current
powered by GE

# Developer Technical Training
# CityIQ APIs

San Diego API Working Session

January 30, 2019

# Training : Developer Technical Training – How To Consume CityIQAPIs

Attendees: Developers & other interested parties (City of San Diego data scientists/developers, SCALE SD, STIR winners, etc)

Agenda

- Intro to CityIQ
- API Structure
- Methods of Accessing the API (REST API vs WebSocket)
- Getting Started
  - Parking API
  - Traffic API
  - Pedestrian API
- Additional Questions / Wrap Up

current
powered by GE

# Intro to CityIQ

The CityIQ IoT Platform is an all-inclusive digital infrastructure living within street lamp posts and on the Cloud.

## San Diego Vision Zero

- Improving Safety and Reducing Accidents
- Improving City Planning and Road/Walkway Conditions

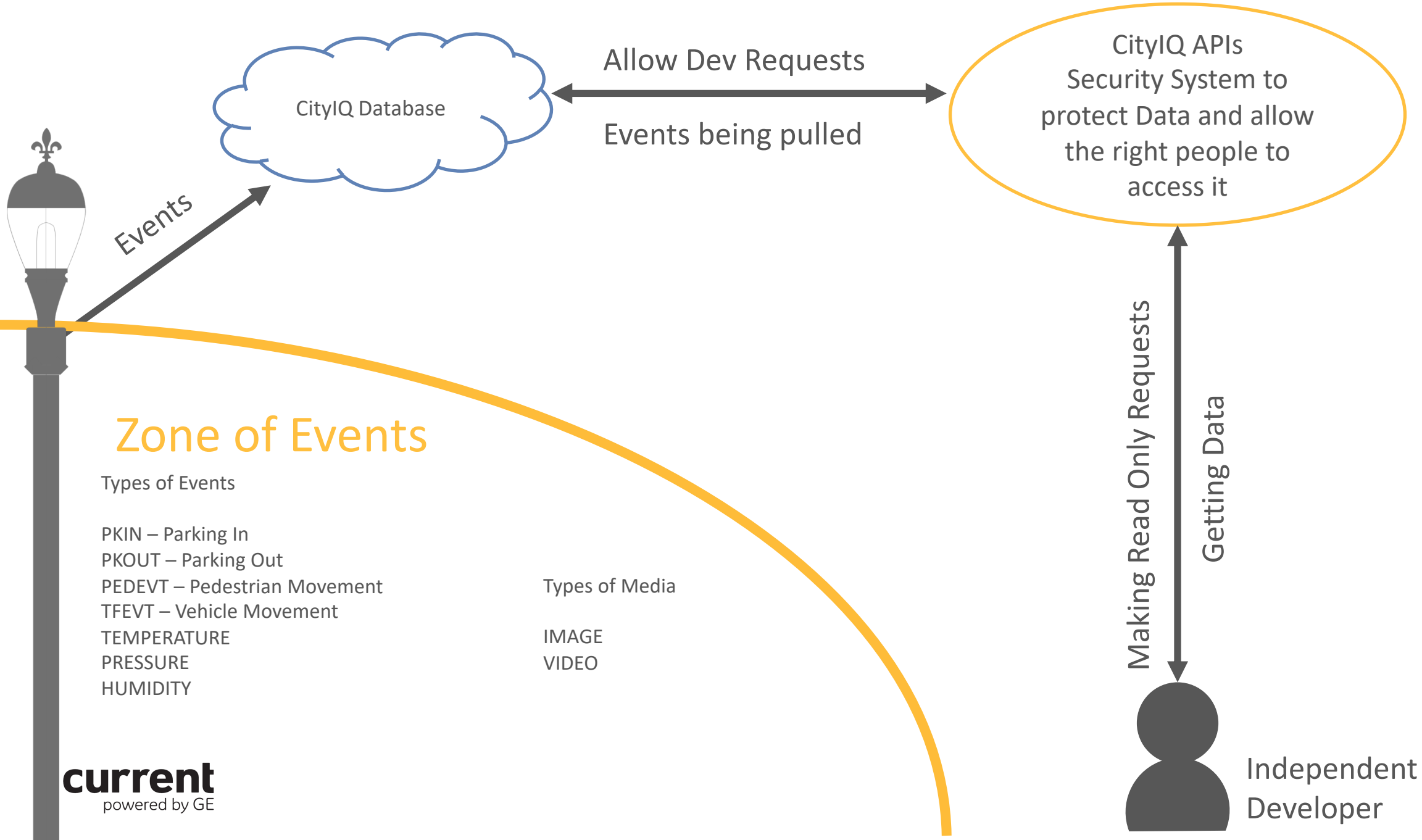
current
powered by GE

# Why is this training necessary?

Data produced by CityIQ Nodes is protected and managed by the CityIQ APIs

Understanding the APIs enables….

- **Distributors** to know who needs what and why.
- Smart City **Advisors** to know how to guide Developers and **Drive Innovation**
- **Developers** to use SD Data optimally and effectively




# Zone of Events

Types of Events

PKIN – Parking In
PKOUT – Parking Out
PEDEVT – Pedestrian Movement
TFEVT – Vehicle Movement
TEMPERATURE
