

# Keys To The Cloud

## Connected Customer Experience

### Overview

In this guide, you will learn how to leverage TIBCO Connected Intelligence Cloud to create a connected customer experience. You will create dynamic event driven APIs that drive an innovative mobile experience for your customers. Hands-on experience will include:

- What is an API is, how to specify and implement an API on TIBCO Connected Intelligence Cloud
- How to rapidly reuse and implement APIs from existing resources
- How to leverage cloud events to drive a better customer experience and keep your customers engaged
- How to leverage TIBCO Cloud Analytics in your solutions to drive better outcomes

### Setup

1. [Sign up for a trial TIBCO Connected Intelligence Cloud account](#). You will need this for the hands-on workshop. Sign-up for TIBCO Cloud Integration, TIBCO Cloud Spotfire and TIBCO Cloud Events.
2. [Download](#) the artifacts that will be used in the hands-on workshop. Extract the zip file to your hard drive.

### Primer

#### API Program

This section of the workshop will introduce you to API concepts and best practices. We will cover API Specification, Mock API, API Implementation, API Documentation, API Managed Services, naming conventions and result codes.

Shown in figure 1 below, are the keys to a good API program that is agile, fast, promotes collaboration between teams both IT and citizen developers (some might call this shadow IT). Typically, a product owner is responsible for the API specification and other artifacts and shares this with teams including citizen developers, line of business, UI and Platform development

teams and others. This process supports continuous planning. A good API program starts with a blueprint of what needs to get done – the API Specification.

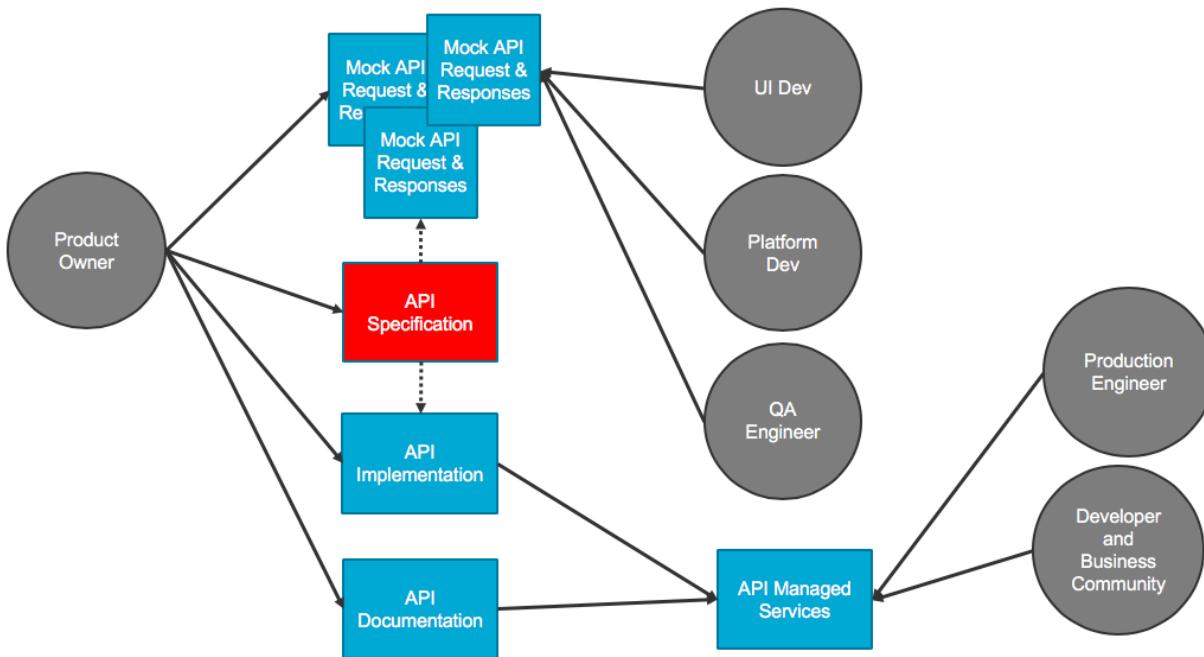


Figure 1

### API Specification

API Specification is the blueprint for the API. Explains how the API will work, every method, parameter and response are planned out. The focus is on defining the API and outlines the structure of the API. This supports continuous planning.

The product owner, shown in figure 2 below, is responsible for the API specification. The product owner will specify the API input, outputs and responses. The product owner will share the specification with other teams and solicit feedback.

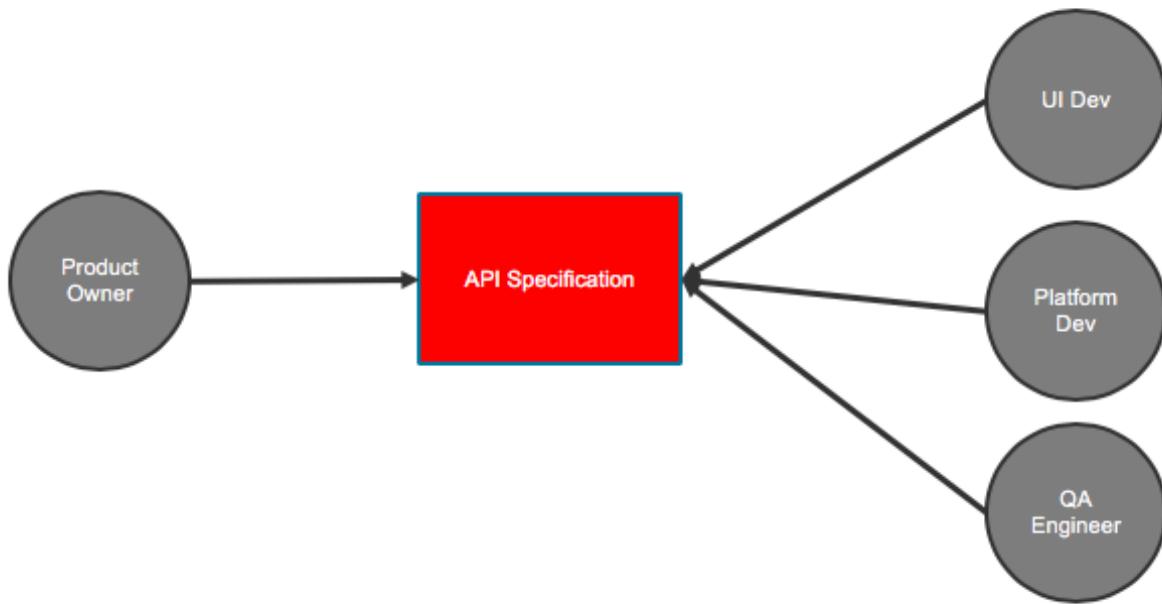


Figure 2

A good API starts with a good name. Examples include:

/passengers – the intent of this API is to return a list of all passengers  
 /passengers/{passengerid} – the intent of this API is to return passenger details for passengerid  
 /flights – the intent of this API is to return a list of all flights  
 /flights/{flightid} – the intent of this API is to return flight details for flightid

The next step is to define the API verbs (also known as methods). The standard practice is to leverage the API verbs in the following manner:

**GET** – is used to retrieve information  
**POST** – is used to create information  
**PUT** – is used to update information  
**DELETE** – is used to delete information

Basic, but powerful, simple and fast.

Then you provide status codes to give a consumer of the API information about the success or failure of the request. The standard practice is to use the following codes:

**1xx: Informational** - Communicates transfer protocol-level information.  
**2xx: Success** -Indicates that the client's request was accepted successfully.

**3xx: Redirection** - Indicates that the client must take some additional action in order to complete their request.

**4xx: Client Error** - This category of error status codes points the finger at clients.

**5xx: Server Error** - The server takes responsibility for these error status codes.

The usual status codes are:

**200** – OK

**201** – OK Created

**204** – OK Deleted

**400** – Bad Request

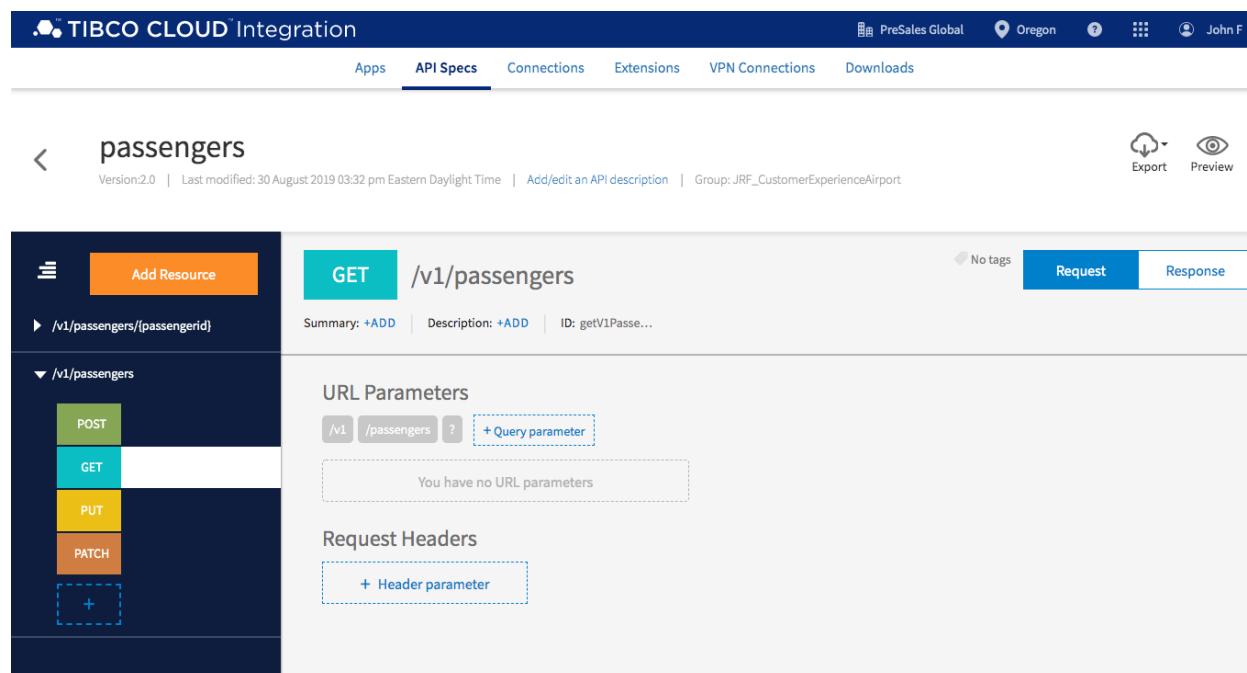
**401** – Unauthorized

**403** – Forbidden

**404** – Not Found

**500** – Internal Server Error

Figure 3 below shows the web browser-based API specification capability of TIBCO Connected Intelligence Cloud. Depicted is the passengers API and methods for POST, GET, PUT, PATCH. This api, /passengers, would return a list of passengers. We will explore API specifications in more detail in a hands-on lab.



The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'PreSales Global', 'Oregon', a user icon for 'John F.', and a 'Cloud' icon. The 'API Specs' tab is selected. The main content area is titled 'passengers' and shows the following details:

- Version: 2.0 | Last modified: 30 August 2019 03:32 pm Eastern Daylight Time | [Add/edit an API description](#) | Group: JRF\_CustomerExperienceAirport
- Export and Preview buttons
- Request and Response tabs
- Summary: +ADD | Description: +ADD | ID: getV1Passe...
- URL Parameters: /v1 /passengers ? + Query parameter
- Request Headers: + Header parameter
- Method list: POST, GET, PUT, PATCH

Figure 3

## Mock API

Before you implement the API it's really a good idea to *fake it before you make it*. Teams can create fake services to indicate what they need an API to do. Architects and developers can get

a head start on their work by leveraging Mock APIs well before the implemented API goes into production; Can also provide feedback and changes earlier in the API development lifecycle. Supports continues planning, development and testing.

Shown in figure 4 below is a typical mock API usage. Here the product owner has generated a Mock API that will enable UI team to get a jump start on their mobile application development, help the platform team to better understand what the intent of the API is and the QA team can start to prepare testing scripts. The most important thing is the product owner can solicit feedback and make changes well before the API is implemented and in production.

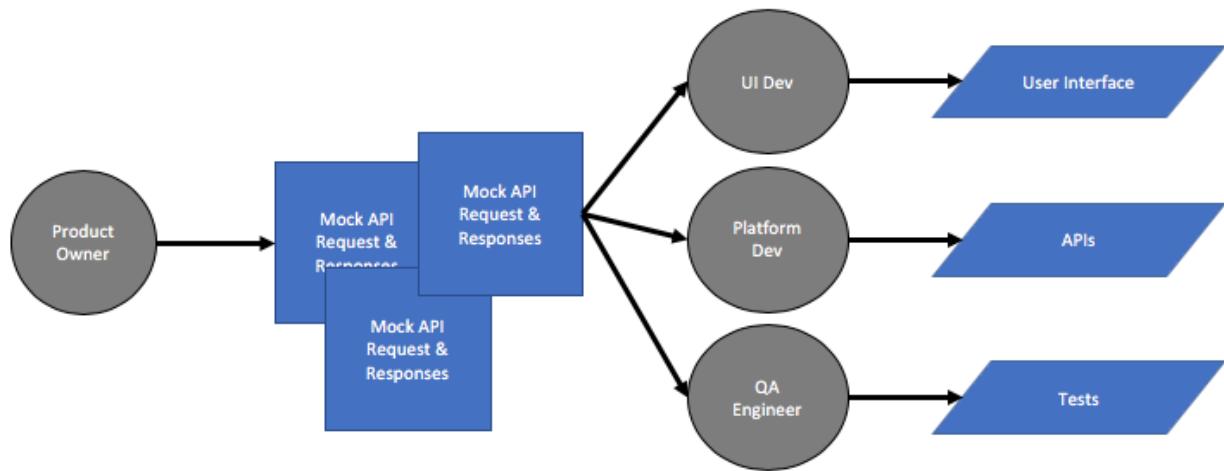


Figure 4

Figure 5 shows the mock application capabilities of TIBCO Connected Intelligence Cloud. Anyone in your organization can preview the API and test input and output results. In the hands-on lab we will show you how to generate a Mock API with a click of your mouse. Its fast and simple.

Figure 5

## API Implementation

Now that the product owner has created API specification, used Mock API to solicited feedback and finetune the requirements for the API, the product owner is ready for an IT savvy citizen developer to reuse system APIs and create a new process API. Some new terms, system API, process API, experience API. Typically, in an organization access to backend systems and opening them up for others to use require specialist to do this, where processes and experience APIs can be implemented by IT savvy citizens in your organization. TIBCO Connected Intelligence Cloud supports these personas. This example and hands-on lab focus on the IT savvy citizen developer.

