

***On Boarding***  
***Postman, API and FAQ***

***Current by GE - Cities***

*December 2018*



## *Hackathon – General FAQ for Cities*

Copyright©2018 General Electric Company. All rights reserved.

GE, the GE monogram, Predix, and Current powered by GE, are either registered trademarks or trademarks of General Electric Company. All other trademarks are the property of their respective owners.

This document may contain Confidential/Proprietary information of GE, GE Software, Current powered by GE, and/or its suppliers or vendors. Distribution or reproduction is prohibited without permission.

THIS DOCUMENT AND IT CONTENTS ARE PROVIDED "AS IS," WITH NO REPRESENTATION OR WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. ALL OTHER LIABILITY ARISING FROM RELIANCE UPON ANY INFORMATION CONTAINED HEREIN IS EXPRESSLY DISCLAIMED.

Access to and use of the software described in this document is conditioned on acceptance of a duly executed agreement with GE and compliance with its terms.



## Table of Contents

Overview .....	3
Postman .....	3
Download Postman .....	3
Pre-Requisite Step: Download and Import the Postman Collection and Environment files .....	4
Environment setup .....	5
Directions for updating the Postman token .....	6
Update client Token - Introduction .....	6
Update client Token - Steps .....	6
Manually Update the UAA URL field.....	7
24 Hour Token.....	9
Exporting Successful Query as Code .....	9
Calling an API .....	11
Search Parameters .....	12
API Response Options .....	12
WebSocket.....	14
Using WebSocket API calls instead of REST API calls .....	14
CityIQ – Python - WebSocket - example code.....	14
CityIQ – Java – WebSocket - example code .....	15
CityIQ – Javascript – nodeJS – WebSocket - example code.....	17
FAQ .....	18
Is there a way to get familiar with using the APIs? .....	18
How can I run the APIs?.....	18
What is the PREDIX Zone-ID? Is it the same as the API Service Subscription? .....	18
Can I run APIs from different datasets, then combine the data for my APP? .....	18
FAQ – API Recipes.....	19
How long have these cars been parked? .....	19
References .....	21
Intelligent Environments – CityIQ .....	21
Reference - API documentation .....	21

## Overview

This document is designed for Current by GE API users. Make sure you have the following items at the start of development, which you should receive by email or from your resources hub. If missing any of the following items, please email [support.cityiq@ge.com](mailto:support.cityiq@ge.com).

1. *The **Developer-Onboarding-Sheet-Production**. This document contains what you need to get started - links to documentation, login information, authentication information, and a simple curl command.*
2. *Download the **Hackathon VPC Postman Collection JSON** file. This JSON file imports Postman collections, which allows you to use Postman to call the APIs and assets. This is not unique and is the collection used in demos for sampling the API.*

This file can be named as follows:

- *IC-VPC-APIs-Hackathon - OnBoarding.postman\_collection.json.*

3. *Download the **Hackathon VPC Postman Environment JSON** file. This file organizes the variables and URLs utilized by Postman. It is unique to your City and contains some empty variables which need to be completed with your credentials.*

The file is can be named as follows:

- *IC-VPC-APIs-Hackathon-OnBoarding.postman\_environment.json.*

You need both files to correctly access Postman. Also, these two Postman files are used in concert with each other; therefore, you will not get successful API responses if the Collection and Environment do not match.

If you do not have these artifacts, contact your Hackathon organizer or email [support.cityiq@ge.com](mailto:support.cityiq@ge.com) before beginning.

## Postman

One of the best ways to utilize the *Current by GE* Intelligent Environments APIs and other RESTful APIs is to use Postman, an open source software 'toolchain' from Postdot Technologies, Inc. Before developing apps and running APIs using Postman, you must:

- *Download and install Postman collections.*
- *Install the Current by GE specified Postman environment.*

### Download Postman

[Postman](#) enables you to develop using *Current by GE* APIs. You need the basic Postman version, which is free to use, and can be downloaded directly onto your development machine.

- *First, download and install Postman.*

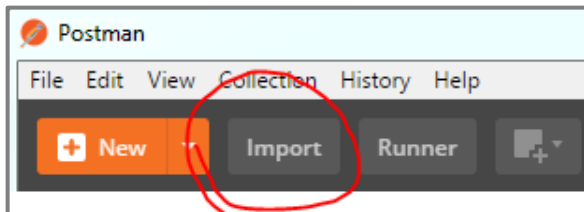
- To do so, click this link: [Install Postman](#)

### *Pre-Requisite Step: Download and Import the Postman Collection and Environment files*

**Note:** You need to import the collection and environment to Postman for everything to work properly. Collections allow you to organize your calls (GET, POST, etc.) from a single UI.

