIREMENTS

4.1 Overview

The Affinity Search application's basic feature is to allow the traveler without specific journey plans to select from a wide variety of options for destinations over a wide date range. The selection process takes place through 2 basic interfaces, coined "level 1" and "level 2", both powered by queries to ITA services. ITA's interaction ends when the traveler has picked an origin, destination, and travel dates, though full itinerary and fare information is sent back via ITA's feedback mechanism (section 4.3) for quality assurance purposes. The [REDACTED] website interface will then use those parameters to pick specific itineraries and pricings based on direct queries to the [REDACTED] host system's [REDACTED] product.

4.2 Level 1 Query

The "level 1" query serves as the point of entry into Affinity Search, and is the dominant feature of the application, using Google Maps as a key component of the interface. The traveler, having entered the website via the country site pick list, is presented with an origin list to pick from. Once one or more origins is selected, a query is sent to ITA to present an array of destinations available for that origin, represented by markers on the geographical locations of those

destinations in the Google Maps interface. Input controls in the level 1 interface allow the traveler to refine the results by date of departure, length of stay, and maximum price.

The level 1 query results will be populated by a query to the QPX Cache, which will return the lowest price for each destination in the departure date range with no greater than 800 destinations (and prices) returned per query. Each set of solutions will contain one solution per destination, regardless of how many origins are entered as input.

Solutions from the QPX Cache will be processed by the application layer of the carrier's websites to display, for each point on the map, the total number of destinations for that map region, as well as the lowest price for all destination / date combinations for that region.

4.2.1 Interface Definition

4.2.1.1 Connectivity

ITA Cache Search is accessed through a stateless protocol a la REpresentational State Transfer (REST) architecture. This means that all query parameters, which specify the requested information, need to be included with each query. This is also the case with HTTP connections that allow multiple queries per connection, utilizing the HTTP 1.1 keep-alive feature.

4.2.1.1.1 HTTP Request

Requests for information are presented to the ITA Cache Search servers using the HTTP protocol over the TCP/IP stack. Clients of ITA Cache Search access the information through a specified IP address resolving to the **cachesearch.itasoftware.com** domain name. The current server supports parts of the HTTP 1.0 and HTTP 1.1 protocol. An HTTP 1.1 request looks like the following:

```
GET /prices.xml?from=BOS&to=ORL&minDepartureDate=2010-08-
14&maxDepartureDate=2010-09-
14&minTripLength=14&maxTripLength=21&POS=[REDACTED] HTTP/1.1
Host  cachesearch
User-Agent  Mozilla/5.0
Accept-Encoding   gzip,deflate
Keep-Alive  300
Connection  keep-alive
Referrer    http://cachesearch/demo/destinationSearch.html
```

The only accepted HTTP method is GET, where the absolute path of the resource is followed by a question mark and ampersand delimited parameters. Interpreted HTTP headers include Accept-Encoding, Keep-Alive, Connection and, Referrer.*Accept-Encoding* may specify that the client wishes for a compressed version of the data which reduces the data flow and transmission times. *Keep-Alive* and *Connection* specify that the connection to the server will be maintained, reducing the number of connection attempts and latencies due to TCP's three way handshake. *Referrer* is used for accounting in the case of the XSS usage scenario.

4.2.1.1.2 HTTP Response

The ITA Cache Search service may respond with any of the following HTTP Status Codes:

Status Code		Description
200	**OK**	Success
304	**Not Modified**	There was no new data to return.
400	**Bad Request**	The request is invalid! The server includes a short explanatory message.
401	**Not Authorized**	Either the client needs to provide authentication credentials, or the credentials provided were not valid.
403	**Forbidden**	This is a valid request, but ITA Cache Search refuses to fullfill it. The server includes an explanatory error message.
404	**Not Found**	Either the requested URI is invalid, the resource in question doesn't exist, or the information for the query parameters specified wasn't found.
500	**Internal Server Error**	The servers could not serve the request. The error has been noted and will be investigated.
502	**Bad Gateway**	The servers are down or being upgraded.
503	**Service Unavailable**	The servers are up running but are overloaded with requests.

Most error codes are accompanied by an explanatory error message. In the case when everything was fine, the response may look like this:

```
HTTP/1.1 200 OK
Date: Fri, 17 Oct 2008 17:55:29 GMT
Server: IS.API/2.x.x
Content-Type: application/xml
Keep-Alive: timeout=60
Connection: Keep-Alive
Content-Length: 147

<results count="1" version="2.x" type="prices">
  <solution price="228.50" from="BOS" to="ORL" departs="2008-11-
01" returns="2008-11-15"/>
</results>
```

This indicates a few things:

- ITA Cache Database Server version was *IS.API/1.3.1*.
- The content served back to the user has the MIME type of *application/xml*.
- The server will keep the connection alive for *60* seconds and expects the client to do the same.

- The content, which follows after one empty line terminated with LF/CR, is *147* bytes long.

4.2.1.2 Syntax

4.2.1.2.1 The Prices Query

When looking at the URI used to access the ITA Cache Search service, the first part after the host name and before the query parameters, as marked by a question mark, is the access method.

```
http://cachesearch/prices.xml?from=BOS&to=ORL&tripLength=14
```

4.2.1.2.2 Parameters

When looking at the URI used to access the ITA Cache Search service, the part after the host name, access method, and question mark, is called the query string. This query string includes the query parameter. In the example below, there are four query parameters with their values: *from, to, departureDate, tripLength*.

```
http://cachesearch/prices?from=BOS&minDepartureDate=2009-11-
02&maxepartureDate=2009-12-
15&minTripLength=7&maxTripLength=14&pos=[REDACTED]
```

Each query parameter is a key-value pair, where the key is separated from the value with an equal sign, '='. The key-value pairs are delimited by an ampersand, '&', or by a semicolon, ';'. The interpretation of each value depends on its associated key. With some parameters, the server takes only the first value specified. With other parameters the server will accept multiple values as specified after the equal sign (such as origins or destinations). Multiple values to a parameter can be separated with space (which is represented as '%20' or '+' when URL encoded) or with a comma, ','. Instead of listing multiple values under one key, the client may specify multiple key-value pairs, each of varying length. The resulting set of values will be the union of all value sets for all key-value pairs for the given key. The server accepts properly URL encoded query lines, where special characters like the space are escaped using the URL encoding schema.

4.2.1.2.3 Encoding

ITA Cache Search servers can produce data at a rate much higher than usual hard disc writes and network throughput. Since transmission times are the main part of time spent before results of the query are available to the client, the ITA Cache Search servers usually encode the response using GZIP as specified by the HTTP standard. This behavior is regulated by the client request HTTP header line stating with *Accept-Encoding*. The presence of the *gzip* flag in the value instructs the server to send gzipped responses. The server will not encode the response if this flag is missing, or if the size of the content is below some specified threshold (usually 1 KB). When the server does send gzipped content, the HTTP response header will include*Content-Encoding: gzip* The debug GET parameter *?gzip=false* will override the default browser-server behavior and switch off the gzip encoding.