Let's discuss what an experience API, process API and system API are:

### Experience API

Enables clients, such as mobile applications, chatbots and voice devices to have access to your API network.

### Process API

Orchestrate the System APIs to fulfill a business requirement such as ShipOrder or ReserveFlight.

### System API

Interface with an existing System with a method, protocol, technology, etc that the system understands.

In your case these system, process and experience APIs might represent an order status and fulfillment request, shown in figure 7. Here a front-end application or mobile device is accessing your API network via the experience API OrderStatus. OrderStatus will return the order details and status to the mobile device, it will call a process API, ShipOrder, the will orchestrate calls to the oracle database to review customer information and calls to SAP to review order status for that customer. For the mobile application developer, it's as simple as invoking a request to /OrderStatus/{customerID}; the complexity is taken care of in the system APIs that IT has exposed, tested and made available to everyone in the enterprise.

Figure 6, depicts how the IT savvy citizen developer leverages the API specification and mock API to implement the process API. TIBCO Connected Intelligence Cloud enables the IT savvy citizen to generate a process API skeleton with a single mouse click. Then with no-code, the process API can leverage and reuse existing API to complete the implementation. We will explore this later in a hands-on lab.

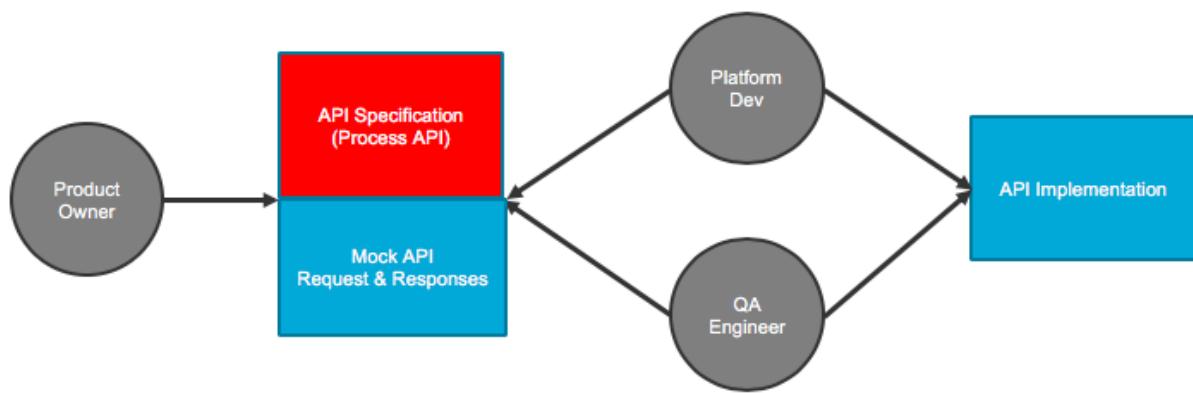


Figure 6

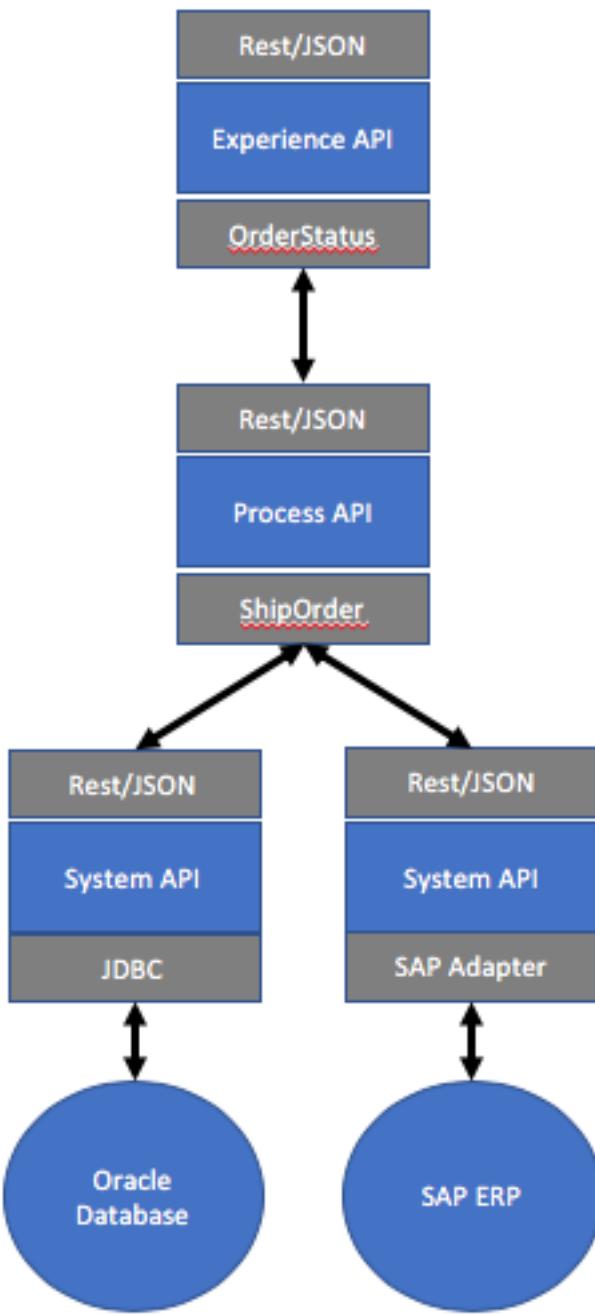


Figure 7

Figure 8 shows a process API that is in the final stages of being implemented on the TIBCO Connected Intelligence Cloud. We will explore API implementation later in a hands-on lab.

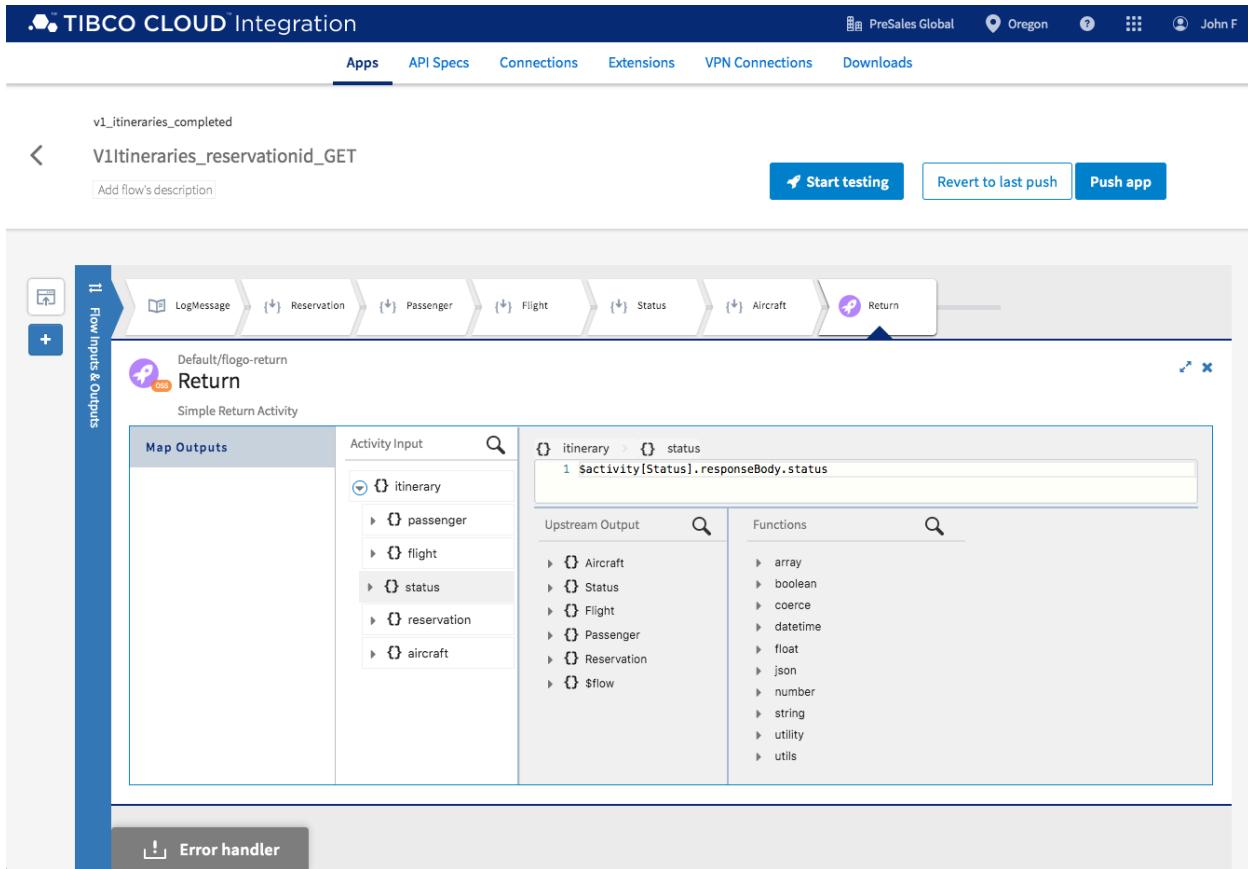


Figure 8

## API Documentation

API Documentation is the reference manual for the API. It tells the API consumers how to use the APIs. It is meant for humans to read. Includes quick start guides, tutorials, interactive documentation so you can try the API. Should include an example for every call, every parameter and responses for every call. An example of API documentation is shown below in figure 9.

Signed in

tibcodev.com

Home Documentation API Console Forum

## Customer API

[Edit](#)

Customer API can be used to request all customers, retrieve a customer based on customerID, retrieve all orders for a customer, or retrieve a specific order based on orderid.

### What can you do with Customer API.

API	Description
<a href="#">GET /customer</a>	Retrieves a list of customers.
<a href="#">GET /customers/{customerid}</a>	Retrieves a specific customer.
<a href="#">GET /customers/{customerid}/orders</a>	Retrieves a list of a specific customer's orders.
<a href="#">GET /orders/{orderid}</a>	Retrieves a specific order.

## Details

## GET /customers

Retrieves a list of customers

## Query Parameters

api\_key Your developer api key.

### Request URL

<https://integration.cloud.tibcoapps.com:443/bbclbm2iiizf7ggao254eb7idu5vuca1/customers>

### Response Body

```
[  
  {  
    "id": 1,  
    "first_name": "Jeanette",  
    "last_name": "Pendreth",  
    "email": "jpPENDRETH@census.gov"  
  },  
  {  
    "id": 2,
```

Figure 9

API Managed Services are the process of publishing APIs, enforcing API usage policy, controlling access and collecting use statistics, performance and reporting. TIBCO Connected Intelligence Cloud provides a full lifecycle API platform and the screenshot below in figure 10 is where managed services starts.

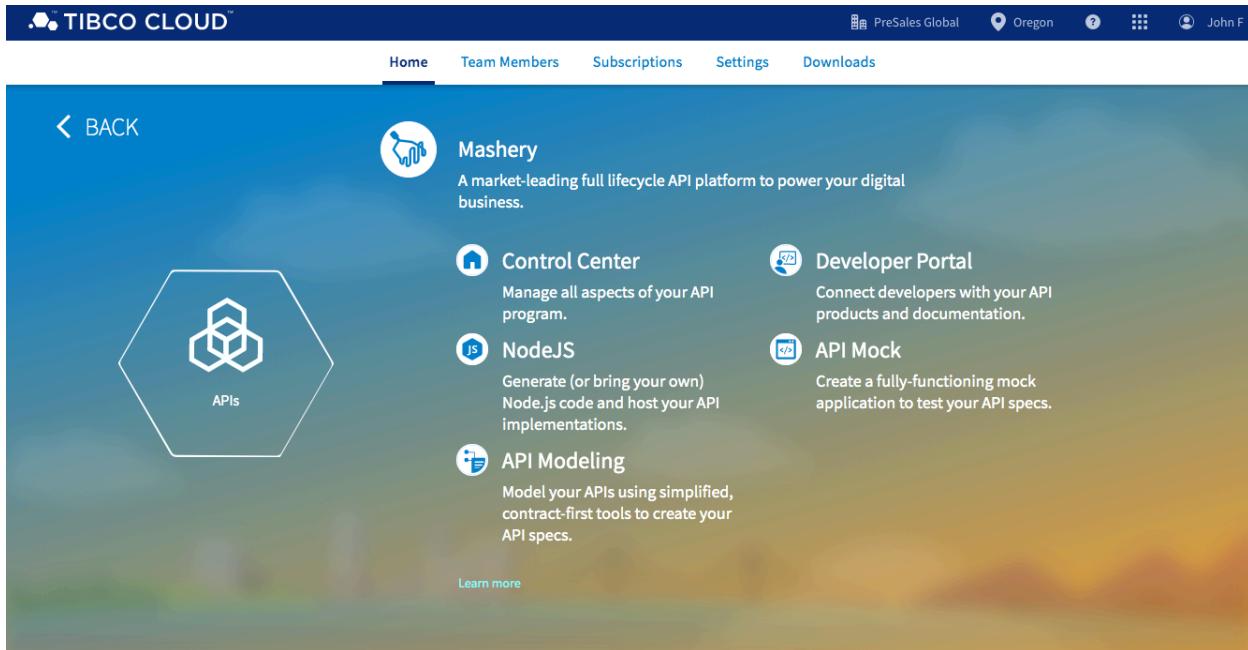


Figure 10

Figure 11 depicts how enterprises typically implement managed services. A production engineer or API specialist will leverage the API implementation and API documentation and expose the API endpoints, such as /passengers/{passenger}, to the development and business community. The production engineer would implement security, rate limits, and other aspects of a managed APIs.

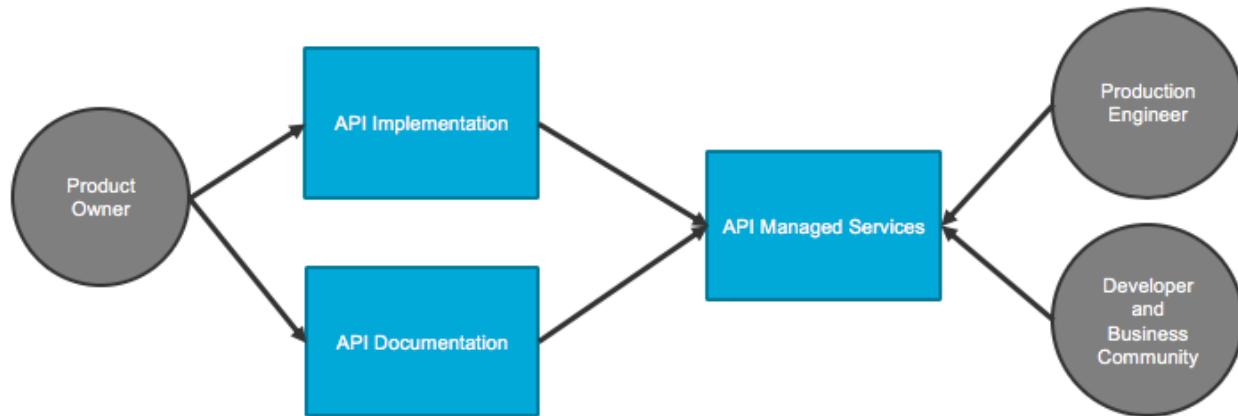


Figure 11

## Summary

In this primer section we explored the key capabilities of an API program that an enterprise would implement to support fast connectivity and innovative app development. Key activities include API Specification, Mock API, API Implementation, API Documentation, and API Managed Services. The next sections will be hands on and cover many of the topics that we just discussed.