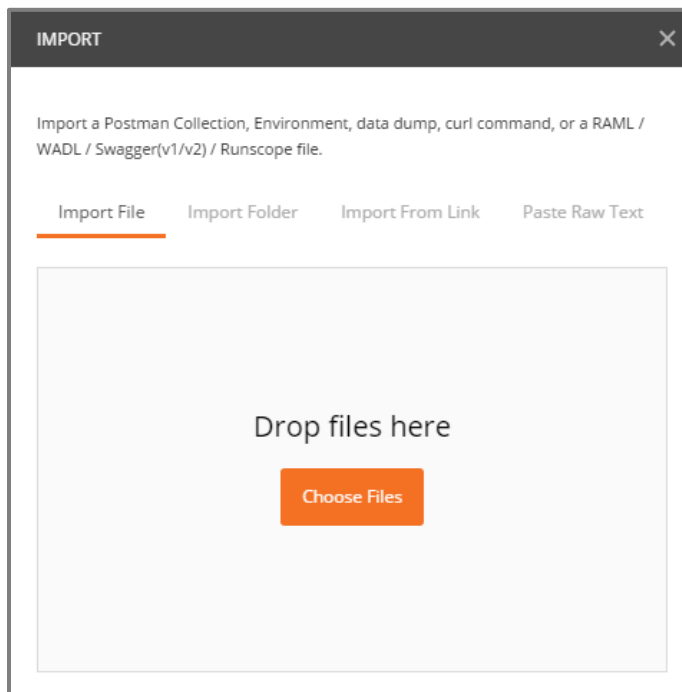
#### **Download the and save JSON files (Collections and Environment).**

1. Save these files to a location on your system; the **Downloads** folder is fine.
2. Open **Postman** and click **File / Import** OR the **Import** button to open the Importer.



There are two ways to import the files:

- You can individually select the files from where you have stored them by clicking on the '**Choose Files**' option, or;
- You can simply open your file explorer, then drag and drop the Collection and Environment files onto the **Drop Files here** window that opens when you click on **Import**.



- *Postman will automatically load the files.*

**Note:** You can obtain general information about the Postman collections and how they work here: [Get started with Collections](#)

### *Environment setup*

You need to setup your environment so it can utilize variables in the API calls.

You can import an environment in the same way you import a collection:

- *Open **Postman***
- *Click **File / Import** to open the Importer.*
- *You can select files from where you have stored them or drag and drop the Collection and Environment files into the Importer. Postman will automatically load the files.*

**Note:** For more information, click [Setting up an environment with variables](#).

**Important Note:** You will receive the following files either packaged with this document or by email : Onboarding Credentials Sheet, Postman collection and environment json files Please contact your CityIQ representative or administrator if you do not have these materials or email [support.cityiq@ge.com](mailto:support.cityiq@ge.com)

**Note:** If you do not import and install Postman collections and setup an environment, Postman will not work properly! Use the following instructions to setup Postman, and get help with the setup if you need it.

## Directions for updating the Postman token

- Generate the client token and use Postman.

### Update client Token - Introduction

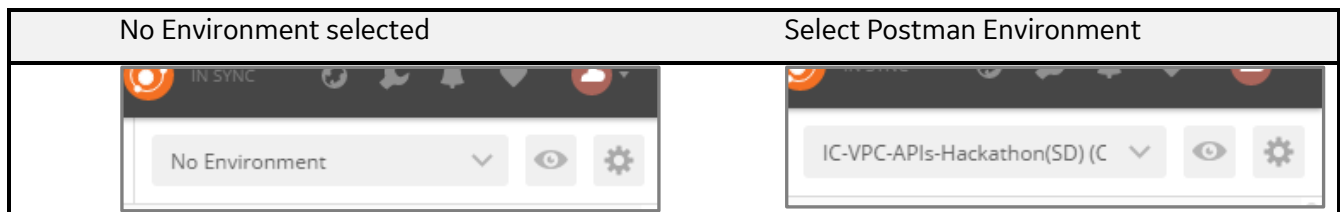
Before starting, make sure you have your **Developer-Onboarding-Sheet-Production** document, which will include:


1. User Account & Authentication (UAA) information;
2. **Client ID & Client Secret** login information or obtain this information from the person who will setup your account.
3. [Complete Pre-Requisite of loading Collection and Environment.](#) Complete this step before beginning!