4.2.1.2.4 Authentication

The ITA Cache Search server supports the following authentication schemata. For a proxy based usage scenario the client will be authenticated by their IP address, access port on the IS Server, and assigned clientID.

An additional level of security can be attained by using a secret exchange schema, where the web page authentication server holds a mutual secret with the ITA Cache Search servers and uses the secret to authenticate online clients by securely hashing the secret with the online client IP. This authentication schema is especially useful in a XSS usage scenario, when the simple HTTP referrer based authentication is not sufficient for the customer.

4.2.1.2.5 Input

The following input parameters can be included in the level 1 query to the QPX Cache:

PARAMETER	NAME	[REDACTED] Mapping	SAMPLE	M?	DESCRIPTION
BRAND	**N/A**	brandCode	N/A	N/A	This will map to the port of the QPX Cache server constructed to accept requests for that brand
ORIGIN	**From**	Station->code	[REDACTED]	Yes	Comma or space separated three letter city or airport codes of the origin location of the first slice.
DEPARTURE DATE RANGE – BEGIN	**minDepartureDate**	DeparturePeriod->startDate	2010-12-01	Yes	A date specifying the beginning of range of requested dates of departure
DEPARTURE DATE RANGE – END	**maxDepartureDate**	DeparturePeriod->endDate	2010-12-31	Yes	A date specifying the end of range of requested dates of departure.

POINT OF SALE	**Pos**	PointOfSale->officeID	[REDACTED]	Yes	Valid office ID
MAXIMUM PRICE	**maxPrice**		1-…	No	The maximum price in the currency designated for the point of sale
DAYS OF STAY RANGE – BEGIN	**minTripLength**		1 … 60	Yes	A number specifying the minimum of the range of trip durations in days used for the trip.
DAYS OF STAY RANGE – END	**maxTripLength**		1 … 60	Yes	A number specifying the maximum of the range of trip durations in days used for the trip.
DEPARTURE DAY OF WEEK	**departureDayOfWeek**	DeparturePeriod->daysOfWeekDeparture	0 - 6 (modulo 7 days of a week; 0 = Sunday)	No	One or more valid days of week
RETURN DAY OF WEEK	**returnDayOfWeek**	DeparturePeriod->daysOfWeekReturn	0 - 6 (modulo 7 days of a week; 0 = Sunday)	No	One or more valid days of week

4.2.1.2.6 Input Example

```
http://cachesearch/prices.xml?from=[REDACTED]&minTripLength=18&maxTripLength=24&minDepartureDate=2010-11-01&maxDepartureDate=2010-12-31
```

4.2.1.2.7 Output

Results are presented in XML. The root element is *results*, and the attribute *count*describes the number of solutions returned. The attribute *type* is the direct reflection of the method used for the search.

Each solution will include the attribute *price*, specifying the total sale price of the solution in the currency associated with the point of sale. The attributes *from* and *to* specify locations of the

origin and destination of the trip. The cities are uppercase the airports are lowercase three letter acronyms. The attributes *departs* and *returns* specify the departure and return dates of the solution in the time zone of the trip origin and destination, respectively.

The following fields will be returned from the QPX Cache for each market in a Level 1 query. Note that the level 1 query returns the lowest price for each market across the entire date range searched:

PARAMETER	**NAME**	[REDACTED] **Mapping**	**SAMPLE**	**DESCRIPTION**
ORIGIN	Results/solution@from		[REDACTED]	3-letter city code of trip origin
DESTINATION	results/solution@to		[REDACTED]	3-letter **city** code of trip destination
CURRENCY CODE	results/solution@currency	currencyCode	EUR / USD	3 letter currency code used in the solution's price
PRICE	results/solution@price	Price	500	Numeric value of cost of solution
PROMOTION FLAG	results/solution@promo	promotionCode	YES / NO	Boolean indicating whether solution is a promotional fare

4.2.1.2.8 Output Example

The response is excerpted, as there would normally be many more rows:

```
<results count="416" version="ITA.CS.API/v2.x.x" type="prices">
<solution price="168.00" currency="EUR" from=”AMS” to="LON"
promo="NO"/>
<solution price="956.50" currency="EUR" from=”AMS” to="cur"
promo="NO"/>
<solution price="170.50" currency="EUR" from=”LON” to="MAD"
promo="NO"/>
<solution price="268.50" currency="EUR" from=”AMS” to="ROM"
promo="YES"/>
<solution price="143.50" currency="EUR" from=”PAR” to="bcn"
promo="NO"/>
<solution price="236.50" currency="EUR" from=”AMS” to="gva"
promo="NO"/>
<solution price="518.00" currency="EUR" from=“LON” to="NYC"
promo="NO"/>
<solution price="255.50" currency="EUR" from=”PAR” to="MIL"
promo="NO"/>

etc...

</results>
```

4.2.1.2.9 Error Handling

Errors are signaled using HTTP response status codes or using an error response content. Errors 'value error' caused by incorrect parameter values and 'zero results' caused by incorrect query specification are signaled using the HTTP 200 OK status and a proper error content for output formats: XML, JSON, and JS.

Example XML: /prices.xml?from=B0S

```
<results count="0" version="2.x.x" type="prices">
  <error type="value error" key="from" value="B0S" reason="value
is not a proper location code"/>
</results>
```

The condition *value error (Bad Request)* is only signaled if the ITA Cache Search server detected that the given set of parameters doesn't pose a valid query. This is done only at parameter validation time. The condition *zero results (Not Found)*signals that the ITA Cache Search server could not find any results, because it could prove that the given parameters would yield an empty set of search results.

HTTP code *Internal Server Error* is returned, when the server was not able to perform the search due to unspecified reasons. The connection to the client is then closed even if a keep-alive flag was set. *Internal Server Error* is always signaled at the HTTP protocol level using HTTP response code. The client application should also handle the case when the server doesn't respond to the request and doesn't close the connection.

4.2.2 Level 2 Query

The level 2 query narrows the result set to the destination selected in level 1, and returns the cheapest solution for each day in a range of one full month of departure days for the origin and destination specified. A price will be returned for each of seven days of stay.

If the GUI will contain multiple calendar months in its display, an additional query will be needed to retrieve data for each month. ITA recommends that multiple queries be performed serially to optimize server usage.

The level 2 interface will employ direct queries to QPX to return fully-priced, availability-checked pricing solutions. Queries will be processed via the QPX middleware, and one or more queries sent to QPX. Results will be collected by the middleware and returned to the calling application.

Settings to be applied to these queries include:

- Only round-trip solutions are considered
- A range of dates for days out from the day the query was made, to 330 days into the future, with a 31 day range in each query.
- Lengths of stay can be specified in a range not exceeding 7 days. The range can begin or end anywhere between 1 and 330 days (with the understanding that solutions likely will not exist for ranges past 330 days)
- Multiple passengers and types will be permitted

- Solutions are returned for all date combinations that conform to the 7-day maximum layover range

4.2.2.1 Interface Definition

4.2.2.1.1 Connectivity

The Boombox API server supports XML requests over HTTP. HTTP response codes and session handling are the same for all protocols. The HTTP header "Accept-Encoding: gzip" is required, and enables gzip'd output for responses.