## Hands-on

The hands-on labs will explore and extend APIs that were levered in the Connected Customer Experience Airport Mobil Application. This application leverages many APIs used to create a great customer experience. These APIs include:

aircrafts – provides information about the aircraft including aircraftid, type and seat capacity  
flights – manages flight details including departure and arrival cities and times  
itineraries – managers detail for passenger itineraries  
passengers -provides details for all passengers  
reservations – managers the passenger reservation details  
statuses – provides flight status details including terminal and gate  
timecodes – standard time codes

Figure 12 shows the relationship between these APIs and the schemas.

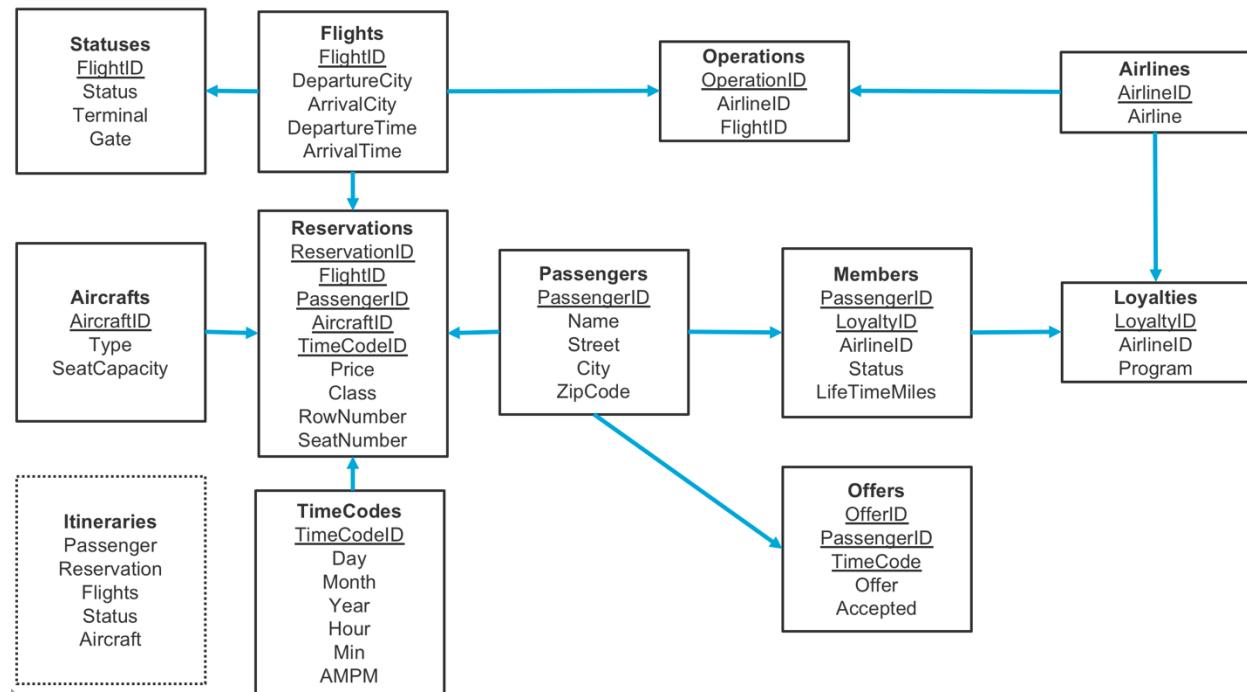


Figure 12

Figure 13 shows how the API network work together to create a Connected Customer Experience. The system APIs, passenger, reservations, flights, and other APIs have been implemented by IT and are ready for IT savvy citizen developers to reuse. The system APIs have hidden the complexity of the under lying systems and make it easy for the mobile application developers and citizen developers to work with and leverage to create better experiences for their customer and create systems of differentiation and innovation. We will be leveraging these system APIs to implement the itineraries API in the upcoming hands-on labs.

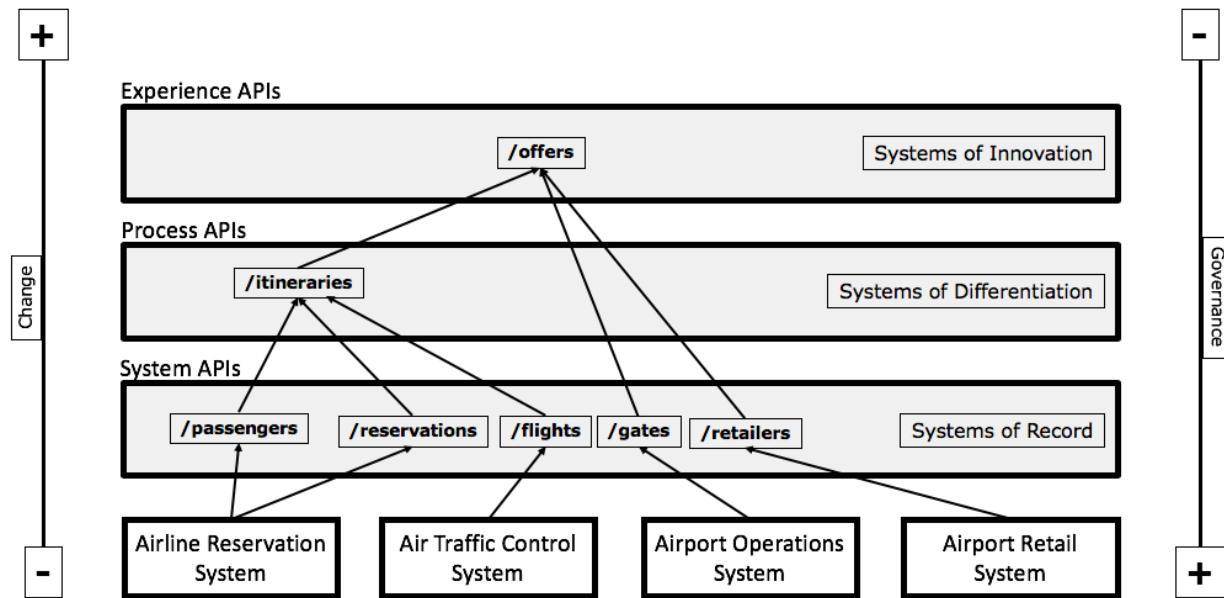


Figure 13

## Hands-on: API Specification

### Overview

In this hands-on lab you will import an existing API specification into TIBCO Connected Intelligence Cloud and explore the API. The goal is to give you a best practice overview of an API that you can leverage in future projects.

### Get Started

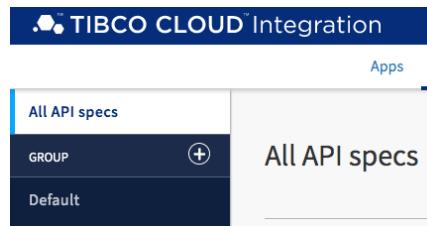
Start by signing into TIBCO Cloud and opening API Specs. There are many ways to navigate to API Spec but let's start with this.

- 1) Start at Welcome to your TIBCO Cloud.
- 2) Select Integration.
- 3) Select Flogo.
- 4) Select API Specs.

## Create Group

Let's create a GROUP to organize your APIs.

- 1) Select the + sign next to GROUP.

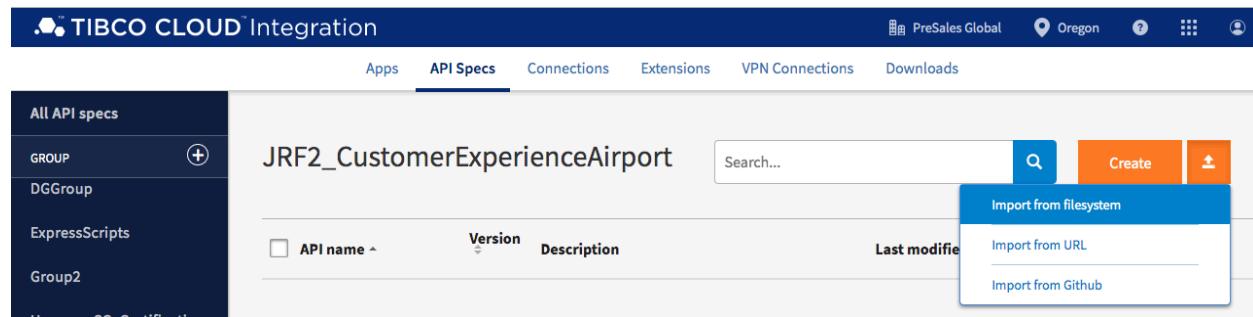


- 2) Name your group CustomerExperienceAirport and press Create group.
- 3) Select your CustomerExperienceAirport group.

## Import API Spec

API Spec allows you to start from scratch or start with an existing specification that you can import from your file system, URL or Github. We are going to start with an existing API spec that will be imported from your file system.

- 1) Select the up-arrow next to create to and choose Import from filesystem.



- 2) Navigate to where you extracted the connected customer experience artifacts and choose CustomerExperienceAirport/APISpecs/v2/passengers.json.

## Explore API Spec

Now let's explore the passenger API spec. The purpose of this exploration is to showcase best practices as discussed in the above sections.

- 1) Select passengers.

**TIBCO CLOUD Integration**

Apps API Specs Connections Extensions VPN Connections Downloads

**passengers**

Version:2.0 | Last modified: 12 September 2019 01:13 pm Eastern Daylight Time | Add/edit an API description | Group: JRF\_CustomerExperienceAirport

**GET /v2/passengers/{passengerid}**

Summary: +ADD | Description: +ADD | ID: getV1Passen...

No tags Request Response

**URL Parameters**

/v2 /passengers /{passengerid} ? + Query parameter

**passengerid** Describe this parameter string Default value

**Request Headers**

+ Header parameter

- 2) Passengers API spec is opened to the `/v2/passengers/{passengerid}` resource and the GET method is highlighted and the request is shown. The `/v2/passengers/{passengerid}` is the path developers will use to access this resource, the API is expecting a passengerid to be passed to it in the request.
- 3) Select Response. 200 and 404 responses have been specified. 200 is the success response and 404 will be returned if no passenger is found for `{passengerid}`.

**TIBCO CLOUD Integration**

Apps API Specs Connections Extensions VPN Connections Downloads

**passengers**

Version:2.0 | Last modified: 12 September 2019 01:13 pm Eastern Daylight Time | Add/edit an API description | Group: JRF\_CustomerExperienceAirport

**GET /v2/passengers/{passengerid}**

Summary: +ADD | Description: +ADD | ID: getV1Passen...

No tags Request Response

**Response Content Type**

application/json  
 application/xml

**Responses**

200/OK Success response

404/Not Found Not Found

+ Add

- 4) Select Response 200/OK. This shows an example of the 200 success response. The passenger API spec is telling us that this API will respond with passenger details that include passengerid, name, street, city, and zip code. Select Cancel.

Edit Response

http status code

200

Description

Success response

Headers +

Your operation has no header objects

Schema  String  Reference  None Generate Schema from Sample Data

passenger  array

```
1 {
2   "type": "object",
3   "properties": {
4     "passenger": {
5       "type": "object",
6       "properties": {
7         "passengerid": {
8           "type": "string",
9           "default": "AAAAA"
10      },
11      "name": {
12        "type": "string",
13        "default": "John Doe"
14      }
15    }
16  }
17}
```

Cancel Save

- 5) Explore some more. Navigate to Resource /v2/passengers. Resources for POST, GET, PUT and PATCH have been specified.

**TIBCO CLOUD Integration**

PreSales Global | Oregon | John F

Apps API Specs Connections Extensions VPN Connections Downloads

**passengers**

Version:2.0 | Last modified: 12 September 2019 01:13 pm Eastern Daylight Time | Add/edit an API description | Group: JRF\_CustomerExperienceAirport

**Export** **Preview**

**POST /v2/passengers**

Summary: +ADD | Description: +ADD | ID: postV1Passe...

No tags

**Request** **Response**

**URL Parameters**

/v2 /passengers ? + Query parameter

You have no URL parameters

**Request Headers**

+ Header parameter

**Request Body Parameter**

Request Content Type Body

application/json Passenger Details

application/xml

Switch to Form parameters

## Preview API

Previewing the API gives you a complete overview of the structure of your API along with input parameters, output responses and samples.

- 1) Select Preview.
- 2) You are presented with interactive documentation showing all resources, parameters, methods and responses. You can share this API specification with anyone in your organization and solicited feedback before you begin to implement. This supports continuous planning and is a best practice development methodology.

The screenshot shows the TIBCO CLOUD Integration API Specs page for the passengers API. The left sidebar lists operations for the /v2/passengers/{passengerid} endpoint, including DELETE, GET, PATCH, POST, and PUT. The main content area shows the default operation for DELETE /v2/passengers/{passengerid}. It includes a table for parameters (passengerid, path, string), response messages (HTTP Status Code 200, Reason Success response, Response Model Headers), and a sample JSON response:

```
{
  "application/json": {
    "passenger": {
      "passengerid": "AAAAA",
      "name": "John Fisher"
    }
  }
}
```

## Summary

API Specification on the TIBCO Connected Intelligence Cloud platform enables you to create blueprint structures for your APIs using a browser-based web GUI. Allows you to collaborate with others in your organizations to get early feedback and allow you to identify needed changes earlier in the API lifecycle.

## Hands-on: Mock API

### Overview

In this hands-on lab you will create a mock API for passengers. it's really a good idea to *fake it before you make it*. Teams can create mock services to indicated what they need an API to do. Architects and developers can get a head start on their work by leveraging the Mock API well before the implemented API goes into production; Can also provide feedback and changes earlier in the API development lifecycle. Supports continues planning, development and testing.

### Get Started

Start by having your API Specs group CustomerExperienceAirport open. There are many ways to open your spec group. Here are the steps starting from at Welcome to your TIBCO Cloud.