### Update client Token - Steps

Follow the steps below to update your client token:

- Select the correct **Environment** (top right in Postman in **Manage Environments** – see screenshot below). By default, the drop down shows **No Environment**; select the Environment needed from the OnBoarding zip file.



- Authorization tab: set it to **Basic Auth**.
- Open the **Collection** on the left sidebar;
- Open the **Tokens** folder;
- Select **GET | Get client Token** in the **Tokens** folder;
- Click the **Send** button to update the client token. [  ]

**Note:** When generating the client token, the **Headers** tab should already contain the updated **Basic Auth. Key**.

It may look like this: **< Y2xpZW50aWQ6QGNsaWVudHNIY3JldA==>**


The **GET** call uses the following URL for the token:

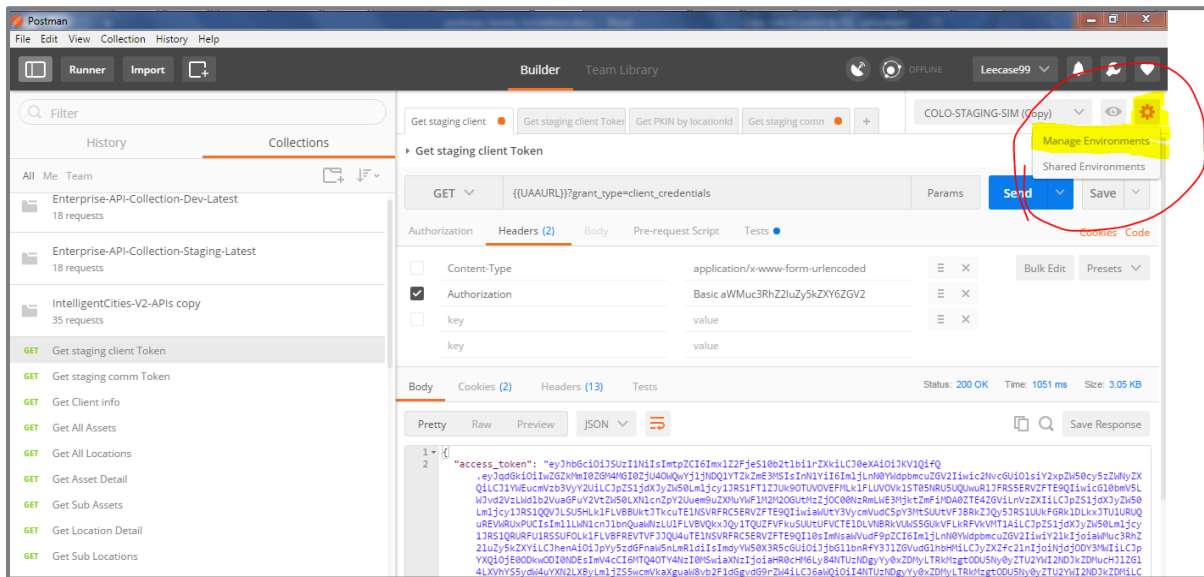
```
{{UAAURL}}/oauth/token?grant_type=client_credentials
```

The UAA URL should already be in the Environment key value variables, and will populate this variable automatically.

## Manually Update the UAA URL field

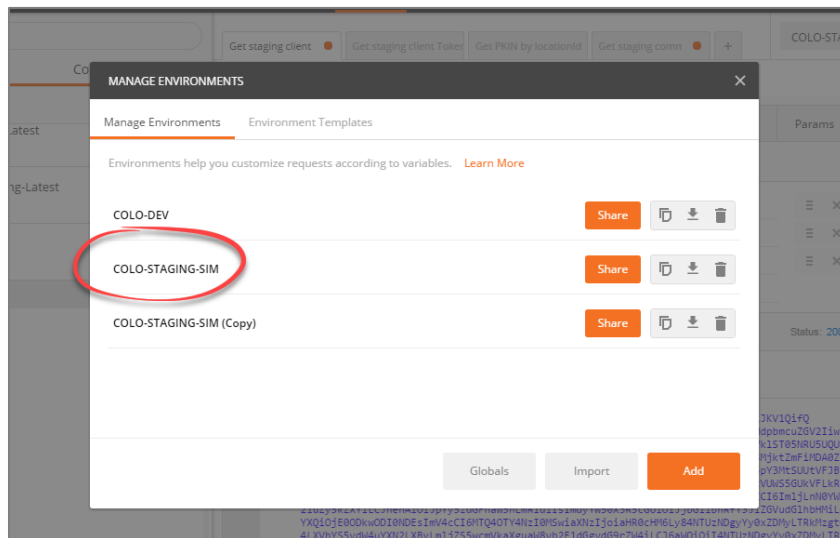
**Note:** If the UAAURL is missing or not updated in the variables, follow this process:

- Click the sprocket icon (far upper right); [  ]
- Click Manage Environments;



- Select the correct Environment name that you are using.