4.2.2.2 Input

The following values can be passed to the level 2 query for consideration in QPX results.

PARAMETER	NAME	[REDACTED] Mapping	SAMPLE	M?	DESCRIPTION
ORIGIN	inputs@origin	Station->Code	[REDACTED]	Yes	3-letter airport code or city code to use as journey origin
DESTINATION	inputs/destination	Station->Code	[REDACTED]	Yes	3-letter airport or city code to use as journey destination – maximum of 20 allowed
PASSENGER TYPE	inputs/passengers	PaxType->code/ PaxType->number	ADT, INF, CNN, SRC	Yes	One or more passenger type codes, one for each passenger to be included in the search
DEPARTURE DATE	inputs/departure	DeparturePeriod->startDate	2010-11-12	Yes	Beginning date for 30 day range of trip departure
POINT OF SALE	inputs@officeID	PointOfSale->officeID	[REDACTED]	Yes	Valid Office ID
LENGTH OF STAY – MIN	inputs@minStay	Stay->minNumberOfDays;	Integer	Yes	Integer value representing minimum desired number of days of stay

LENGTH OF STAY – MAX	inputs@maxStay	Stay->maxNumberOfDays	Integer	Yes	Integer value representing maximum desired number of days of stay. Note difference of max length of stay and min length of stay must not exceed 7 days
BRAND	search@key	Brand->brandCode	[REDACTED]: ymRz0AbZICqLLtuFnVqhPs [REDACTED]: iaLFQhmTxgInrVDwazFNFu	Yes	Unique key identifying the Boombox query to be used – this will be individual between [REDACTED] and [REDACTED]
SUMMARIZER	search/summarizer	N/A	pricePerDate	Yes	The pre-configured output mask that will be used to return results.

4.2.2.3 Input Example

```
<?xml version="1.0" encoding="UTF-8"?>
<search key="EdQCLyJBTlAfLwNtIFRTU1" name="multiDate" version="?">
  <inputs departure="2008-10-27" maxStay="9" minStay="3"
officeID="test123" origin="BOS">
    <destination>[REDACTED]</destination>
    <passengers adults="1"/>
  </inputs>
  <summarizer>pricePerDate</summarizer>
</search>
```

4.2.2.4 Output

The following values will be returned in the level 2 query results.

PARAMETER	NAME	[REDACTED] Mapping	SAMPLE	DESCRIPTION
DEPARTURE	result/pricePerDate/depart	DepartureDate-	YYYY-MM-DD	Date and time of

DATE	ure@date	>departuredate		trip departure
LENGTH OF STAY	result/PricePerDate/departure/options/return/lengthOfStay		5	Integer indicating duration of stay at destination
PRICE	result/pricePerDate/departure/options/return/solution/displayTotal@amount	Calendar->minimumPrice	321.40	Price in local currency of the cheapest solution to the entry's destination, displayed according to price detail instructions (see section 5.9)
CURRENCY	result/pricePerDate/departure/options/return/solution/displayTotal@currency	Calendar->currencyCode;	EUR	3-letter code of the currency in which price is display
PROMOTION	solution/ext/@promo	ReturnDate->promotionCode	Alphanumeric	Indicates whether the cheapest solution in this market is a special marketing promotion (see section 4.5.3 for details)
SOLUTION ID	result/PricePerDate/departure/options/return/solution/Id		Alphanumeric	Unique identifier for a specific solution, used in the Get Itinerary query to retrieve complete solution information
SOLUTION SET	result/solutionSet		Alphanumeric	Unique ID for this set of solutions, stored in the Boombox session

4.2.2.5 Output Example

```
<?xml version="1.0" encoding="UTF-8"?>
<result id="?" session="?" solutionCount="217" solutionSet="?">
```

```
<pricePerDate>
  <departure date="2008-10-27">
    <options>
      <return lengthOfStay="3">
        <solution id="00D">
          <displayTotal amount="2210.37" currency="EUR"/>
          <ext promo="ABC"/>
        </solution>
      </return>
      <return lengthOfStay="4">
        <solution id="00T">
          <displayTotal amount="1373.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="5">
        <solution id="00T">
          <displayTotal amount="1373.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="6">
        <solution id="00P">
          <displayTotal amount="609.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="7">
        <solution id="00H">
          <displayTotal amount="592.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="8">
        <solution id="00J">
          <displayTotal amount="592.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="9">
        <solution id="00N">
          <displayTotal amount="592.37" currency="EUR"/>
        </solution>
      </return>
    </options>
  </departure>
  <departure date="2008-10-28">
    <options>
      <return lengthOfStay="3">
        <solution id="00E">
          <displayTotal amount="2210.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="4">
        <solution id="00S">
          <displayTotal amount="1373.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="5">
        <solution id="00K">
          <displayTotal amount="609.37" currency="EUR"/>
        </solution>
      </return>
      <return lengthOfStay="6">
```

```
            <solution id="00J">
              <displayTotal amount="592.37" currency="EUR"/>
            </solution>
          </return>
          <return lengthOfStay="7">
            <solution id="00F">
              <displayTotal amount="592.37" currency="EUR"/>
            </solution>
          </return>
          <return lengthOfStay="8">
            <solution id="00K">
              <displayTotal amount="592.37" currency="EUR"/>
            </solution>
          </return>
          <return lengthOfStay="9">
            <solution id="00O">
              <displayTotal amount="592.37" currency="EUR"/>
            </solution>
          </return>
        </options>
      </departure>
...
```

4.2.2.6 Get Itinerary

The level 2 query has a follow-up query available to retrieve all information about a specific solution. This will be used to pass information from the carrier website layer to the ITA Feedback Mechanism, described in the next section.

Some notes about the Get itinerary query:

- Returns details on a specific solution
- Uses session and solution set from Level 2 query
- solution input is the solutionSet/solutionId from the previous response

4.2.2.6.1 Input

PARAMETER	NAME	[REDACTED] Mapping	SAMPLE	M?	DESCRIPTION
KEY	summarize@key		Alphanumeric	Yes	The unique key to identify the source of the query. This will be individual to [REDACTED] and [REDACTED]
SESSION	summarize/session		Alphanumeric	Yes	The unique session id used in the initial level 2 query, to ensure that this

					followup query refers to the correct set of results
SOLUTION SET	summarize/solutionSet		Alphanumeric	Yes	Unique ID for the set of solutions returned in the intial level 2 query
SOLUTION	summarize/inputs/solution		Alphanumeric	Yes	The unique ID for the specific solution for which details are being requested
SUMMARIZER	summarize/summarizer		Details	Yes	Required, the name of the output mask to be used to return itinerary details

4.2.2.6.2 Input Example

```
<?xml version="1.0" encoding="UTF-8"?>
<summarize key="EdQCLyJBTlAfLwNtIFRTU1" session="?" solutionSet="?"
version="?">
  <inputs solution="?"/>
  <summarizer>details</summarizer>
</summarize>
```