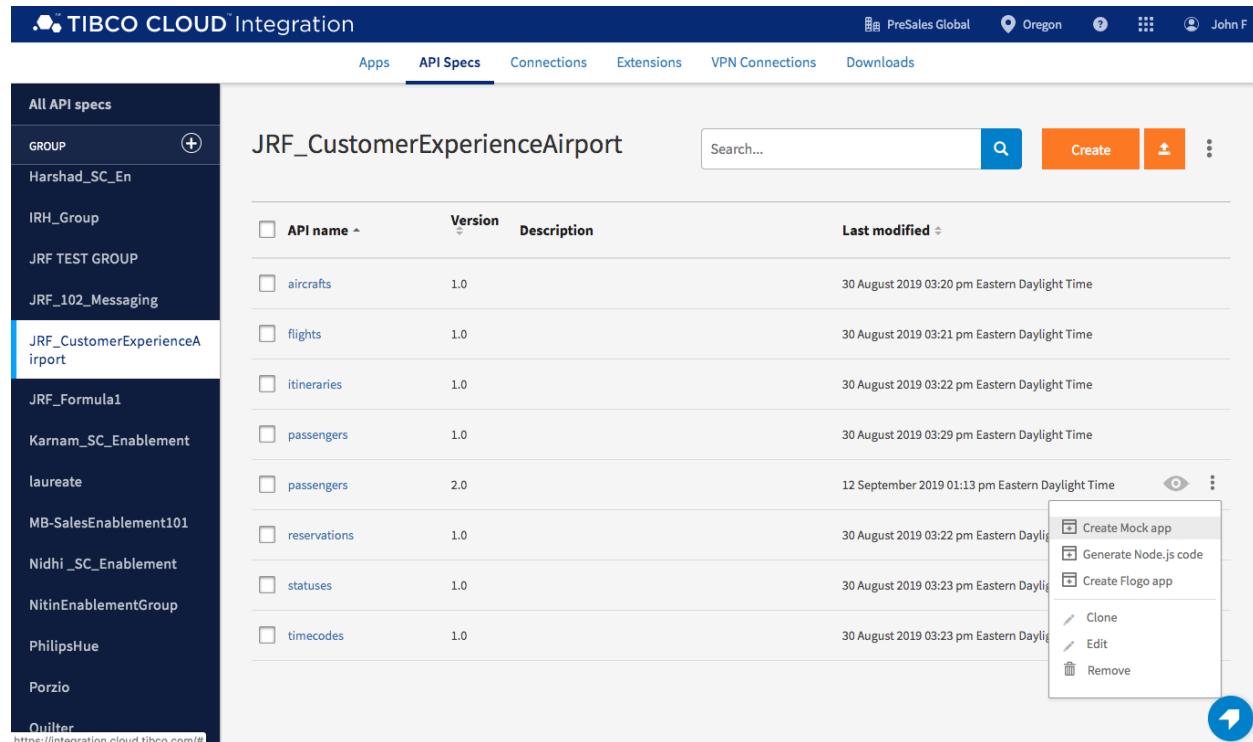
- 1) Start at Welcome to your TIBCO Cloud.
- 2) Select Integration.
- 3) Select Flogo.

4) Select API specs CustomerExperienceAirport.

Create Mock App

We are now going to create a mock application for passengers from your passenger API spec.

- 1) Navigate to passengers, hover over passengers in the API spec list, select ... and choose Create Mock app.



The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'PreSales Global', 'Oregon', a user icon for 'John F', and a 'Create' button. The main menu has tabs for 'Apps', 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. On the left, a sidebar lists 'All API specs' with various groups and their sub-APIs. The 'JRF\_CustomerExperienceAirport' group is selected. The main content area shows a table of API specifications for the 'passengers' API, including 'aircrafts', 'flights', 'itineraries', 'passengers' (version 1.0), 'reservations', 'statuses', and 'timecodes'. The 'passengers' row is highlighted, and a context menu is open over it, showing options: 'Create Mock app' (which is highlighted in grey), 'Generate Node.js code', 'Create Flogo app', 'Clone', 'Edit', and 'Remove'. A blue circular icon with a white arrow is in the bottom right corner of the content area.

API name	Version	Description	Last modified
aircrafts	1.0		30 August 2019 03:20 pm Eastern Daylight Time
flights	1.0		30 August 2019 03:21 pm Eastern Daylight Time
itineraries	1.0		30 August 2019 03:22 pm Eastern Daylight Time
passengers	1.0		30 August 2019 03:29 pm Eastern Daylight Time
passengers	2.0		12 September 2019 01:13 pm Eastern Daylight Time
reservations	1.0		30 August 2019 03:22 pm Eastern Daylight Time
statuses	1.0		30 August 2019 03:23 pm Eastern Daylight Time
timecodes	1.0		30 August 2019 03:23 pm Eastern Daylight Time

- 2) Give your mock app a name, v2\_passenger and select Create.

Create Mock app X

Give your new Mock app a name

Cancel Create

3) Your mock app v2\_passengers will be created and deployed to the cloud.

**TIBCO CLOUD** Integration

PreSales Global | Oregon | John F | More

Apps API Specs Connections Extensions VPN Connections Downloads

Apps All my apps Last modified Create Up

Logo API Mock Node.js BusinessWorks

**v2\_passengers**  
Modified on 12 September 2019 02:47 pm Eastern Daylight Time API Mock App 1 Endpoint More

Running

## Test

This next section will show you how to test your mock application end points.

1) Select the down-arrow next to 1 Endpoint and choose view and test.

**TIBCO CLOUD Integration**

PreSales Global Oregon John F

Apps API Specs Connections Extensions VPN Connections Downloads

Apps All my apps Create

Flogo Api Mock Node.js BusinessWorks

**v2\_passengers** 1 Running

Modified on 12 September 2019 02:47 pm Eastern Daylight Time API Mock App

1 Endpoint

https://integration.cloud.tibco... View and Test

Copy URL Publish to Mashery

2) Select Try it out! Under GET /v2/passengers.

**TIBCO CLOUD Integration**

PreSales Global Oregon John F

Apps API Specs Connections Extensions VPN Connections Downloads

passengers Tester See more

Search resources

**default**

GET /v2/passengers

Response Class (Status 200)  
Success response

Model Example Value

```
{
  "application/json": {
    "passengers": [
      {
        "city": "Philadelphia",
        "name": "John Fisher",
        "passengerid": "AAAAAA",
        "street": "1 Market Street",
        "zipcode": "19350"
      }
    ]
  }
}
```

Response Content Type application/json

Try it out!

PATCH /v2/passengers

3) Your request for passengers will return a list of passengers.

.• TIBCO CLOUD™ Integration

PreSales Global | Oregon | John F

Apps API Specs Connections Extensions VPN Connections Downloads

passengers Tester

See more

Search resources

Collapse all | Expand all

default

**GET** /v2/passengers

**PATCH** /v2/passengers

**POST** /v2/passengers

**PUT** /v2/passengers

**DELETE** /v2/passengers/{pa...}

**GET** /v2/passengers/{pa...

Request URL

https://integration.cloud.tibcoapps.com:443/caegglliqqqvxxu6ju2p7qsenfcnsa6z/v2/passengers

Response Body

```
{
  "passengers": [
    {
      "passengerid": "AAAAAA",
      "name": "John Fisher",
      "street": "1 Market Street",
      "city": "Philadelphia",
      "zipcode": "19350"
    },
    {
      "passengerid": "BBBBBB",
      "name": "Jim Smith",
      "street": "2 Spruce Street",
      "city": "Boston",
      "zipcode": "29350"
    },
    {
      "passengerid": "CCCCCC",
      "name": "Sally Jones",
      "street": "3 Chestnut Street",
      "city": "Philadelphia",
      "zipcode": "19350"
    }
  ]
}
```

Response Code

- 4) Let's try testing /v2/passengers/{passengerid}. Select or scroll down to /v2/passengers/{passengerid}. Enter 11111 in passengerid parameter and select Try it out!