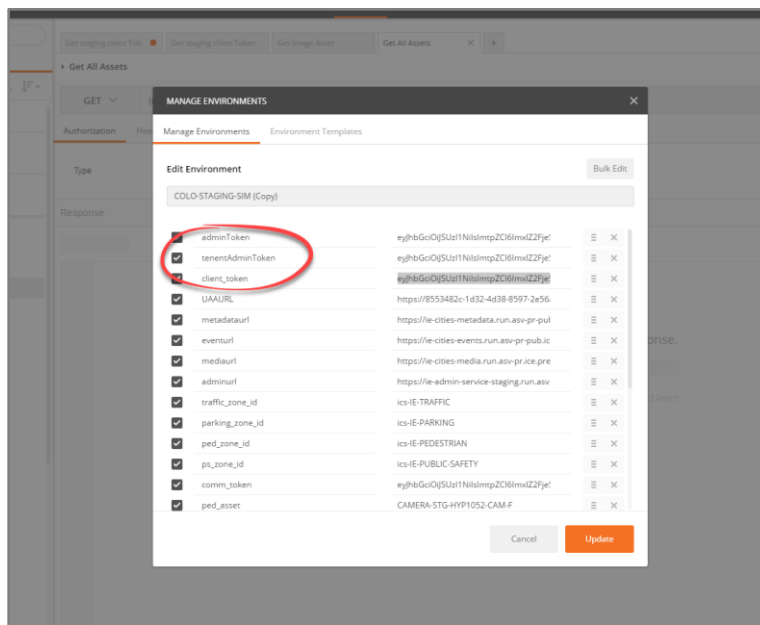
For this example, click **COLO-STAGING-SIM**





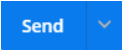
- This opens the **Edit Environment** screen. This view shows a list of key value variables. The client token key value is the value that will get updated with a new client token value.

**Note:** Users must input the URLs for metadata service, event service, and media service as they are provided by their administrator. As well, users need to input the Predix Zone IDs where they belong (i.e. parking with parking).



- For Cities users, the URL is provided in the *Developer-OnBoarding - Sheet*. If you are not given a UAA URL but have a client ID and client secret, please contact [support.cityiq@ge.com](mailto:support.cityiq@ge.com).

To ensure you have the correct **UAA URL** address:

1. Copy and paste the address from your **Developer-Onboarding-Sheet-Production** into the '**UAAURL**' variable field.
2. Copy and paste your **Metadata URL**, **Event Service URL**, **Media Service URL** (if provided) and **Predix-Zone Ids** to their indicated field.
  - **Note:** In the environment provided in this package, these values are all empty and therefore must be filled in to obtain any access to the system.
3. Click **Update**.
4. Click '**X**' to exit the **Edit** screen.
5. Finally, in Postman, go to the **Get client token** call and click **Send** [  ]

This generates the client token and replaces the client token variable value in the Environment.

## 24 Hour Token

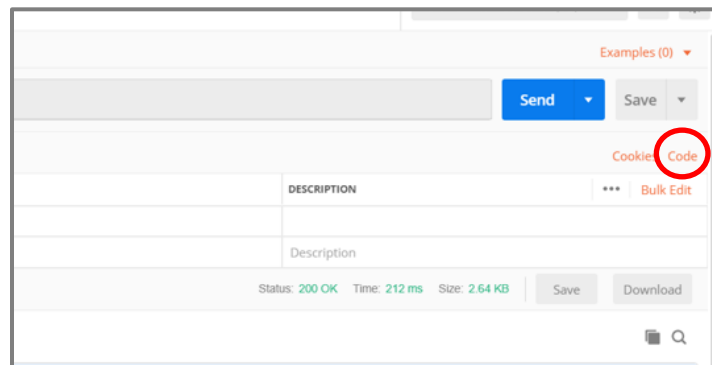
**Important Note:** Tokens expire every 24 hours. You need to regenerate the token after 24 hours of use. Data Access Errors can occur in Postman simply because you have not regenerated the Client Token.

Simply repeat the process used in the section [Update client Token](#) in this document.