4.2.2.6.3 Output

PARAMETER	NAME	[REDACTED] Mapping	SAMPLE	DESCRIPTION
ID	result@id		Alphanumeric	Unique ID of the specific solution
SESSION	result@session		Alphanumeric	Session ID from initial level 2 query
SOLUTION SET	result/SolutionSet		Alphanumeric	Unique ID of the solution returned
SOLUTION	result@solutionCount		Numeric	The ordinal of the

COUNT				solution in the original level 2 query
ORIGIN	result/details/itinerary/slice/segment/origin		[REDACTED]	3 letter airport code for flight origin
DESTINATION	result/details/itinerary/slice/segment/destination		[REDACTED]	3 letter airport code for flight destination
DEPARTURE DATE	result/details/itinerary/slice/segment/depaurture		2010-10-27T3:00-4:00	Departure date and time of flight
RETURN DATE	result/details/itinerary/slice/segment/returnDate		2010-10-27T3:00-4:00	Return date and time of flight
FLIGHT DETAIL	result/details/itinerary/slice/segment/flight		[REDACTED]	Carrier code and flight number of flight
TOTAL PRICE	result/details@displayTotal		EUR500.00	Complete total price of solution
BOOKING CODE	result/details/pricing/fare/bookingInfo@bookingCode		Y / J / C …	One-letter booking code
FARE	result/details/pricing/fare/bookingInfo@fareCode		Alphanumeric	Fare basis code used in solution
PASSENGERS	result/details/pricing/passenger/ptc		ADT, INF, CNN, SNR	Valid passenger type code; this will repeat for each passenger
EXT EBT	result/details/ext@ebt		Alphanumeric	URL-encoded string of all necessary data sufficient to be passed to the feedback mechanism

4.2.2.6.4 Output Example

```
<?xml version="1.0" encoding="UTF-8"?>
<result id="?" session="?" solutionCount="217" solutionSet="?">
  <details>
    <displayTotal amount="2210.37" currency="EUR"/>
```

```
    <ext ebt="?%2C1%2C1%2C2%2CS%2CYEERT%2CS%2CYEERT%2C1%2C1%2CADT%2C2%2C2008-
10-27T23%3A00-04%3A00%[REDACTED]%2CBOS%2C2008-10-
30T15%3A10%2B01%3A00%2CBOS%2C[REDACTED]"/>
    <itinerary>
      <slice>
        <segment departure="2008-10-27T23:00-04:00" destination="[REDACTED]"
origin="BOS"/>
      </slice>
      <slice>
        <segment departure="2008-10-30T15:10+01:00" destination="BOS"
flight="[REDACTED]"/>
      </slice>
    </itinerary>
    <pricing>
      <fare>
        <bookingInfo bookingCode="S" fareCode="YEERT"/>
      </fare>
      <fare>
        <bookingInfo bookingCode="S" fareCode="YEERT"/>
      </fare>
      <passenger>
        <ptc>ADT</ptc>
      </passenger>
    </pricing>
  </details>
</result>
```

4.2.2.7 The "Clear" Query

ITA recommends sending the clear query at the time the calling application clears out a shopper's web session. This is an optional step.

```
POST /services/xml HTTP/1.1
Content-Type: text/xml
...
<clear key="FqDazRWkytwzuALGYwtnOE"
       verstion="17"
       session="9xGJernoIzK28jE8UEUkwS29Z0000LaJ3WSw6RDB0"/>
```

4.2.2.8 Level 2 / Get Itinerary Error Handling

HTTP responses from Boombox will include one of the following status codes:

- 200 - Success
- 400 - Invalid parameters
- 500 - Internal error
- 503 - Capacity exceeded

Responses with status codes 200, 400, and 500 contain:

- an error type,
- an optional error code,
- a textual description of the error, and
- a result ID.

500-level errors indicate Boombox encountered an error. 400-level errors indicate that there is an issue on the [REDACTED] side, such as submission of bad data (e.g., an airport code that we know nothing about).

4.2.2.8.1 Error Example

```
HTTP/1.1 400 Bad Request
Content-Type: text/xml
...
<?xml version='1.0' encoding='UTF-8'?>
<result id="13DeOm26f0000LaJ1qaB7"
    solutionSet="sx0A3TdJVU6bSVSjuFRXWAOtA"
    session="9xGJernoIzK28jE8UEUkwS29Z0000LaJ3WSw6RDB0">
  <error type="input">
    <message>
QPX Warning. Latest departure time of first slice (2007-11-
09T05:59 UTC) is in the past; current time is 2008-11-07T14:08
UTC.
    </message>
  </error>
</result>
```

More description of errors is available in Section 9, QPX errors.

4.3 Quality Feedback Mechanism

When a traveler selects a departure and return date in Level 2, the browser exits the Affinity Search application and transfers control to the booking tool. The booking tool accepts values passed from the Level 2 query page to display the best solutions for the included parameters. It is a key element of the Affinity Search application that at least one solution be available in EBT at the price quoted by ITA for each Level 2 to EBT transition. Based on the RFP, the goal is for the price for the dates chosen in the level 2 query to match the EBT lowest price 90% of the time with a tolerance of 5 Euros.

ITA will implement a service that will enable [REDACTED] application layer to transmit and record the cheapest ITA solution returned for an O&D date combination, as well as the corresponding cheapest solution from the [REDACTED] host system. This will allow monitoring of the following states:

- ITA and [REDACTED] match prices.
- [REDACTED] returns a cheaper price than ITA.
- [REDACTED] returns a more expensive price than ITA.
- ITA itinerary not available in [REDACTED]

This mechanism is based on ITA's Booking Information Stream (BIS) technology, which was designed to allow ITA to audit bookability of solutions returned. It is therefore ideally suited for this purpose, as even though the ITA price will not be used for booking, the demands of the Affinity Search application in terms of matching prices is just as high.

4.3.1 Interface Definition

4.3.1.1 Connectivity

To communicate with the BIS, the customer application will send data to ITA's server via an authenticated HTTP POST request. The BIS server will first log all data, then attempt to validate formatting.

- If data validation succeeds, the server will respond with an HTTP 200 OK status.
- If validation fails, the server will respond with an HTTP 400 BAD REQUEST status.
- Any other status code indicates that the server did not successfully process the request, and if possible it should be repeated.

ITA offers access to the BIS over public internet. Customer applications can submit to **https://itabis.itasoftware.com/report** once login credentials have been issued. ITA can also offer access to the BIS via a customer's private lines. Please contact the ITA account service manager to coordinate this.

The body of each BIS client request should consist of HTML form urlencoded data, containing any number of name-value pairs, each with name `a`. Each value should contain a record consisting of a single line of comma-delimited ASCII text, encoding a sequence of fields as specified below.

4.3.1.2 Syntax

The new XML interface, designed with [REDACTED] in mind, is currently under final review at ITA and will be communicated to [REDACTED] at the earliest possible time.

4.3.1.3 Examples

Example feedback submissions will be communicated at the time of the delivery of the new XML interface.