.• TIBCO CLOUD™ Integration

PreSales Global Oregon John F

Apps API Specs Connections Extensions VPN Connections Downloads

passengers Tester

See more

Search resources

GET /v2/passengers/{passengerid}

Response Class (Status 200)  
Success response

Model Example Value

```
{
  "application/json": {
    "passenger": {
      "city": "Philadelphia",
      "name": "John Fisher",
      "passengerid": "AAAAA",
      "street": "1 Market Street",
      "zipcode": "19350"
    }
  }
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
passengerid	11111		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	Not Found	<pre>{   "code": 404,   "message": "passengerid: 1233 Not Found" }</pre>	

Try it out!

5) Your request for /v2/passengers/11111 will return passenger details.

## Summary

Creating Mock Apps is as simple as a mouse click on TIBCO Connected Intelligence Cloud. In this section we leveraged the passengers API and generated and deployed a mock passengers app to the cloud and tested the API end point.

## Hands-on: API Implementation and Reuse

### Overview

In this hands-on lab you will extend itineraries API implementation by reusing aircrafts API. The current itineraries API responds with reservation, passenger, flight and status details. We want to also include aircraft details.

### Get Started

Start by signing into TIBCO Cloud and opening API Specs. There are many ways to navigate to API Spec but let's start with this.

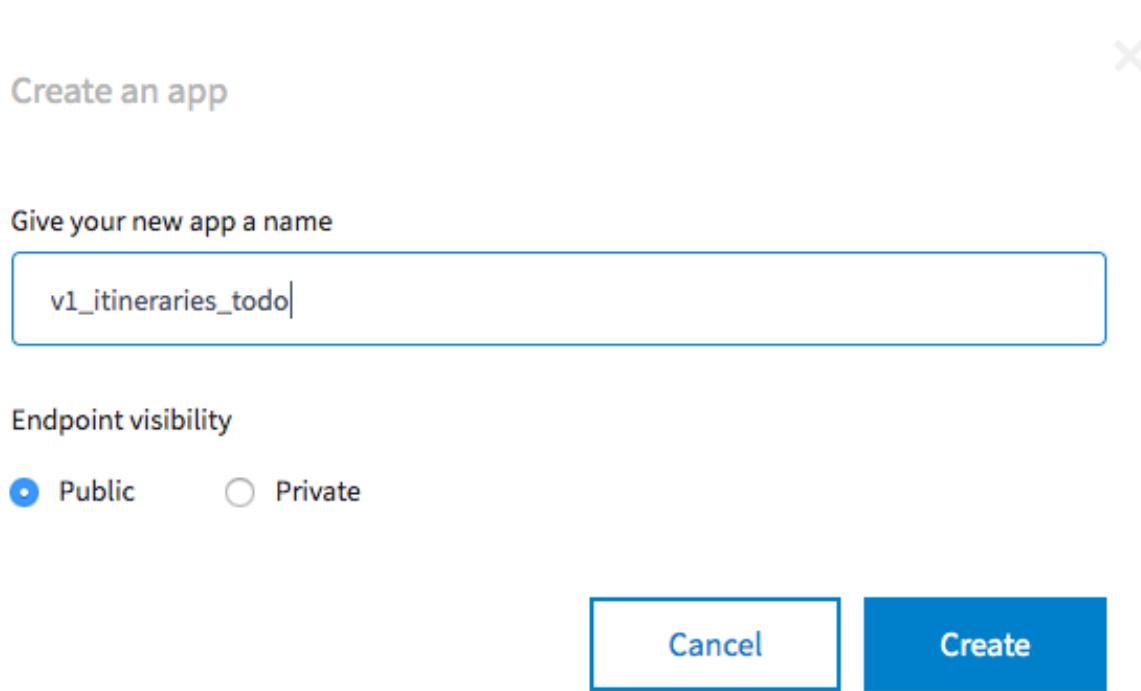
- 1) Start at Welcome to your TIBCO Cloud.
- 2) Select Integration.

- 3) Select Flogo.

#### Import Application

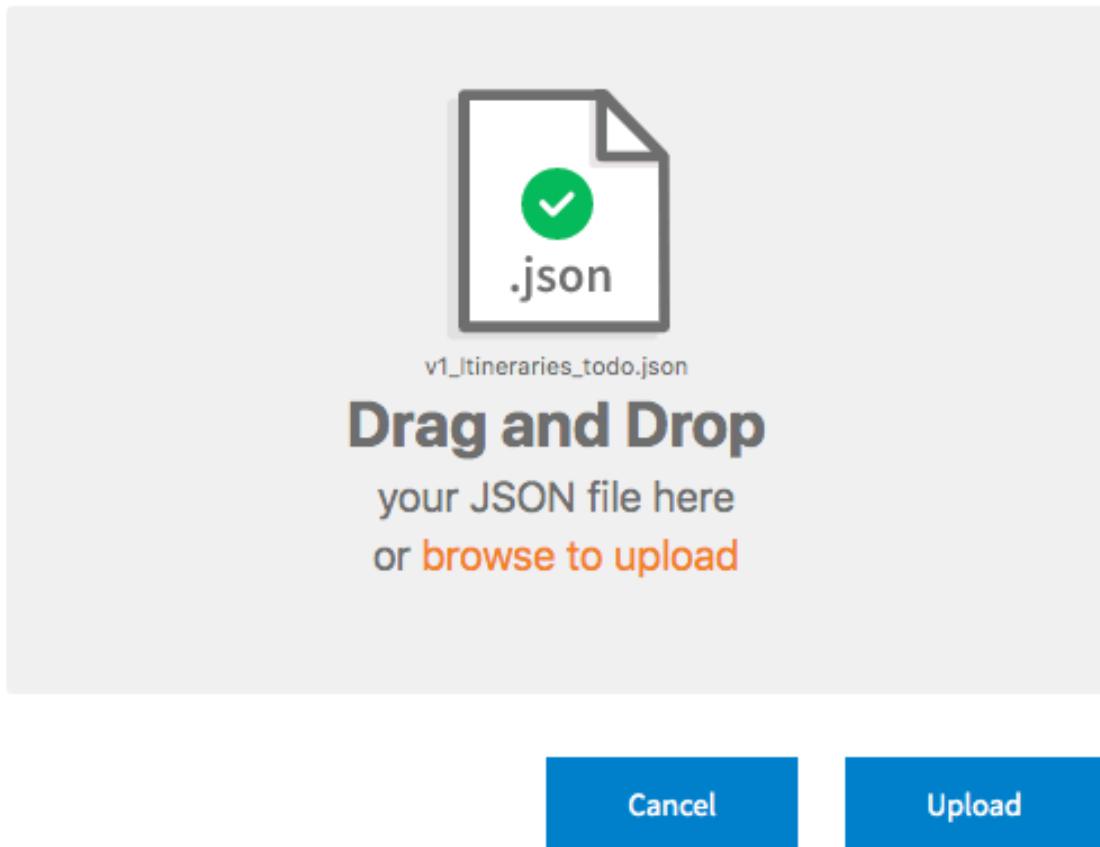
We are going to use a prebuilt application for that hands-on lab and extend it by reusing flights API.

- 1) Select Create.
- 2) Give your new app a name, v1\_itineraries\_todo and select create.

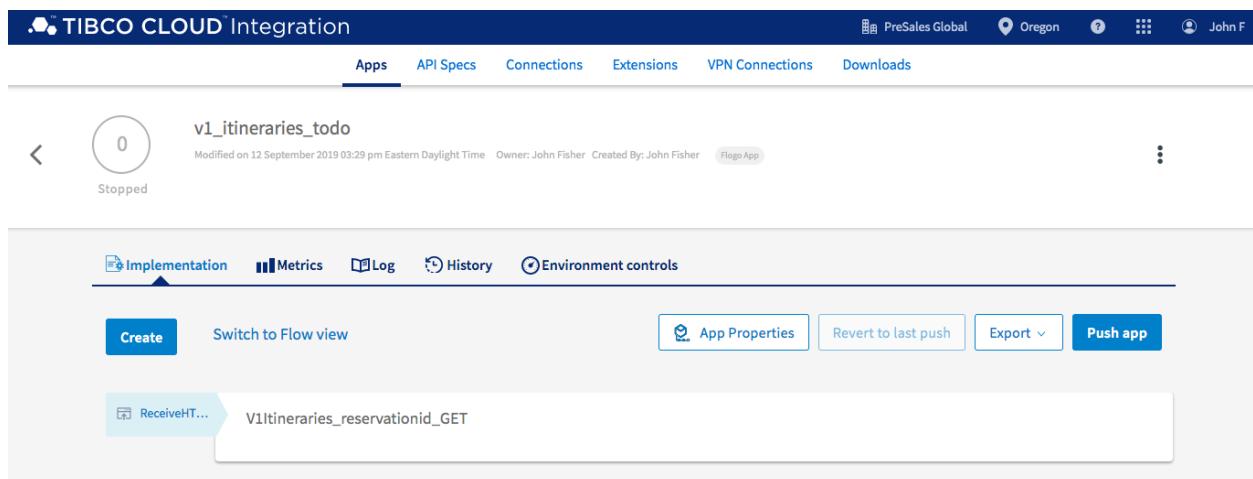


- 3) Select Create a TIBCO Flogo App.
- 4) Select +Import app.
- 5) Navigate to where you extracted the connected customer experience artifacts and choose drag CustomerExperienceAirport/TCI/v1\_itineraries\_todo.json into upload files and select Upload.

## Import app



- 6) Press continue when you get the warning dialog for passwords.
- 7) You now ready to extend this API with aircraft details.

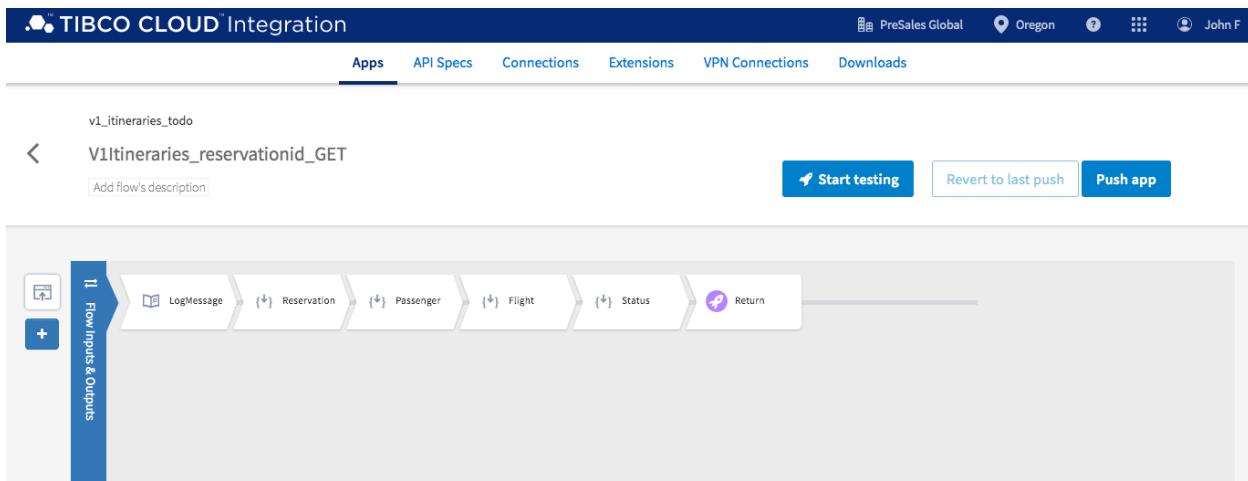


The image shows the 'TIBCO CLOUD Integration' application details page. The top navigation bar includes 'Apps', 'API Specs', 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. The main content area shows an application named 'v1\_itineraries\_todo' with a status of 'Stopped'. It was modified on 12 September 2019 at 03:29 pm Eastern Daylight Time, owned by John Fisher, and created by John Fisher. The 'Implementation' tab is selected, showing a 'ReceiveHTTP' endpoint and a 'V1itineraries\_reservationid\_GET' flow. Other tabs include 'Metrics', 'Log', 'History', and 'Environment controls'. Action buttons at the bottom include 'Create', 'Switch to Flow view', 'App Properties', 'Revert to last push', 'Export', and 'Push app'.

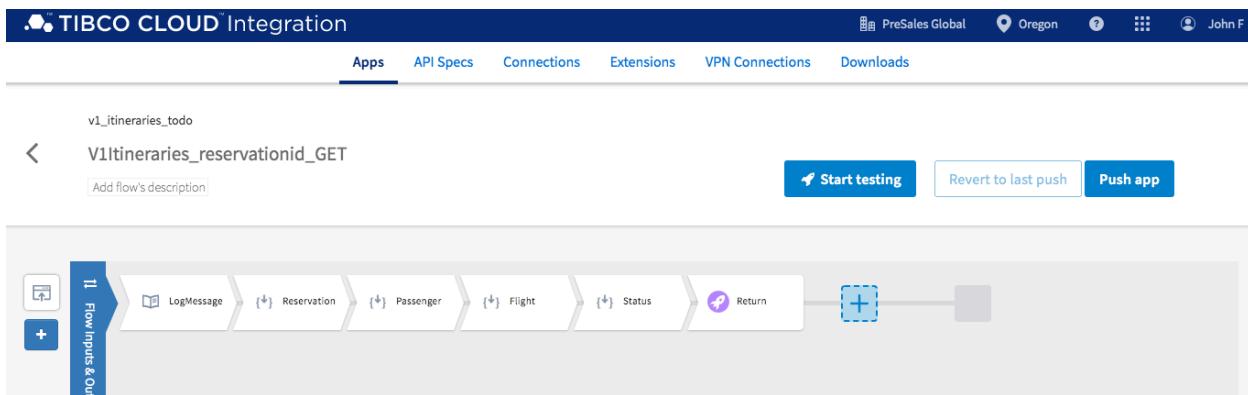
## Extend and Reuse API

Let's extend the itineraries API to include aircrafts details.

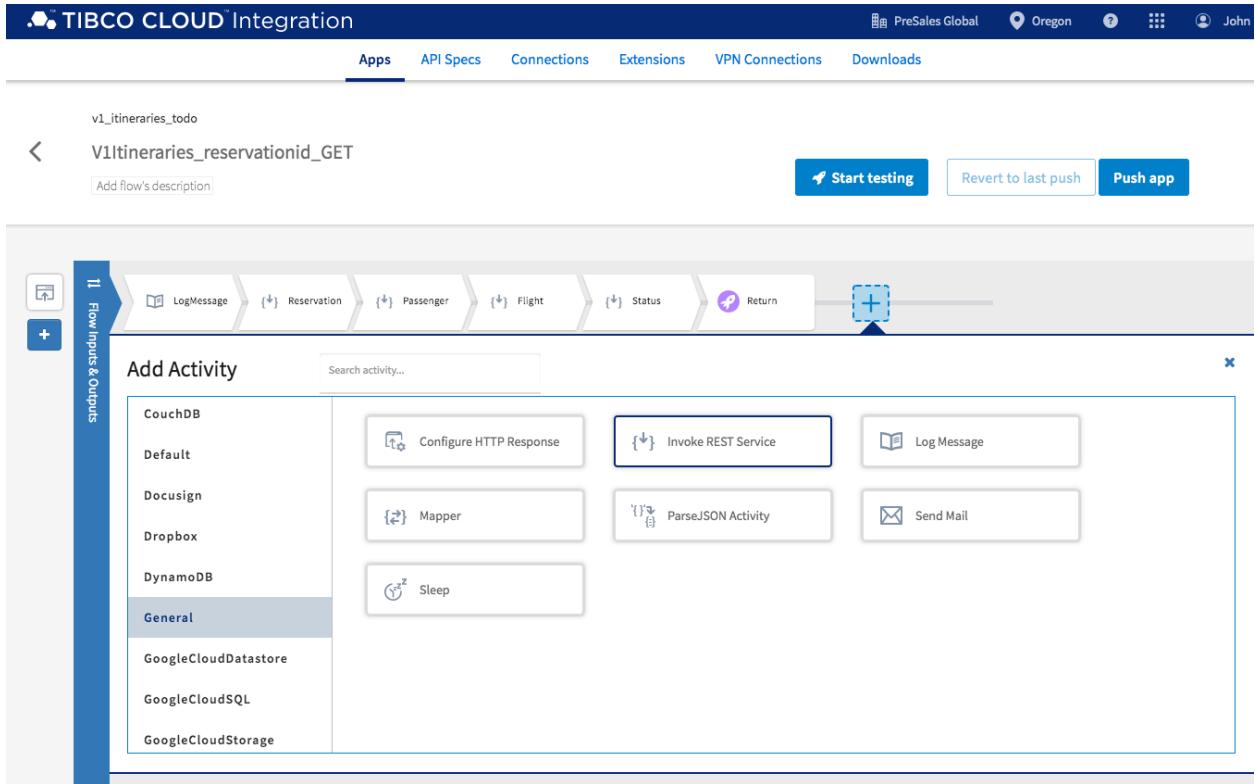
- 1) Select V1Itineraries\_reservationid\_GET. GET /itineraries/{reservationid} implementation will be opened. Here you see the orchestration that invokes reservations, passengers, flights, statuses APIs. You are now going invoke an API call to aircrafts in the next steps.



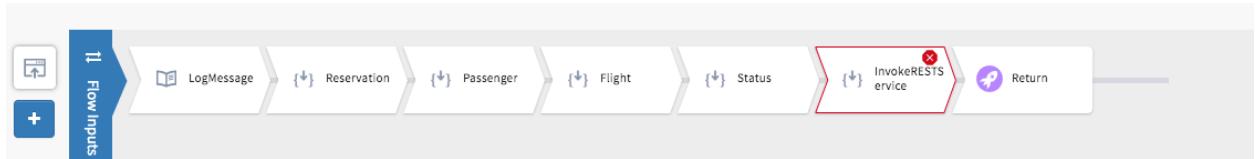
- 2) Hover your mouse over the gray line and select the + to add a new activity.



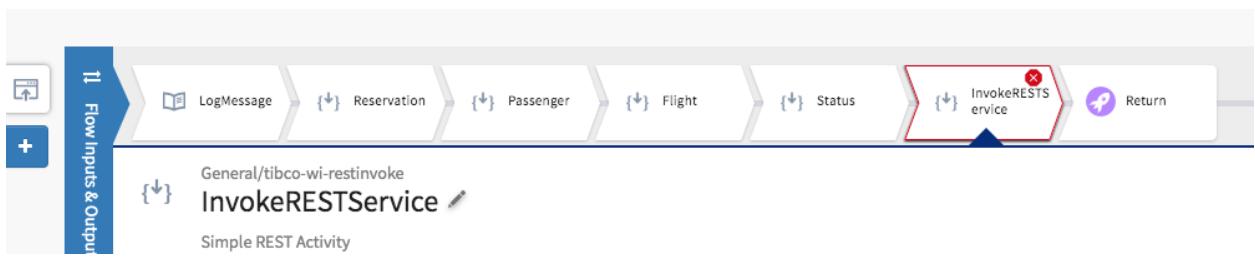
- 3) From Add Activity, scroll down, select General, select Invoke REST Service.



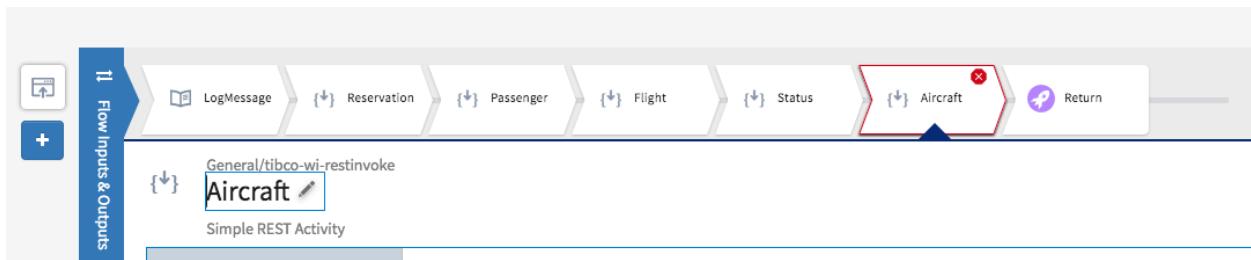
4) Drag the InvokeRESTService to the right of Status.



- 5) Now let's configure the Invoke Rest Service activity.  
 6) Let's rename this activity. Select InvokeRESTService, hover your mouse over the name InvokeRESTService and select the edit icon.



7) Rename InvokeRESTService Aircraft.



8. Select setting and past the following URL in URL.

<https://integration.cloud.tibcoapps.com:443/wz3dr4zotkxmqg5au6yhc526iw2otlc2/v1/aircrafts/{aircraftid}>

9. Now let's map activity input. Select Input, expand pathParams and select aircraftid. Expand Upstream Output->Reservation->responseBody->reservation and select aircraftid.

TIBCO CLOUD Integration

PreSales Global Oregon John P

Apps API Specs Connections Extensions VPN Connections Downloads

v1\_itineraries\_todo

V1ltineraries\_reservationid\_GET

Add flow's description

Start testing Revert to last push Push app

10) Configure the Output settings. Select Output Settings and past the following schema into Response Schema.