A good practice is to regenerate the Token right when you open Postman for the day as part of your routine, especially if you are using Postman often.

## Exporting Successful Query as Code

To get started with your own REST calls using the language of your choice, you can click on **Code**. This is available in the upper right-hand corner of your query.



- Select **Code** to view your query in any language supported by Postman in a pop-up box:

GENERATE CODE SNIPPETS

Python Requests

Copy to Clipboard


```
1 import requests
2
3 url = "https://890407d7-e617-4d70-985f-01792d693387.predix-uaa.run.aws-usv02-pr.1ce.predix.io/oauth/token"
4
5 querystring = {"grant_type": "client_credentials"}
6
7 headers = {'cache-control': 'no-cache'}
8
9 response = requests.request("GET", url, headers=headers, params=querystring)
10
11 print(response.text)
```

"8f67039b1cc0485294823a072904374c"

## Calling an API

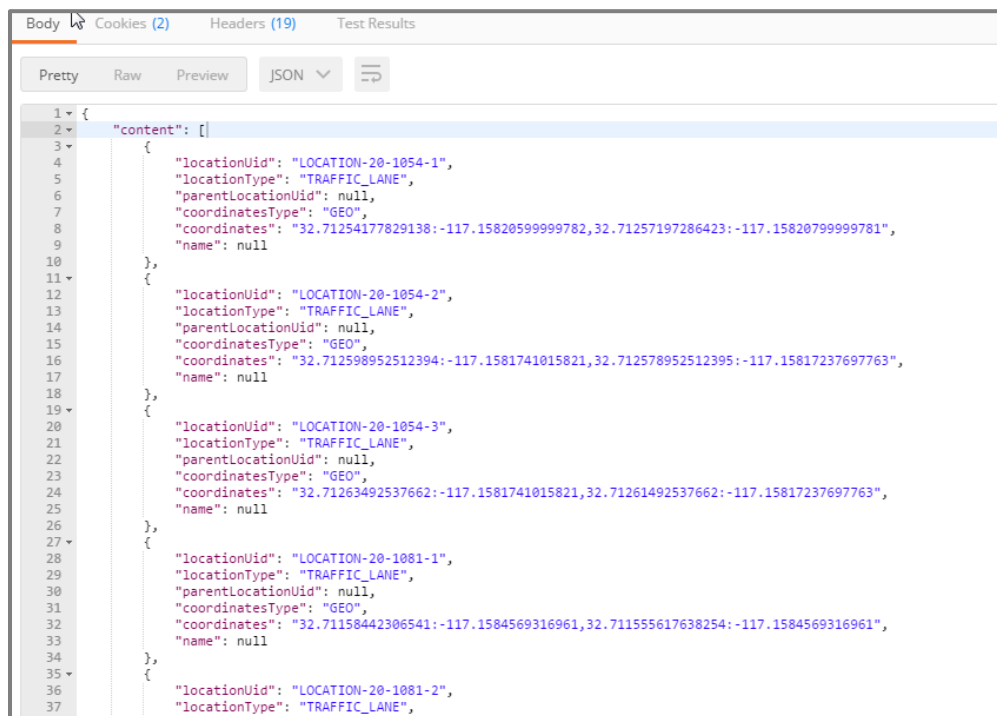
The process for making an API call is the same as the process for updating a client token.

Follow the steps below to call an API:

- In Postman, open the folder containing the API call (Events, Metadata, or Media).
- Click on the API name (for example: **GET** Get All Locations).
- Click **Send** in the upper right-hand corner of Postman [  ]



Your API response will appear as follows:



## Search Parameters

There are various ways to search parameters.

You can search with string input parameters such as `assetUid`, `locationUid`, `eventTypes`, `assetType` and `mediaType`. Additionally, you can search by location. Searching by location requires you to format your query to input two coordinate locations that result in a bounding box (`bbox`).

This bounding box is a parameter that bounds your search area and establishes the periphery of a searchable area for assets and nodes. It is specified with decimal geographic coordinates.

```
bbox= "<lat1:long1>,<lat2:long2>"
```

*example*

```
bbox = "45.467306:-73.714244,45.459210,-73.699994"
```

Where `lat1` and `long1` are two coordinates located northwest of `lat2` and `long2`. These values are decimal coordinates with 6 decimal places and can be obtained visually from Google maps.

**Note:** To understand further how the location search function works, please reference the API section [Get List of Locations](#).