4.4 Special Considerations

As part of the Affinity Search application, ITA will extend QPX to support custom business logic to meet [REDACTED] requirements as follows in the below sections.

4.4.1 Business Rule data

ITA will accept custom data files with [REDACTED] business rules, and use those files to guide QPX in generating the correct fare market prices., These business rules determine the schedules and fares to be included in QPX solutions. See section 5 for a listing of the data elements needed.

4.4.2 Booking Fees

QPX will require enhancements to support booking fees to be provided by [REDACTED]. These fees will be added to the total price for QPX solutions when appropriate.

Please see section 5.10 for a description of the data fields needed to properly implement booking and ticketing fees.

4.4.3 Promotions

QPX will need to be enhanced to support custom functionality to indicate whether the price of a QPX solution is a special marketing promotion. This information will be fed to QPX via multiple processes: For [REDACTED], ITA will use ATPCO filed data. For [REDACTED], ITA will receive a data file via FTP.

Please see section 5.11 for a description of the data fields needed to properly implement the display of promotions indicators.

5 DATA REQUIREMENTS

Apart from the standard data feeds that are essential to QPX processing, from sources such as OAG, ATPCO and IATA, the Affinity Search application will require additional data specific to [REDACTED] to process solutions in accordance with requirements. The following table illustrates a normalized, logical model of the data ITA will need, including the source and brief descriptions of relevant fields. The below tables do not represent a definitive data model that ITA plans to implement, but are meant to enumerate the basic data requirements that need to be fulfilled for the Affinity Search queries to run properly. Please note that the processes used to update this information are still to be determined.

5.1 Data Update Process

ITA and [REDACTED] have agreed on a process to employ for updating of this data. ITA will generate sample comma-delimited data files based on the field structure itemized below. This data will be imported by [REDACTED] into an admin tool they plan to implement, which will

allow business owners at [REDACTED] to manipulate the entries. Updated data files will then be exported from the [REDACTED] admin tool and transmitted to ITA via FTP. ITA automated processes will detect new files at the FTP server, and import the data into the Affinity Search services. This approach is effective for all data indicated below with the exception of [REDACTED] booking fees and promotions information, which will be derived from ATPCO data.

5.2 Data Update Specifications

The mechanism for receipt of custom data files is under final review at ITA, and will be communicated to [REDACTED] at the earliest possible time.

5.3 Origins

The origins table describes the various points of departure to be included in the QPX Cache. The information in this table will control multiple elements of how markets are represented in the cache data.

Source: [REDACTED] business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office id of origin
AIRPORT / CITY CODE	[REDACTED]	3 letter airport or city code
PRIORITY	0-10000	Indicator of relative importance of origin for inclusion in cache – a higher number indicates higher priority; This value is used to determine the order of updating the cache
COVERAGE	0-10000	Indicator of relative emphasis of market in cache – higher number indicates more markets will be saved in the cache related to other markets with lower numbers; it has been agreed that this number will be the same across all markets at launch, as for any market represented in the cache, all possible destinations is desired
POLLING	FAST / SLOW	Indicates cache data refresh rate

5.4 Destinations

In conjunction with the Origins data, the Destinations information indicates which locations will appear as options in level 1 query results

Source: [REDACTED] business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
AIRPORT / CITY CODE	[REDACTED]	3 letter airport or city code
PRIORITY	1-10000	Indication of relative importance of destination in cache – higher number indicates higher priority. This number is independent of the priority in the 'origins' table

5.5 Blackout Journeys

Blackout journeys represent markets that must be excluded from the cache

Source: [REDACTED] Business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office ID or "World", which indicates the record is relevant for all office ids
ORIGIN	[REDACTED]	3 letter city or airport code
DESTINATION	[REDACTED]	3 letter city or airport code
CARRIER	[REDACTED]	2-letter carrier code specific to blackout market. "All" indicates the blackout should apply to all carriers

5.6 Maximum Travel Time

Maximum travel time allows the indication of trip time limits for markets where ITA returns solutions not returned by the [REDACTED] host system due to duration of travel.

Source: [REDACTED]

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website

POINT OF SALE	[REDACTED]	Valid office ID or “World”, which indicates the record is relevant for all office ids
ORIGIN	[REDACTED]	3 letter airport or city code
DESTINATION	[REDACTED]	3 letter airport or city code
CARRIER	[REDACTED]	2-letter carrier code. “All” indicates the threshold should apply to all carriers
MINUTES	0-10000	Threshold in minutes of maximum trip durations; solutions whose total trip durations exceed this number will be suppressed from solutions

5.7 Carriers / Fares

This data represents the inventory of carriers to be supported on each branded website, how many segments can be flown by this carrier in a single itinerary, and whether or not the carriers’ fares should be supported.

Source: [REDACTED] Business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office ID (citycode) or “World”, which indicates the record is relevant for all office ids
ORIGIN	[REDACTED]	3 letter airport or city code
DESTINATION	[REDACTED]	3 letter airport or city code
CARRIER	[REDACTED]	2-letter carrier code
SEGMENTS	0-4	Count of maximum number of segments to be considered in solutions for this carrier
FARES	YES / NO	Indication of whether QPX should consider use of fares published by this carrier in addition to fares associated with the branded website’s carrier ([REDACTED] or [REDACTED])
SORT ORDER	Integer	A number that represents the enumerated order in which

		this row should be processed in relation to other rows

5.8 SPA Carriers

This information represents the inventory of carriers with interline agreements with [REDACTED]

Source: [REDACTED] Business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office ID (city code) or "World", which indicates the record is relevant for all office ids
ORIGIN	[REDACTED]	3 letter airport or city code
DESTINATION	[REDACTED]	3 letter airport or city code
CARRIER	[REDACTED]	2-letter carrier code
TRANSFER POINTS	[REDACTED]	3 letter airport or city code, representing allowed connection points for the carrier from the [REDACTED] network

5.9 SPA Connecting Points

Allowed connecting points for interline partners

Source: [REDACTED] Business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office ID or "World", which indicates the record is relevant for all office ids
CARRIER	[REDACTED]	2-letter carrier code
CONNECTING AIRPORT	[REDACTED]	3 letter airport code / 2 letter country code indicating an allowed connection point for the carrier partner within its own network

5.10 Excluded Fares

The excluded fares information informs QPX which carrier's fares should be excluded for consideration in solutions returned via the level 1 and 2 queries.

Source: [REDACTED] Business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid office ID
ORIGIN	[REDACTED]	3 letter airport or city code
DESTINATION	[REDACTED]	3 letter airport or city code
CARRIER	[REDACTED]	2-letter carrier code
FARE BASIS CODES	Alphanumeric strings	List of farebasis codes to be excluded from solutions for this office id & market. Standard ATPCO wildcard allowed

5.11 Pricing Detail

This information indicates the currency to use for each point of sale, as well as how the price should be expressed.