```
{"aircraft":  
  {"aircraftid":"111",  
  "type":"767",  
  "seatcapacity":"280"}  
}
```

TIBCO CLOUD Integration

PreSales Global Oregon John F

Apps API Specs Connections Extensions VPN Connections Downloads

v1\_itineraries\_todo

V1ltineraries\_reservationid\_GET

Add flow's description

Start testing Revert to last push Push app

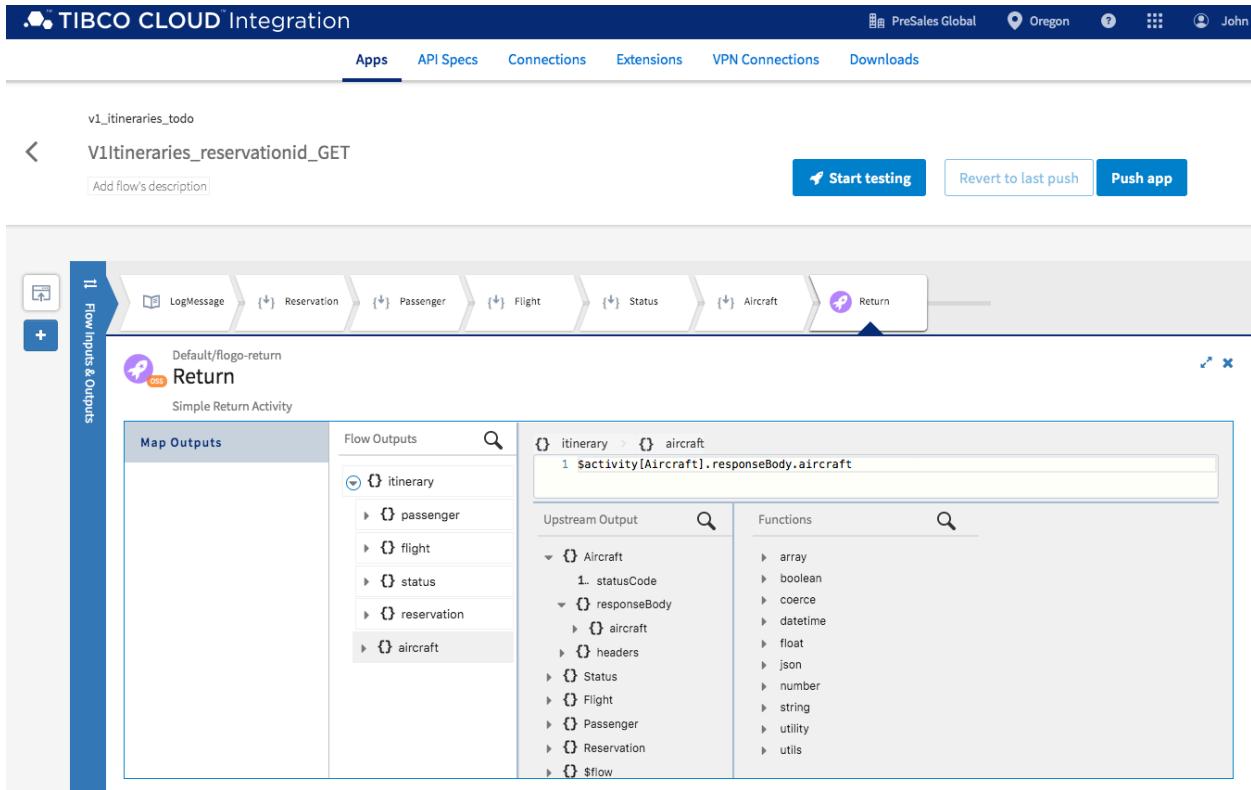
The screenshot shows a flow configuration in the TIBCO CLOUD Integration platform. The flow consists of several steps: LogMessage, Reservation, Passenger, Flight, Status, Aircraft, and Return. The 'Aircraft' step is currently selected, and its configuration panel is open. The panel title is 'General/tibco-wi-restinvoke Aircraft Simple REST Activity'. The 'Output Settings' tab is selected, showing the following configuration:

- Configure Response Codes: True (radio button selected)
- Response Type: application/json
- Response Schema (JSON):

```

1- {"aircraft":
2-   {"aircraftid": "111",
3-     "type": "767",
4-     "seatcapacity": "280"
5-   }
6- }
```
- Response Headers: Add row

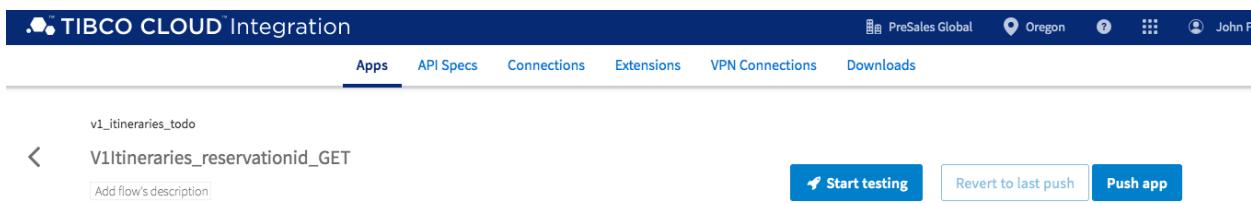
11) Let's map the aircraft details into the Return activity. Select Return activity, expand Flow Outputs itinerary and select aircraft. In upstream Output Aircraft, expand Aircraft->responseBody, select aircraft.



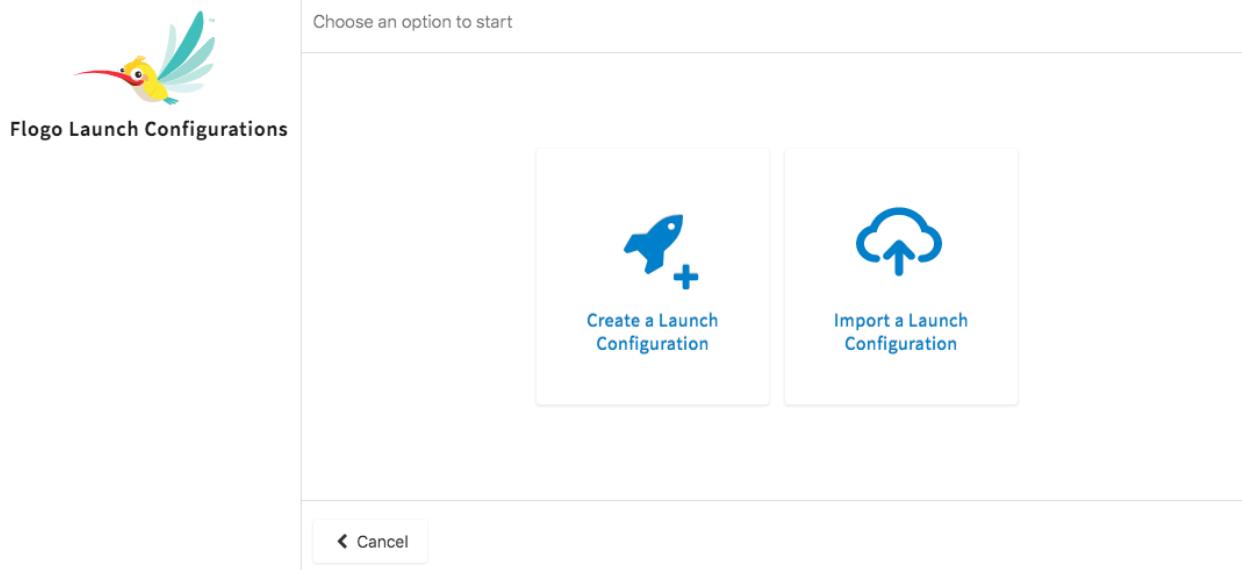
## Start Testing

In this section we will test itineraries API implementation.

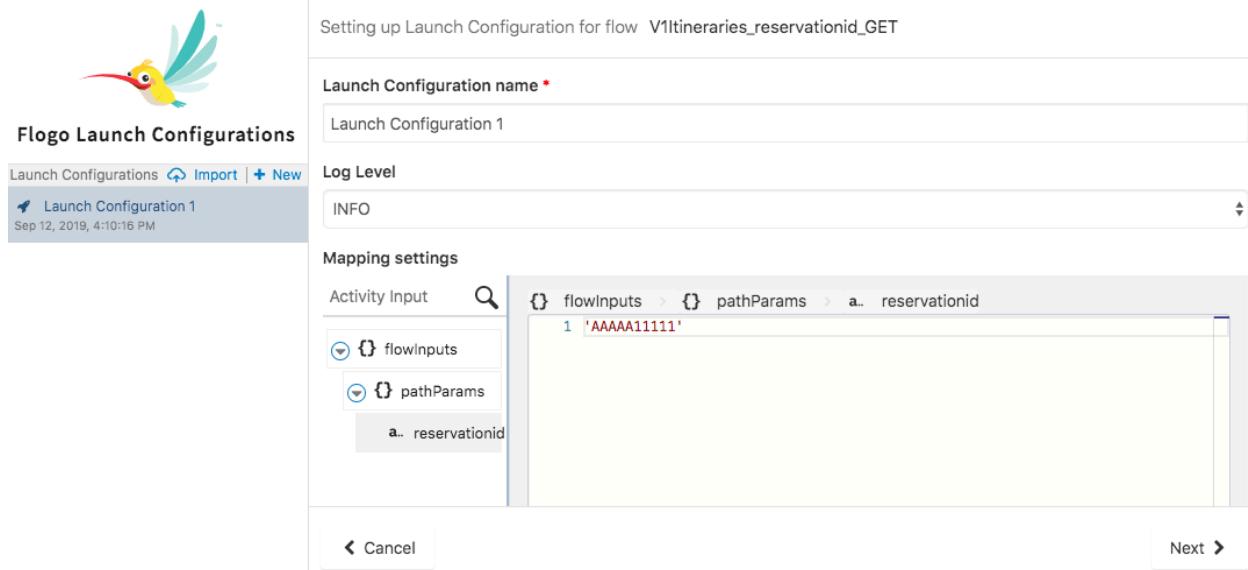
- 1) Select Start testing.



- 2) Select Create a Launch Configuration.



- 3) Expand Activity Input flowInputs->pathParams->reservationid. And enter 'AAAAA11111' into flowInputs. Use quotes. Select Next. Select Run.



- 4) Your Testing is complete. Scroll down to see the response and confirm that aircraft details were added to the response.

## Testing: Launch Configuration 1

[Stop testing](#)

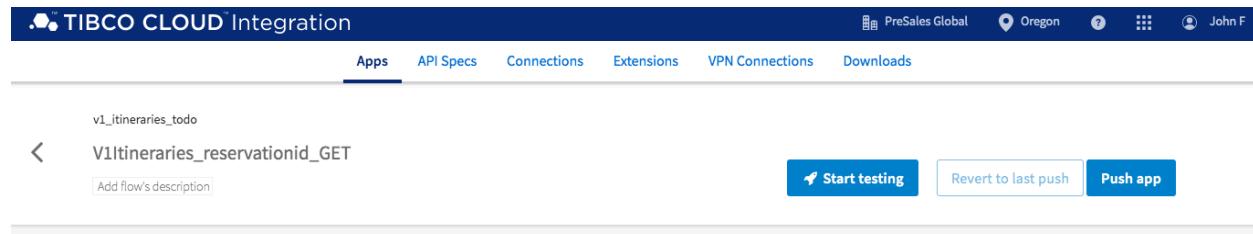
Console output	<pre>Test output 2019-09-12T20:13:03.987Z INFO [general-activity-restinvoke] - REST Call: [GET] https://integration.cloud.tibcoapps.com:443/wz3dr4zotkxmqg5au6yhc526iw2otlc2/v1/aircrafts/111 2019-09-12T20:13:03.996Z INFO [flogo.flow] - Instance [5e3656a31e4af29cb7a884895e3eae6a] Done Flow execution successful {   "itinerary": {     "aircraft": {       "aircraftid": "111",       "seatcapacity": "280",       "type": "767"     },     "flight": {       "arrivalcity": "Boston",       "arrivaltime": "02:30 PM",       "departurecity": "Philadelphia",       "departuretime": "01:00 PM",       "flightid": "11111"     },     "passenger": {       "city": "Philadelphia"     }   } }</pre>
----------------	--

### 5) Select Stop Testing. Cancel.

## Push App

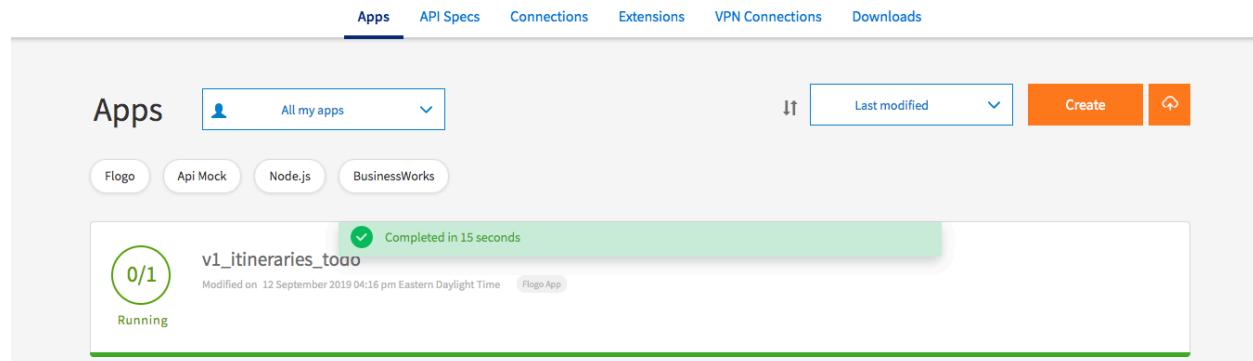
In this section we are going to push the itineraries app to the cloud and test the live endpoints.

### 1) Select Push app.



The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'PreSales Global', 'Oregon', and a user profile for 'John F'. Below the navigation, there are tabs for 'Apps', 'API Specs', 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. The 'Apps' tab is selected. A sub-menu for 'v1\_itineraries\_todo' is open, showing a 'V1itineraries\_reservationid\_GET' endpoint. To the right of this endpoint are three buttons: 'Start testing' (with a play icon), 'Revert to last push', and 'Push app' (highlighted in blue). Below the sub-menu, there is a text input field for 'Add flow's description'.

### 2) The itineraries app will be pushed to the cloud and you will see a progress bar and a completion message.



The screenshot shows the 'Apps' section of the TIBCO CLOUD Integration interface. The top navigation bar and tabs are the same as the previous screenshot. The 'Apps' tab is selected, showing a list of apps: 'Flogo', 'Api Mock', 'Node.js', and 'BusinessWorks'. Below this, a specific app entry for 'v1\_itineraries\_todo' is shown. The entry has a green circular icon with '0/1' and the status 'Running'. To the right of the icon is a green progress bar with a checkmark icon and the text 'Completed in 15 seconds'. Below the progress bar, it says 'Modified on 12 September 2019 04:16 pm Eastern Daylight Time' and 'Flogo App'. There are also 'Last modified' and 'Create' buttons.

### 3) Test your live endpoint by selecting Endpoint, select View and Test.

4) Enter AAAAAA11111 for reservationid input parameter and select Try it out!

## Summary

In this exercise you imported application and extended by reusing aircrafts api. You performed a local test and pushed the application to the cloud. You finished by testing the applications live endpoints.

## Hands-on: Cloud Events

### Overview

In this hands-on lab you will use Cloud Events to create a dynamic API that will respond with contextual offers. Cloud Events will make decisions based on itineraries, gate information, flight status and passenger profile. Cloud Events will respond with the best possible offer for the passenger.

### Get Started

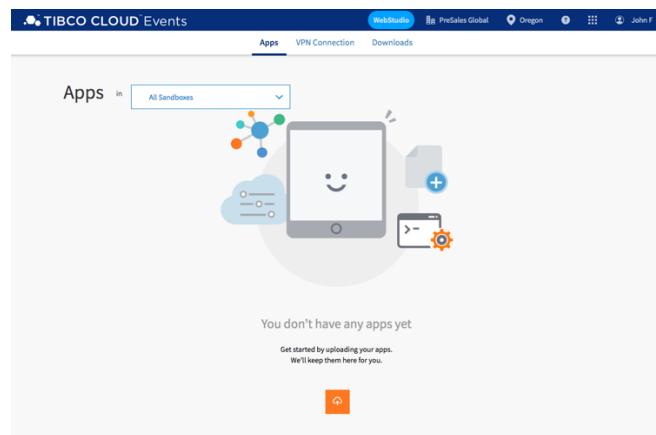
Start by signing into TIBCO Cloud and opening API Specs. There are many ways to navigate to API Spec but let's start with this.

- 1) Start at Welcome to your TIBCO Cloud.
- 2) Select Event Processing.
- 3) Select Events.

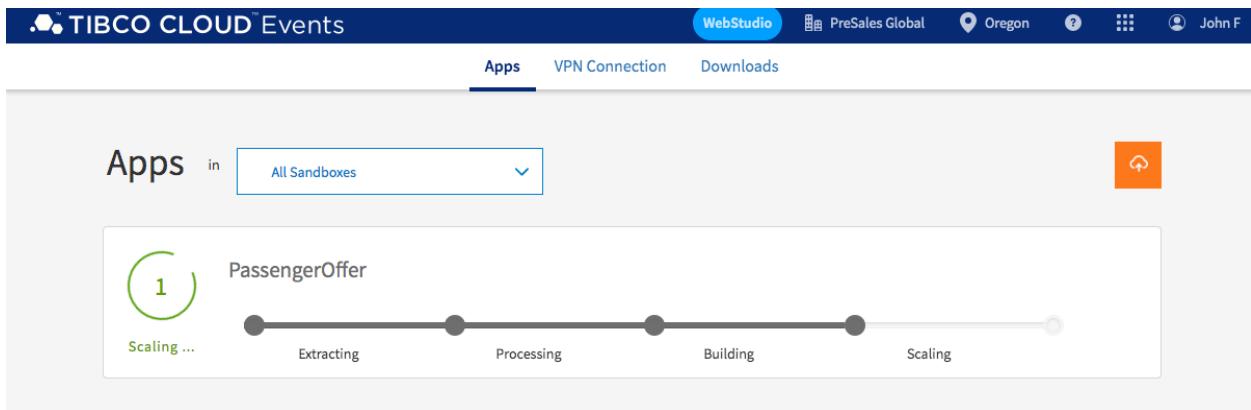
### Uploading Cloud Event Offer Project

For this exercise we are going to start with a prebuilt solution. This solution will make an offer decision based on passenger loyalty and flight status.

- 1) The You don't have any apps yet page should be opened.



- 2) Select Upload, cloud with up-arrow.
- 3) Navigate to where you extracted the connected customer experience artifacts and drag CustomerExperienceAirport/CloudEvents/PassengerOffer.ear into Drag and Drop and select Push.
- 4) The PassengerOffer project files will be pushed to the cloud and started.



Apps in All Sandboxes

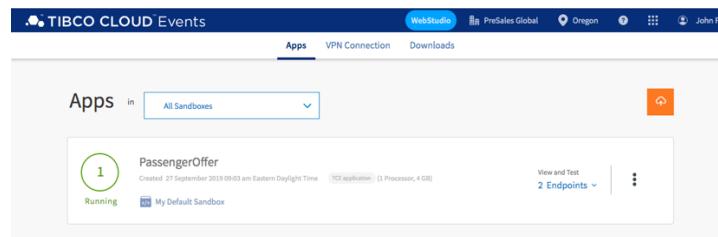
PassengerOffer

Scaling ...

Extracting Processing Building Scaling

## Testing PassengerOffer

- 1) The PassengerOffer project should be running.



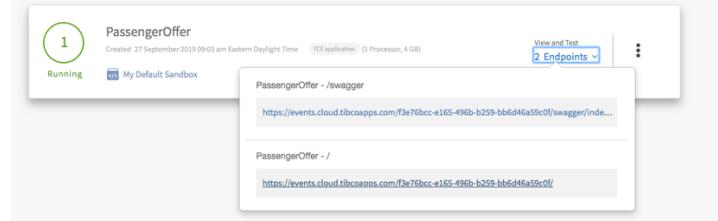
PassengerOffer

Created: 27 September 2019 09:03 am Eastern Daylight Time

Running My Default Sandbox

View and Test 2 Endpoints

- 2) Select 2 Endpoints and choose PassengerOffer Swagger link.



PassengerOffer

Created: 27 September 2019 09:03 am Eastern Daylight Time

Running My Default Sandbox

View and Test 2 Endpoints

PassengerOffer - /swagger

<https://events.cloud.tibcoapps.com/f3e76bcc-e165-496b-b259-bb6d46a59c0f/swagger/index.html>

PassengerOffer - /

<https://events.cloud.tibcoapps.com/f3e76bcc-e165-496b-b259-bb6d46a59c0f/>

- 3) Expand GET Method and select Try it out.

**PassengerOffer** 

[ Base URL: events.cloud.tibocapps.com/fe7ebcc-e165-494b-b259-b0dd46a59c0f ]  
[/PassengerOffer](#)

Schemes

**/Channels/HTTP**

GET /Channels/HTTP/offers

Parameters

Name	Description
passengerid string (query)	passengerid
loyaltystatus string (query)	loyaltystatus
flightstatus string (query)	flightstatus
_extid_ string (query)	_extid_
_ns_ string (query)	_ns_
_nm_ string (query)	_nm_

4) Enter values for parameters as shown below and press Execute.

**PassengerOffer** 

[ Base URL: events.cloud.tibocapps.com/fe7ebcc-e165-494b-b259-b0dd46a59c0f ]  
[/PassengerOffer](#)

Schemes

**/Channels/HTTP**

GET /Channels/HTTP/offers

Parameters

Name	Description
passengerid string (query)	11111
loyaltystatus string (query)	GOLD
flightstatus string (query)	DELAY3
_extid_ string (query)	_extid_
_ns_ string (query)	_ns_
_nm_ string (query)	_nm_

5) The offer, shown below, is Dinner At Chilis.