## API Response Options

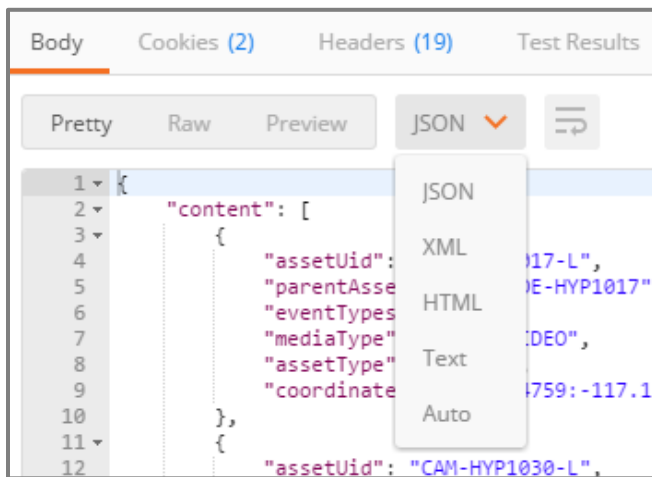
You can view Postman API Responses in 3 display formats:

- *Pretty*
- *Raw*
- *Preview*


You can change the output format of the API response into several formats, including:

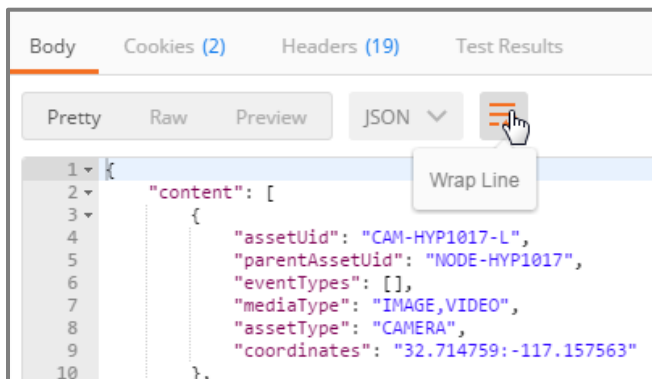
- *XML*
- *HTML*
- *Text*
- *Auto (JSON default)*

JSON is the automatic default output response displayed in Postman.



You can wrap the line of the API response with the **Wrap Line** button.

- Click on the icon [  ] to enable it and wrap your output; click on it once again to unwrap your output.



## WebSocket

### Using WebSocket API calls instead of REST API calls

This section demos three simple WebSocket demos and is yours to copy / paste. WebSocket allows you to request specific types of events as they occur in near real time.

1. [WebSocket Python sample](#)
2. [WebSocket Java sample](#)
3. [WebSocket Javascript NodeJs sample](#)

CityIQ – Python - WebSocket - example code

```
import os
import time
import thread

# pip install WebSocket-client
import WebSocket

# pip install predix
import predix.security.uaa

def get_auth_token(uaa_id):
    uaa_uri = '---UAA URL HERE ---'
    os.environ['PREDIX_SECURITY_UAA_URI'] = uaa_uri
    uaa = predix.security.uaa.UserAccountAuthentication()

    client_id = '---Client ID HERE ---'
    client_secret = '---Client Secret HERE ---'
    uaa.authenticate(client_id, client_secret)

    return uaa.get_token()

def on_message(ws, message):
    print(message)

def on_close(ws):
    print('### closed ###')

def on_open(ws):
    print('### connected ###')

# Example for Traffic Predix Zone
ws.send('{"bbox": "--LAT Pt 1--:--LONG Pt 1--:--LAT Pt 2--:--LONG Pt 2--", "eventTypes": ["TFEVT"]}')

# Example for Parking Predix Zone
# ws.send('{"bbox": "--LAT Pt 1--:--LONG Pt 1--:--LAT Pt 2--:--LONG Pt 2--", "eventTypes": ["PKIN", "PKOUT"]}')

if __name__ == '__main__':
```

```

WebSocket.enableTrace(True)

# Use Predix-Zone-ID to match events of interest
# cityiq_zone = '---Parking Predix Zone HERE ---'
cityiq_zone = '---Traffic Predix Zone HERE ---'

cityiq = '---WEBSOCKET URL HERE ---'
token = get_auth_token('shhh! this is secret so replace with predix uaa id')
headers = {
    'Authorization': 'Bearer ' + token,
    'Predix-Zone-Id': cityiq_zone,
    'Cache-Control': 'no-cache',
}
ws = WebSocket.WebSocketApp(cityiq,
    header = headers,
    on_message = on_message,
    on_close = on_close)
ws.on_open = on_open
ws.run_forever()