Source: [REDACTED] business

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website
POINT OF SALE	[REDACTED]	Valid Office ID
CURRENCY	USD / EUR	3-letter currency code to use for the point of sale
PRICE DETAIL LEVEL	F / S / B	F = Fares only; S = Fares + taxes + surcharges; B = Fares + taxes + surcharges + booking fees

5.12 Booking Fees

This information represents the additional booking fees imposed by [REDACTED] on solutions based on regions and booking codes.

Source: [REDACTED]: ATPCO S4 records; [REDACTED]: manual file feed

COLUMN	SAMPLE DATA	DESCRIPTION
BRAND	[REDACTED]	Carrier website

POINT OF SALE	[REDACTED]	Valid Office ID
ORIGIN	[REDACTED]	3 letter airport or city code / 2 letter country code
DESTINATION	[REDACTED]	3 letter airport or city code
AMOUNT	####	Numeric value of booking fee
CURRENCY CODE	USD EUR	3 letter code of currency in which fee is expressed
PASSENGER TYPE	ADT / INF	3 letter passenger type code to which fee applies
CABIN CLASS	F / C / Y	1 letter booking code to which fee applies

5.13 Promotions

The promotions data enables the indication that a solution is a special promotional offer created as part of [REDACTED] or [REDACTED] marketing initiatives.

Source: [REDACTED]: export from promotions web service; [REDACTED]: derived from ATPCO-filed rule data

COLUMN	SAMPLE DATA	DESCRIPTION
PROMOTION CODE	Alphanumeric string	Indicates unique promotional code for the offer
START DATE	Valid date	Indicates date the promotion becomes available
END DATE	Valid date	Indicates date the promotion stops being available
FARE BASIS CODE	Alphanumeric characters	Valid farebasis code to which promotion applies
TRAVEL DATE BEGIN	Valid date	Indicates valid travel begin date
TRAVEL DATE END	Valid date	Indicates valid travel end date
ORIGIN	Valid 3-letter airport code	Origin of the fare to which promo applies
DESTINATION	Valid 3-letter airport code	Destination of the fare to which promo applies

6 KEY DEPENDENCIES

The success of the Affinity Search project will be dependent on a number of factors, including the accurate definition of the features and functions considered as in scope for the initial launch. Access to subject matter experts at [REDACTED] will be critical in completing the

scoping and sizing of this project, as well as access to documentation of relevant business rules as needed.

The following additional key dependencies exist.

- A signed Master Services Agreement (MSA) and Service Level Agreement (SLA) between ITA and [REDACTED]
- Due to the quality of availability data for [REDACTED] currently received by ITA, the [REDACTED] DACS has been identified as a requirement to have in place for the rollout of Affinity Search
- Relevant business data will be delivered and updated by [REDACTED]
- Development and deployment of all elements interacting to the ITA API's will be managed by [REDACTED]
- The [REDACTED] DACS, while not critical for launch, is still considered a key dependency for the project

7 NON-FUNCTIONAL REQUIREMENTS

The following components characterize the non-functional requirements for Affinity Search as they are currently known.

- **Test environment –** ITA will deploy a test environment sufficient to enable [REDACTED] to accomplish integration testing with other components of the Affinity Search front-end applications
- **Production environment –** ITA will deploy a production environment suitable to handle anticipated traffic and performance goals, subject to Operational support outlined in the SLA, in time for [REDACTED] UAT
- **Communications lines –** ITA will work with [REDACTED] to implement the network setup necessary to enable communication between [REDACTED]'s datacenters and the ITA system
- **Quality Feedback mechanism –** ITA will implement, support and monitor the described quality feedback mechanism to facilitate assurance of price matching with [REDACTED] host system

8 XSD DEFINITIONS

8.1 Level 2 Query

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="cityOrAirportCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="error">
    <xs:sequence>
      <xs:element name="message" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="code" type="xs:string"/>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="internal"/>
          <xs:enumeration value="input"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
  <xs:simpleType name="key">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{22}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="localDate">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{4}-\d{2}-\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="outputFormat">
    <xs:restriction base="xs:string">
      <xs:enumeration value="json"/>
      <xs:enumeration value="xml"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="resultId">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{22}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="sessionId">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{25}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="solutionSetId">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{23}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="version">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]+"/>
```

```
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="search">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="summarizer" maxOccurs="unbounded"
minOccurs="0">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="pricePerDate"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="inputs" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="destination" maxOccurs="unbounded"
minOccurs="0" type="cityOrAirportCode"/>
                <xs:element name="passengers">
                  <xs:complexType>
                    <xs:attribute name="adults" type="xs:int"/>
                    <xs:attribute name="children" type="xs:int"/>
                    <xs:attribute name="infants" type="xs:int"/>
                    <xs:attribute name="seniors" type="xs:int"/>
                    <xs:attribute name="youths" type="xs:int"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
              <xs:attribute name="departure" use="required"
type="localDate"/>
              <xs:attribute name="maxStay" use="required"
type="xs:int"/>
              <xs:attribute name="minStay" use="required"
type="xs:int"/>
              <xs:attribute name="officeID" use="required"
type="xs:string"/>
              <xs:attribute name="origin" use="required"
type="cityOrAirportCode"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="format" type="outputFormat"/>
        <xs:attribute name="key" use="required" type="key"/>
        <xs:attribute name="name" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="multiDate"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="requestId" type="xs:string"/>
        <xs:attribute name="session" type="sessionId"/>
        <xs:attribute name="solutionSet" type="solutionSetId"/>
        <xs:attribute name="version" use="required" type="version"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="result">
      <xs:complexType>
        <xs:sequence>
```

```
        <xs:element name="error" minOccurs="0" type="error"/>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="pricePerDate">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="departure" maxOccurs="unbounded"
minOccurs="0">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="options" minOccurs="0">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="return"
maxOccurs="unbounded" minOccurs="0">
                              <xs:complexType>
                                <xs:sequence>
                                  <xs:element name="solution"
minOccurs="0">
                                    <xs:complexType>
                                      <xs:sequence>
                                        <xs:element
name="displayTotal">
                                          <xs:complexType>
                                            <xs:attribute
name="amount" use="required" type="xs:decimal"/>
                                            <xs:attribute
name="currency" use="required" type="xs:string"/>
                                          </xs:complexType>
                                        </xs:element>
                                        <xs:element name="ext"
minOccurs="0">
                                          <xs:complexType>
                                            <xs:attribute
name="promo" type="xs:boolean"/>
                                          </xs:complexType>
                                        </xs:element>
                                      </xs:sequence>
                                      <xs:attribute name="id"
use="required" type="xs:string"/>
                                    </xs:complexType>
                                  </xs:element>
                                </xs:sequence>
                                <xs:attribute name="lengthOfStay"
use="required" type="xs:int"/>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                    <xs:attribute name="date" use="required"
type="localDate"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:sequence>
```

```
      <xs:attribute name="id" use="required" type="resultId"/>
      <xs:attribute name="requestId" type="xs:string"/>
      <xs:attribute name="session" use="required"
type="sessionId"/>
      <xs:attribute name="solutionCount" use="required"
type="xs:int"/>
      <xs:attribute name="solutionSet" type="solutionSetId"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.2 Get Itinerary