```

curl -X GET "https://events.cloud.tibcoapps.com/f3e7fbcc-e165-496b-b259-bb646a59c0f/channels/HTTP/offers?passengerid=11111loyaltystatus=GOLDflightstatus=DELAY1" -H "accept: application/xml"

```

Request URL: <https://events.cloud.tibcoapps.com/f3e7fbcc-e165-496b-b259-bb646a59c0f/channels/HTTP/offers?passengerid=11111loyaltystatus=GOLDflightstatus=DELAY1>

Server response

Code	Details
200	<pre> &lt;xml version="1.0" encoding="UTF-8"&gt; &lt;event id="18"&gt; &lt;message&gt;Offer 1111 Dinner At Chilis&lt;/message&gt; &lt;/event&gt; </pre>
	Response headers
	<pre> cache-control: max-age=0, no-cache, no-store, must-revalidate content-type: application/xml;charset=UTF-8 date: Mon, 27 May 2019 14:47:19 GMT expires: -1 pragma: no-cache strict-transport-security: max-age=11316000; includeSubDomains; preload x-content-type-options: nosniff x-frame-options: sameorigin x-xss-protection: 1; mode=block </pre>

Responses

Code	Description
200	200 success
400	400 Bad Request

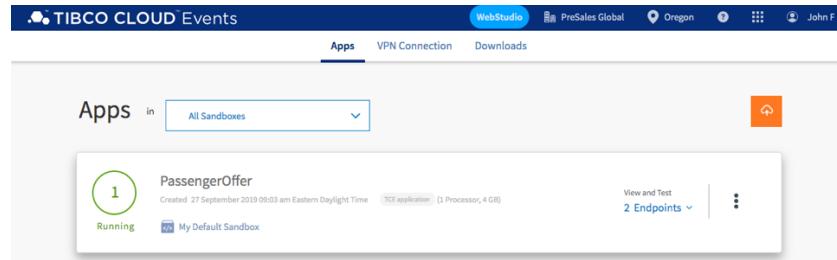
- 6) Try a couple of different combinations of loyalty status and flight status. Use values in the table below.

passengerid	loyaltystatus	flightstatus
11111	BRONZE	DELAY1
11111	SILVER	DELAY1
11111	GOLD	DELAY1
11111	BRONZE	DELAY2
11111	SILVER	DELAY2
11111	GOLD	DELAY2
11111	BRONZE	DELAY3
11111	SILVER	DELAY3
11111	GOLD	DELAY3

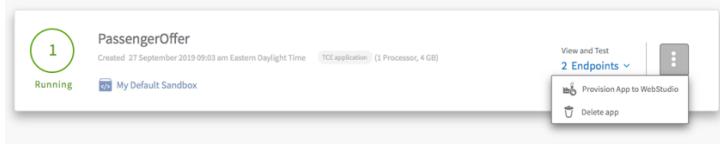
## Working with Cloud Events

In this next section we are going to make a change to the decision table. Currently a passenger with GOLD loyalty and a 2-hour delayed flight is offered a Happy Meal. We are going to change that offer to a Philly Cheesesteak and Coke.

- 1) Navigate to Cloud Events Apps page by Logging into [cloud.tibco.com](https://cloud.tibco.com) and select Event Processing and select Events. You should see your running PassengerOffer App.



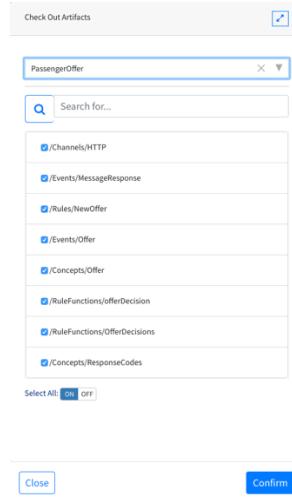
- 2) You are going to use WebStudio to make changes to the Cloud Events decision table. But before you can do this, you need to provision the PassengerOffer App to WebStudio. Select Provision App to WebStudio from the dropdown and wait for dialog indicating the provisioning was successful.



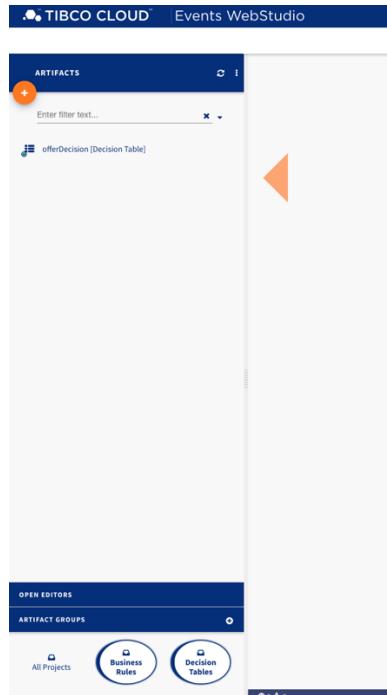
- 3) Select WebStudio. Navigate to Dashboard. The dashboard shows all recent activities. Navigate to Workspace. We are going to check out project artifacts and make changes. Select +.



- 4) Select PassengerOffer and select Confirm. This will check out all PassengerOffer artifacts.



- 5) You are going to make changes to the PassengerOffer decision table. Select offerDecision from ARTIFACTS list.

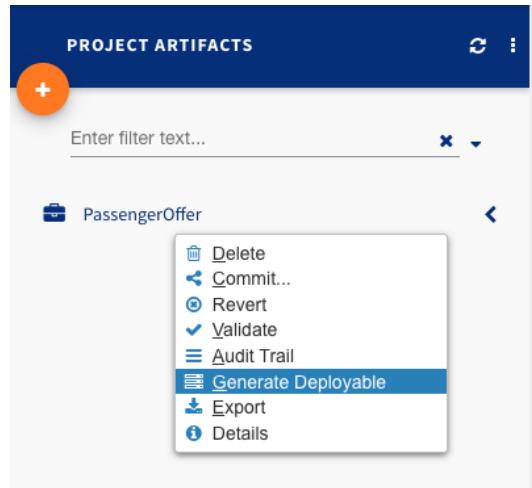


- 6) In the offerDecision table scroll down to row 6 where SILVER, DELAY2 and “Happy Meal” is displayed. Change “Happy Meal” to “Philly Cheesesteak and Coke”. Select Save.

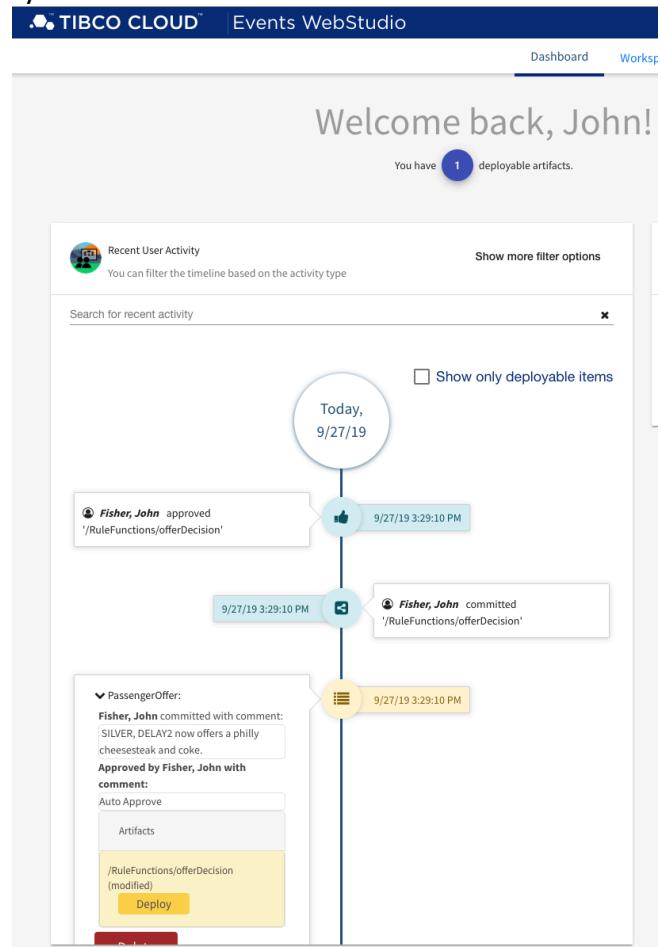
- a. Commit your changes. Select Commit, add a message and select Confirm.

- 7) Select show resources as tree.

- 8) Right-click on PassengerOffer and select Generate Deployable. Select PassengerOffer and select Confirm.

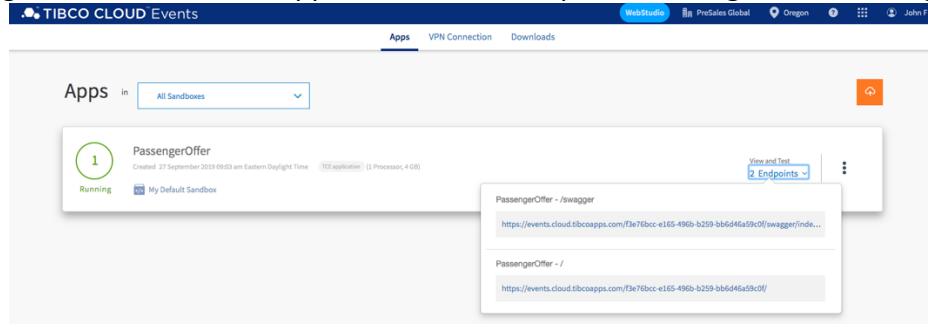


- 9) Let's deploy the changes. Select Dashboard, select >PassengerOffer from recent activity and select Deploy.



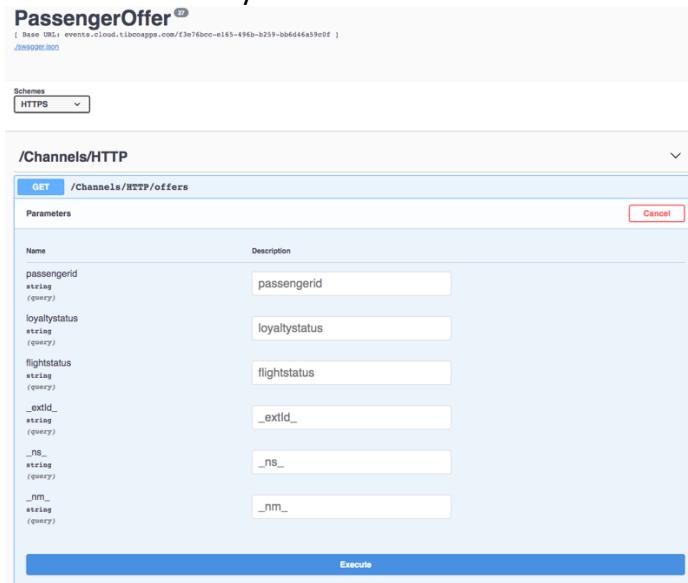
## Testing PassengerOffer Changes

- 1) Navigate to Cloud Events Apps and select 2 Endpoint PassengerOffer - /swagger.



The screenshot shows the TIBCO CLOUD Events interface. In the center, there is a list of apps, with 'PassengerOffer' highlighted. A context menu is open over the app, showing options like 'View and Test' and '2 Endpoints'. The '2 Endpoints' option is specifically highlighted with a blue box.

- 2) Expand GET Method and select Try it out.



The screenshot shows the PassengerOffer /Channels/HTTP swagger interface. It displays the GET /Channels/HTTP/offers method. The method parameters are listed as follows:

Name	Description
passengerid	passengerid
loyaltystatus	loyaltystatus
flightstatus	flightstatus
_extid_	_extid_
_ns_	_ns_
_nm_	_nm_

At the bottom of the interface, there is a blue 'Execute' button.

- 3) Enter values for parameters as shown below and press Execute.

The screenshot shows the PassengerOffer API interface. A GET request is being made to the endpoint /Channels/HTTP/offers. The request parameters are as follows:

- passengerid: 11111
- loyaltystatus: SILVER
- flightstatus: DELAY2
- extid\_

The response content type is set to application/xml.

- 4) The offer, shown below, is Philly Cheesesteak and Coke.

The screenshot shows the API response details. The curl command used is:

```
curl -X GET "https://events.cloud.tibcoapps.com/2a78cc-e165-494b-b259-bb644a59c0f/Channels/HTTP/offers?passengerid=11111&loyaltystatus=SILVER&flightstatus=DELAY2" -H "accept: application/xml"
```

The Request URL is: <https://events.cloud.tibcoapps.com/2a78cc-e165-494b-b259-bb644a59c0f/Channels/HTTP/offers?passengerid=11111&loyaltystatus=SILVER&flightstatus=DELAY2>

The Server response is a 200 OK. The response body is:

```
<xml version="1.0" encoding="UTF-8"?>
<event id="101" type="Offer" Offer="11111 Philly Cheesesteak and Coke">
</event>
```

## Summary

In this exercise you imported a cloud events project, made changes to decision tables and tested PassengerOffer API.

## Hands-on: Cloud Analytics

### Overview

In this hands-on lab you will use Cloud Analytics to explore the results of a customer airport satisfaction survey.

### Get Started

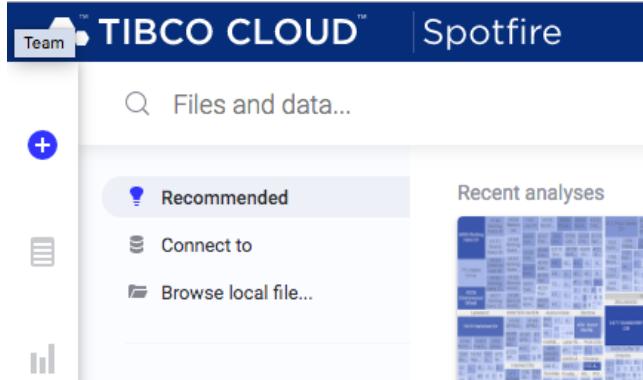
Start by signing into TIBCO Cloud and opening Analytics.

- 1) Start at Welcome to your TIBCO Cloud.
- 2) Select Analytics.
- 3) Select Spotfire.

### Uploading a Spotfire Project

For this exercise we are going to start with a prebuilt solution analytics and upload the solution to your cloud.

- 1) Select Browse Local file...



- 2) Navigate to where you extracted the connected customer experience artifacts and select CustomerExperienceAirport\Analytics\SFO GIS Analysis.dxp and select open.

## Exploring the Analysis

In this exercise you will explore the SFO Airport Survey results.

- 1) Cloud Spotfire has opened to the analysis.



- 2) You are ready to explore.
- 3) Here you can see the results of a customer satisfaction survey.

- 4) Try selecting a category such as Restaurant. You will see the geo map of the airport highlight only Restaurants.
- 5) Click on the background to select all categories. Now under Top Promoters select art work. The map will reflect the selection.
- 6) Click on the tabs to see Terminal View and Survey Data.
- 7) To save this analysis click on Viewing and select Editing.

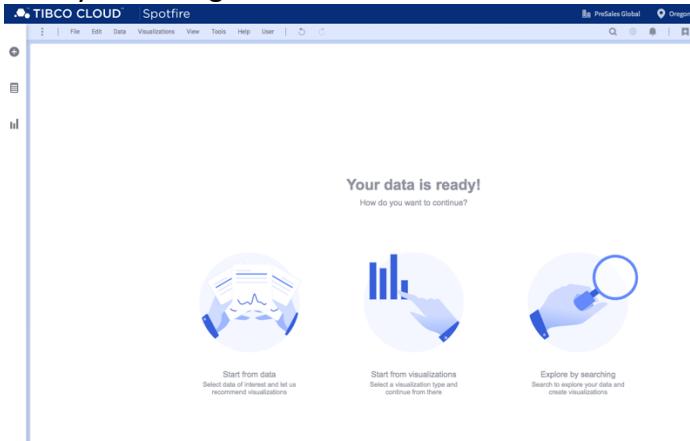


- 8) Select File, select save and select save. This will save this analysis to your library.

### Creating your Own Analysis

In this exercise you will be using natural language processing to create a pie chart to show survey Reponses.

- 1) Click on + to add a new tab.
- 2) Select Explore By Searching

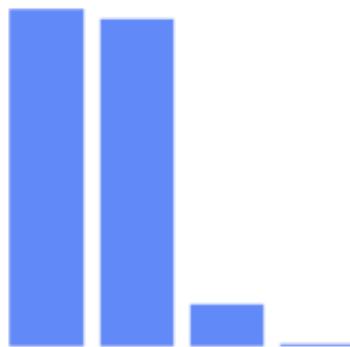


- 3) In What are you interested in? type Female versus Male and select create pie chart.

Q Female versus Male X

**Create bar chart**

View Row Count per Q20GENDER.



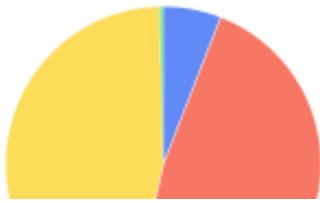
**Create treemap**

View distribution of Q20GENDER.



**Create pie chart**

Compare Row Count per Q20GENDER.

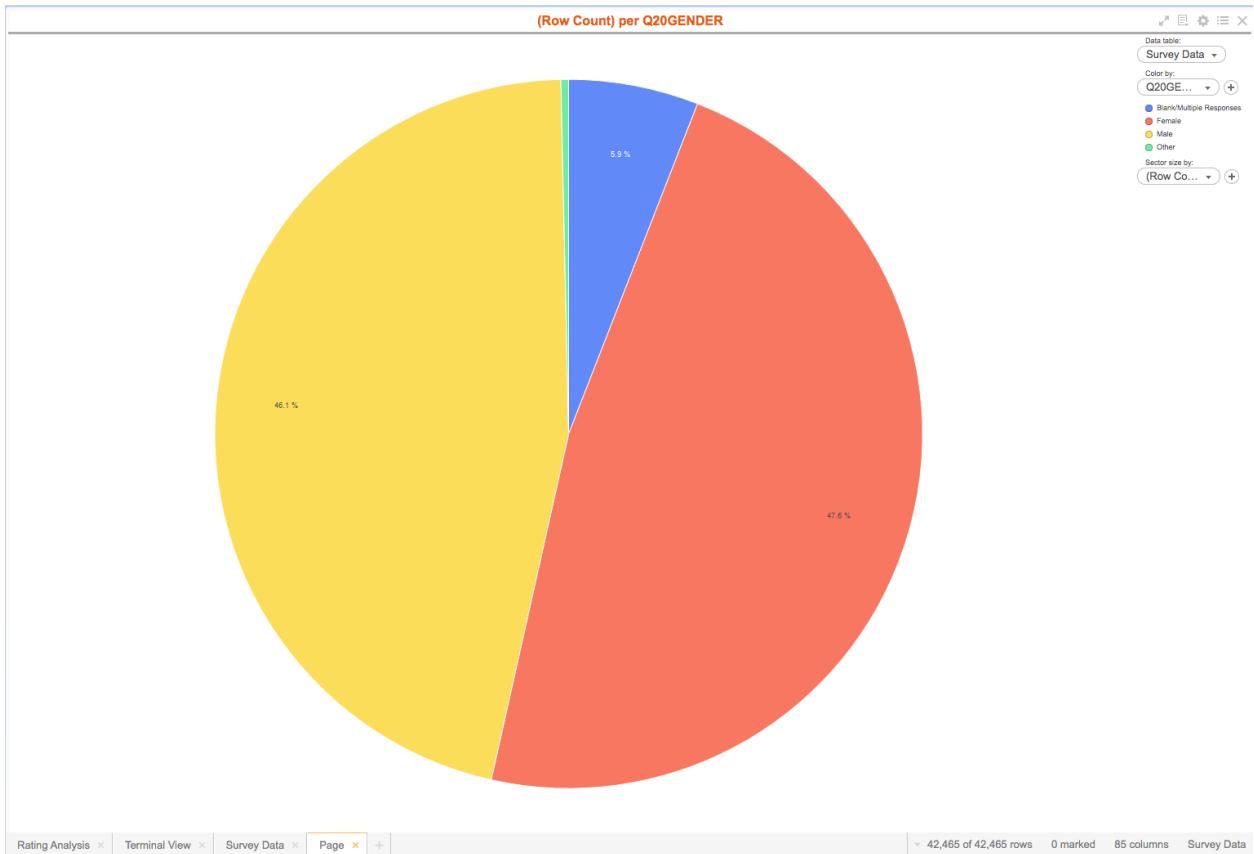


**Create waterfall chart**

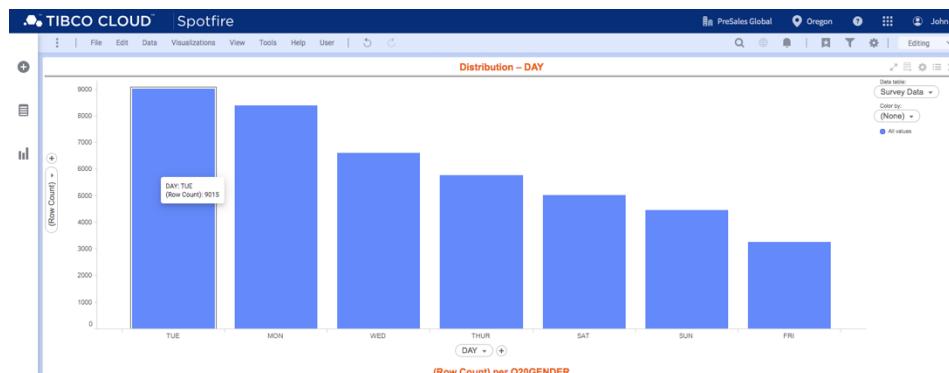
Show Row Count accumulated per Q20GENDER.



- 4) A pie chart will be created showing the result of your query.



- 5) Try other queries, select the spy glass icon in the upper right menu, and type Day and select create bar chart.



## Summary

We just saw how easy it is to import an analysis to the cloud and use natural language processing to add visualizations.