```

## CityIQ – Java – WebSocket - example code

```

/*
 * Copyright (c) 2016 GE. All Rights Reserved. GE Confidential: Restricted
 * Internal Distribution
 */
package web;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.atomic.AtomicReference;

import javax.websocket.ClientEndpointConfig;
import javax.websocket.ContainerProvider;
import javax.websocket.DeploymentException;
import javax.websocket.Endpoint;
import javax.websocket.EndpointConfig;
import javax.websocket.MessageHandler;
import javax.websocket.OnMessage;
import javax.websocket.Session;
import javax.websocket.WebSocketContainer;

```



```

public class WSClient {

    private static Object waitLock = new Object();

    @OnMessage
    public void onMessage(String message) {
        System.out.println("Received msg: " + message);
    }

    public static void main(String[] args)
        throws URISyntaxException, DeploymentException, IOException, InterruptedException {
        Session = null;
        System.out.println("Start...");

        WebSocketContainer container = ContainerProvider.getWebSocketContainer();

        final AtomicReference<String> message = new AtomicReference<>();
        final CountDownLatch latch = new CountDownLatch(1);

        Endpoint endpoint = new Endpoint() {

            @Override
            public void onOpen(Session session, EndpointConfig config) {
                session.addMessageHandler(new MessageHandler.Whole<String>() {

                    @Override
                    public void onMessage(String content) {
                        message.set(content);
                        latch.countDown();
                    }
                });
            }
        };

        URI uri = new URI("wss://---WebsocketURI---/events");
        ClientEndpointConfig.Configurator configurator = new ClientEndpointConfig.Configurator() {

            @Override
            public void beforeRequest(Map<String, List<String>> headers) {
                headers.put("Authorization", Collections.singletonList("Bearer ---TOKEN HERE---");
                headers.put("Predix-Zone-Id", Collections.singletonList("---PREDIX-ZONE-ID HERE---"));
            }
        };

        ClientEndpointConfig clientConfig =
        ClientEndpointConfig.Builder.create().configurator(configurator).build();

        session = container.connectToServer(endpoint, clientConfig, uri);

        System.out.println("Connecting to " + session.getRequestURI());
        session.getBasicRemote()
            .sendText("{\"bbox\":\"--LAT Pt 1--:--LONG Pt 1--:--LAT Pt 2--:--LONG Pt 2--\"}");
        session.send("\"eventTypes\":[\"PKIN\"]");
    }
}

```

```

    latch.await();
  }
}

```

## CityIQ – Javascript – nodeJS – WebSocket - example code

```

/* Before Beginning save script as WebSocket.js, then:
  1. input your client client_id and client_secret to developer.credentials
  2. verify node.js installed with this command: $ node -v
  3. verify node package manager installed with this command: $ npm -v
  4. navigate to the directory you have saved this file.
  5. install node-fetch and ws packages with this command:
    $ npm install node-fetch ws
  6. in the directory that this file is saved, run this command:
    $ node WebSocket.js
*/

const credentials = {
  uaa: '---UAA URL HERE ---',
  WebSocket: '---WEBSOCKET URL HERE ---/events',
  developer: '---Client ID HERE ---;---Client Secret HERE ---',
  parking: '---Parking Prediz Zone HERE ---',
}

const fetch = require('node-fetch')
const WebSocket = require('ws')
const btoa = str => new Buffer(str).toString('base64')

async function listen() {
  let result = fetch(credentials.uaa, {headers:{authorization: 'Basic '+btoa(credentials.developer)}})
    .then(res => res.json())
  let token = (await result).access_token
  let ws = new WebSocket(credentials.WebSocket,
    {headers: {authorization: 'Bearer '+token,
      'predix-zone-id': credentials.parking }})

  ws.on('open', function open() {
    console.log('listening')
    ws.send(JSON.stringify({assetUID:"---Known AssetUid HERE ---",eventTypes:["PKIN","PKOUT"]}));
  })

  ws.on('message', data => console.log(data))
  ws.on('error', err => console.log(err))
}

listen()

```

## FAQ

### *Is there a way to get familiar with using the APIs?*

Start by looking over the online API web-help documentation. You can get an overview on how the CityIQ APIs work in the [General APIs](#) section of the API web-help. After familiarizing yourself with the **General** section proceed to each individual API in [Intelligent Cities APIs](#) section.

### *How can I run the APIs?*

We provide both real and simulated data streams for both historical data (via Metadata, Event and Media URLs) and near real-time data (via WebSocket Service URL) for each user. Check your Onboarding Sheet or your onboarding email for the Service Subscriptions and URLs that are specific to you. If you are missing any of the credentials, please contact [support.cityiq@ge.com](mailto:support.cityiq@ge.com).