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="airportCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="bookingCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{1,2}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="dateTime">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}([+-
]\d{2}:\d{2})?"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="error">
    <xs:sequence>
      <xs:element name="message" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="code" type="xs:string"/>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="internal"/>
          <xs:enumeration value="input"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
  <xs:simpleType name="flightCode">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9]{2}\d+[a-z]?"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="key">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{22}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="outputFormat">
    <xs:restriction base="xs:string">
      <xs:enumeration value="json"/>
```

```
        <xs:enumeration value="xml"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="paxTypeCode">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Z _]{1,6}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="resultId">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z0-9]{22}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="sessionId">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z0-9]{25}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="solutionId">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z0-9]{23}/[A-Za-z0-9]{25}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="solutionSetId">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z0-9]{23}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="version">
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Za-z0-9]+"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="summarize">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="summarizer" maxOccurs="unbounded">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="details"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:element name="inputs" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="solution" type="solutionId"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="format" type="outputFormat"/>
        <xs:attribute name="key" use="required" type="key"/>
        <xs:attribute name="requestId" type="xs:string"/>
        <xs:attribute name="session" use="required"
type="sessionId"/>
        <xs:attribute name="solutionSet" use="required"
type="solutionSetId"/>
        <xs:attribute name="version" use="required" type="version"/>
      </xs:complexType>
    </xs:element>
```

```
  <xs:element name="result">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="error" minOccurs="0" type="error"/>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="details">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="displayTotal">
                  <xs:complexType>
                    <xs:attribute name="amount" use="required"
type="xs:decimal"/>
                    <xs:attribute name="currency" use="required"
type="xs:string"/>
                  </xs:complexType>
                </xs:element>
                <xs:element name="ext">
                  <xs:complexType>
                    <xs:attribute name="ebt" use="required"
type="xs:string"/>
                    <xs:attribute name="promo" type="xs:boolean"/>
                  </xs:complexType>
                </xs:element>
                <xs:element name="itinerary">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="slice"
maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="segment"
maxOccurs="unbounded">
                              <xs:complexType>
                                <xs:attribute name="departure"
type="dateTime"/>
                                <xs:attribute name="destination"
use="required" type="airportCode"/>
                                <xs:attribute name="flight"
use="required" type="flightCode"/>
                                <xs:attribute name="origin"
use="required" type="airportCode"/>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
                <xs:element name="pricing" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="fare"
maxOccurs="unbounded">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="bookingInfo"
maxOccurs="unbounded">
                              <xs:complexType>
```

```
                                    <xs:attribute name="bookingCode"
use="required" type="bookingCode"/>
                                    <xs:attribute name="fareCode"
use="required" type="xs:string"/>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element name="passenger"
maxOccurs="unbounded" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="ptc"
maxOccurs="unbounded" minOccurs="0" type="paxTypeCode"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:choice>
          </xs:sequence>
          <xs:attribute name="id" use="required" type="resultId"/>
          <xs:attribute name="requestId" type="xs:string"/>
          <xs:attribute name="session" use="required"
type="sessionId"/>
          <xs:attribute name="solutionCount" use="required"
type="xs:int"/>
          <xs:attribute name="solutionSet" use="required"
type="solutionSetId"/>
        </xs:complexType>
      </xs:element>
</xs:schema>
```

9 QPX ERROR MESSAGES

The errors returned from the level 2 live QPX queries may originate from QPX. The following table enumerates the types of errors that may be returned from QPX.

Code	Name	Severity	Meaning
200	UNCONNECTED-ORIGIN-AIRPORT	6 - User Warning	The search-start airports are unconnected (i.e. there are no flights between any origin airport and airports through which destination airports can be reached).
201	UNCONNECTED-DESTINATION-AIRPORT	6 - User Warning	The search-end airports are unconnected (i.e. there are no flights between any destination airport and airports that can be reached from an origin airport).

202	EMPTY-TIME-WINDOW	4 - Query Data	The latest specified departure time is after the earliest specified arrival time.
203	NO-VALID-ORIGIN-AIRPORT	4 - Query Data	No origin airports have flights departing in the specified time period.
204	NO-VALID-DESTINATION-AIRPORT	4 - Query Data	No destination airports have flights arriving in the specified time period.
205	UNKNOWN-AIRPORT/CITY-CODE	4 Query Data	The identifier supplied is not in the QPX database of locations/airports/cities.
206	BAD-AIRPORT-SPECIFICATION	4 - Query Data	An unrecognized origin or destination airport identifier was provided.
207	NEXT-DEPARTING-FLIGHT	6 - User Warning	The next departure time of any flight from this airport is outside the requested departure time-window.
208	NEXT-ARRIVING-FLIGHT	6 - User Warning	The next arrival time of any flight to this airport is outside the requested arrival time-window.
209	COMMON-ORIGIN-AND-DESTINATION-AIRPORT	4 - Query Data	The scheduler was fed a common airport for origin and destination.
210	SEARCH-PRODUCED-NO-ITINERARIES	6 - User Warning	Scheduler search produced no answers, i.e. no flight options were found for this trip.
211	INCONSISTENT-ITINERARY-TIMES	6 - User Warning	See 308 - ERROR-INCONSISTENT-ITINERARY-TIMES
212	NO-ORIGIN-AIRPORT	4 - Query Data	No airport specified/matched for origin.
213	NO-DESTINATION-AIRPORT	4 - Query Data	No airport specified/matched for destination.
223	TOO-MANY-SLICES	4 - Query Data	The query specifies more than the maximum number of slices.
224	SLICE-NOT-COVERED-BY-SPECIFIED-CARRIERS	6 - User Warning	The carriers specified in the query do not cover this slice. This is a complex scheduler error message.
225	CROSS-SLICE-TR-MCT	6 - User Warning	No itineraries satisfy basic Traffic Restriction (TR) or Minimum Connection Time (MCT) constraints.
226	MIN-MAX-LAYOVER-TIME	6 - User Warning	Could not combine itineraries for two consecutive slices because of minimum/maximum layover restrictions at

			the connecting point.
227	ITINERARIES-SOLD-OUT	6 - User Warning	No seats are available for this itinerary. This is a complex pricing error message.
300†	ERROR-MCT-VIOLATION	4 - Query Data	A connection time is shorter than the specified minimum connection time.
301†	ERROR-ILLEGAL-NUMBER-OF-SLICES	3 - Query Format	The number of slices specified is either less than one or more than the maximum number of slices permitted.
302†	ERROR-NO-ITINERARIES-FOR-SLICE	3 - Query Format	Within a pricing query, no itineraries were specified for a slice.
303†	MULTIPLE-ITINERARIES-FOR-SLICE	3 - Query Format	Within a pricing query, multiple itineraries were specified for a slice.
304†	ERROR-ILLEGAL-NUMBER-OF-FLIGHT-SEGMENTS	3 - Query Format	The number of flight segments is either less than one or more than the maximum number of slices permitted (50 at the time of this documentation).
305†	ERROR-MISSING-FLIGHT-SEGMENT	4 - Query Data	A flight segment was missing during pricing query processing.
306†	ERROR-STRUCTURAL-ITINERARY-PROBLEM	4 - Query Data	A structural consistency check on an itinerary failed, e.g. there wasn't an arrival or departure time for some leg, or the number of stopovers did not correspond properly to the number of legs.
307†	ERROR-DUPLICATE-AIRPORTS-IN-ITINERARY	4 - Query Data	An airport appeared more than once in an itinerary.
308†	ERROR-INCONSISTENT-ITINERARY-TIMES	4 - Query Data	Arrival and departure times do not make sense, e.g. departure time for a flight is more than its arrival time, or the arrival time for a flight is more than the departure time of the next flight.
309†	ERROR-SURFACE-GAP-IN-ITINERARY	4 - Query Data	A surface gap exists, i.e. a flight does not depart from the same place it arrives.
310	ERROR-TRAFFIC-RESTRICTION-VIOLATION	4 - Query Data	Itinerary is invalid because it violates traffic restrictions.
312†	ERROR-MCT-XXX-VIOLATION	4 - Query Data	A connection is prohibited, regardless of the connection time.
320	ERROR-FARE-	4 - Query	Illegal input was given for a fare reduction.

	REDUCTION-SPECIFICATION	Data	
330‡	ERROR-BAD-PASSENGER-SPECIFICATION	3 - Query Format	An ill-formed passenger specification was submitted, which can, during internal expansion, yield an empty age range or submit to a recursion loop.
331‡	ERROR-BAD-ASSERT-GROUPS	3 - Query Format	An ill-formed assert group was detected.
332‡	ERROR-BAD-SLICE-SPECIFICATION	3 - Query Format	An ill-formed slice specification was detected.
333‡	ERROR-QUERY-LINE-ERROR	3 - Query Format	An unrecoverable error occurred during query pre-processing.
334‡	ERROR-QUERY-LINE-WARNING	4 - Query Data	A recoverable error occurred during query pre-processing.
335‡	ERROR-QUERY-LINE-UNKNOWN	3 - Query Format	An unknown error occurred during query pre-processing.
336‡	ERROR-BAD-XML	3 - Query Format	The XML submitted did not translate correctly within QPX.
337	ERROR-BAD-QUERY-HEADER	3 - Query Format	A bad magic query header was specified.
338	ERROR-BAD-AVAILABILITY-SPECIFICATION	3 - Query Format	A bad availability specification was found.
339‡	BAD-QUERY-GROUP-QUERY-NUMBER	3 - Query Format	A bad query group query number was specified.
340	TIMEOUT	5 - Timeout	A timeout occurred.
341	PROCESSING-ERROR	2 - Processing	An internal processing error occurred.
342	MISSING-TPM	6 - User Warning	The ticketed point mileage for a segment is missing, so a TPM will be constructed.
343	MULTIPLE-TPMS	4 - Query Data	There are multiple TPMs between the given cities, and the query did not include a global indicator that can be used to select which TPM should be used.
344	MISSING-ROUTING	6 - User Warning	Missing routing for a fare between two cities.
345	NO-MATCHING-FARE-BASIS	4 - Query Data	There was no matching fare basis found.
346	NO-MATCHING-FARE-TYPE	4 - Query Data	There was no matching fare type found.
347	CHECK-HIP	6 - User Warning	An IATA HIP check was supposed to take place against a "comparable fare," but no comparable fare could be found.

350	CHECK-CTM	6 - User Warning	An IATA circle trip minimum check was supposed to take place against a "comparable fare," but no comparable fare could be found.
356	AIRPORT-TAX-MAY-APPLY	6 - User Warning	An airport tax may apply to one or more of the solutions returned.
357	NO-TAX-FARES	6 - User Warning	There are no fares for performing the tax calculation between locations.
358	AIRPORT-RESTRICTED-FARE	6 - User Warning	The fare for the city referenced is an airport-restricted fare.
359	ILLEGAL-DISPLAY-CURRENCY	4 - Query Data	The input currency is not permitted.
360	FAR-OFF-DEPARTURE-DATE	6 - User Warning	Departure date more than 30 days away.
361	DISALLOWED-CAT-15-CURRENCY	6 - User Warning	The sales currency is not allowed for the origin or destination country.
376	SLICE-DEP-TIME-IN-THE-PAST	4 - Query Data	The latest departure time specified for this slice is in the past.
400	WARNING-SWITCH-TO-UNKNOWN-AVAILABILITY	6 - User Warning	In situations where solutions cannot be priced because of availability, unknown availability has been accepted by QPX, in order to generate solutions.
401	ERROR-NO-ITINERARIES	6 - User Warning	None of the itineraries found are available.
403	ERROR-NO-INTERNATIONAL	6 - User Warning	When an international query completely fails to give answers, return this error.
404	WARNING-NO-ANSWER	6 - User Warning	When a query completely fails to give answers, and no other specific explanation for failure is produced, return this error.
405	WARNING-SINGLE-PASSENGER-CLASS	6 - User Warning	Multiple passenger classes submitted in a query have been merged down to a single class.
406	ITINERARIES-ALREADY-DEPARTED	6 - User Warning	The only flights found for this search have already departed. This is a complex scheduler error message.
408	TOO-TIGHT-MAX-LEGS-SEGMENTS-CONSTRAINTS	6 - User Warning	No flights were found within the constraints that the query imposed on the maximum number of legs or segments. This is a complex scheduler error message.
409	NO-	6 - User	The carriers specified in the query cover this

	ITINERARIES-SUGGESTION-OTHER-TIME	Warning	slice, but the only flights are before the earliest time or after the latest time specified by the query. This is a complex scheduler error message.
410	NO-FARES	6 - User Warning	No fares were found between the specified cities. This is a complex pricing error message.
413	PRICING-CONDITIONS-NOT-MET	6 - User Warning	When trying to do a pricing query, no fares were found that passed the necessary rules and checks between the specified cities. This is a complex pricing error message.
450	TOO-TIGHT-MILEAGE-CONSTRAINTS	6 - User Warning	No flights were found within the constraints that the query imposed on the maximum or minimum mileage. This is a complex scheduler error message.
451	TOO-TIGHT-DURATION-CONSTRAINTS	6 - User Warning	No flights were found within the constraints that the query imposed on the maximum duration. This is a complex scheduler error message.
452	TOO-TIGHT-CONNECTION-CONSTRAINTS	6 - User Warning	No flights were found within the constraints that the query imposed on the maximum or minimum connection time. This is a complex scheduler error message.
453	TOO-TIGHT-ROUTE-LOCATIONS-CONSTRAINTS	6 - User Warning	No flights were found within the constraints that the query imposed on the permitted or prohibited route locales. This is a complex scheduler error message.
500	MISC-SUMMARY-INFO	6 - User Warning	Miscellaneous summary information.
501	MISSING-BSR-ICH	6 - User Warning	No conversion available from a particular currency to another.
502	MISSING-ROE	6 - User Warning	No ROE available for the currency in question in a QPX query.
†Bad itinerary error			
‡Other input error: Not for end-user viewing			