### *What is the PREDIX Zone-ID? Is it the same as the API Service Subscription?*

Yes, these are one and the same – they are different names for the same API Service. The **Service Subscription** is the API service name that Current created, and the ZoneID is a PREDIX moniker for the API service, and these both refer to the same entity.

### *Can I run APIs from different datasets, then combine the data for my APP?*

You can, but remember to keep track of the original data source, especially which **PREDIX-Zone-Id** the assets originate from. You may have access and be using multiple ZoneIds; they may have disparate data sources and are separated by the ZoneId.

For example, you might be attending a CURRENT hackathon event that gives you access to both Smart Cities and Smart Buildings APIs, but they use the same CURRENT Login ID.

Once again, think of these as separate; the data itself is separated by a ZoneID, and in many cases, different ZoneIDs can rely on separate backend data sources, even if the front-end login credentials are the same.

The **Subscription ID** will need to be changed before calling an API in a different PREDIX ZoneID. Once again, find a way to keep track of the **Postman Collection | PREDIX ZoneID | AssetId** lineage from which the specific dataset originated.

## FAQ – API Recipes

### How long have these cars been parked?

Here is an example of how to build a Parking Scenario:

“I would like to see how long the cars in all city parking spaces have been parked during the lunch hour (12-1 PM). Right now, it is 1:15 PM. “

#### STEPS

- |  |                                     |
|--|-------------------------------------|
| 1) Run historical data <b>PKIN</b> API by locationId | Timestamp: 11:45 AM                 |
| 2) Run historical data <b>PKIN</b> API by locationId | Timestamp: 12:01 PM                 |
| 3) Run <b>PKOUT</b> API by locationId                | Timestamp: 1:00 PM                  |
| 4) RUN <b>PKIN</b> API by locationId                 | Timestamp: current time (@ 1:15 PM) |

Action	Timestamp	Reason
Run historical data <b>PKIN</b> API by locationId	Timestamp: 11:45 AM	Establish Baseline of cars in parking zone BEFORE you start the parking zone time-period for the next hour (12-1 PM).
Run historical data <b>PKIN</b> API by locationId	Timestamp: 12:01 PM	This is the START time for the parking time-period of NOON until 1 PM. These cars are within the parking zone's current time-period.
Run <b>PKOUT</b> API by locationId	Timestamp: 1:00 PM	This is the END time for the parking time-period of NOON until 1 PM.
RUN <b>PKIN</b> API by locationId	Timestamp: current time (@ 1:15 PM)	Time-period of NOON-1 PM has EXPIRED. Any cars parked from 12:01 PM that are still parked at this time will be assessed a parking ticket.

This model establishes a baseline of cars that were parked right before noon (11:45AM + 12:00PM), cars that are on the lot one minute after NOON (meaning they are fully IN the current parking zone time-period), determines which cars have left an hour later and have left the parking zone (1:00 PM), and which cars remain parked in the zone and are now late by fifteen minutes (1:15PM).

#### Process

First, compare the datasets, extract what is similar along the compared files, and look at the defined **EventIds** – these will be the identifiers of the specific cars that have remained parked over this total time period. These cars may be liable for a parking ticket!

You would repeat the same steps for the next parking time-period (1:00 PM until 2:00 PM).

**Note:** Currently, the Predix technology does not monitor the physical parking spaces or Zones “live”. It can only monitor the moving assets (i.e. **vehicles**) that are using the parking lot.

## References

### *Intelligent Environments – CityIQ*

If you are a developer, or want to build in-house applications, you can use the **Intelligent Environments by Current** to enable a valuable set of location - based services for your organization and your clients.

Start by looking at the CityIQ IoT Platform resources here:

<http://developer.currentbyge.com/cityiq>

Also, look at the *Predix Developer Network* resources:

<https://www.predix.io/>

On this site, Current reveals the capabilities of the overall Platform and its APIs, Reference Apps, and how to apply for a Business Partnership with Current.

### *Reference - API documentation*

*Developer documentation for the Intelligent Environments APIs.*

CityIQ – API Online help:

<https://ie-cities-docs.run.aws-usw02-pr.ice.predix.io>

Still need Help, Contact Us at [support.cityiq@ge.com](mailto:support.cityiq@ge.com)

Thank you for developing with us, we appreciate your participation and look forward to collaborating with you.

Your City IQ Development Team