PRESSURE
HUMIDITY

Types of Media

IMAGE
VIDEO

current
powered by GE

# API Structure

This will clarify the CityIQ Database Structure to help developers visualize the API

The following questions will be answered:

1. What information do I need when accessing the API?
2. What are the different services and why do I need them?
3. What are the authentication mechanisms for each service?
4. Why do I need Metadata?
5. What does the Asset/Location Event Tree Structure look like?

# API Structure

## Necessary Input Parameters

Client_ID and Client_Secret

Client_Token
- Obtained via UAA URL

*** Always needed**

Predix-Zone-ID
- Subscription ID

*** Always needed**

## Necessary URLs

1

User Account and Authentication Service
- Overarching URL Service which provides security for the overall API Services

2

MetaData Service
- Independent URL Service which provides clients information pertaining to the assets and locations available for monitoring.

3

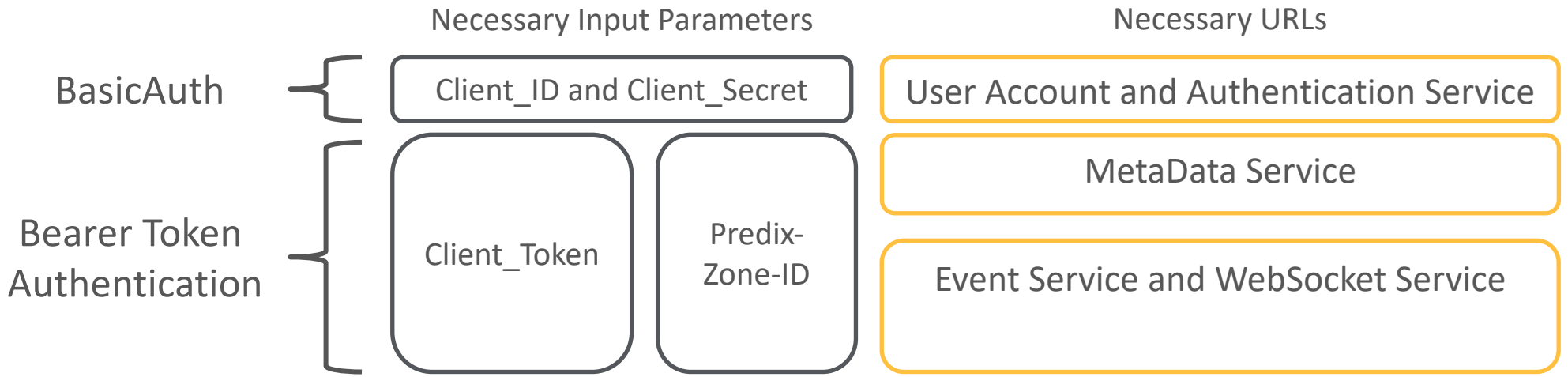
*Event Service and WebSocket Service**
- URL Services which provides clients with the events recorded at specific locations via specific assets.

current
powered by GE

# User Account and Authentication Service

BasicAuth

- Base 64 encoded client_id and client_secret
- Returns Token used for future credential validation



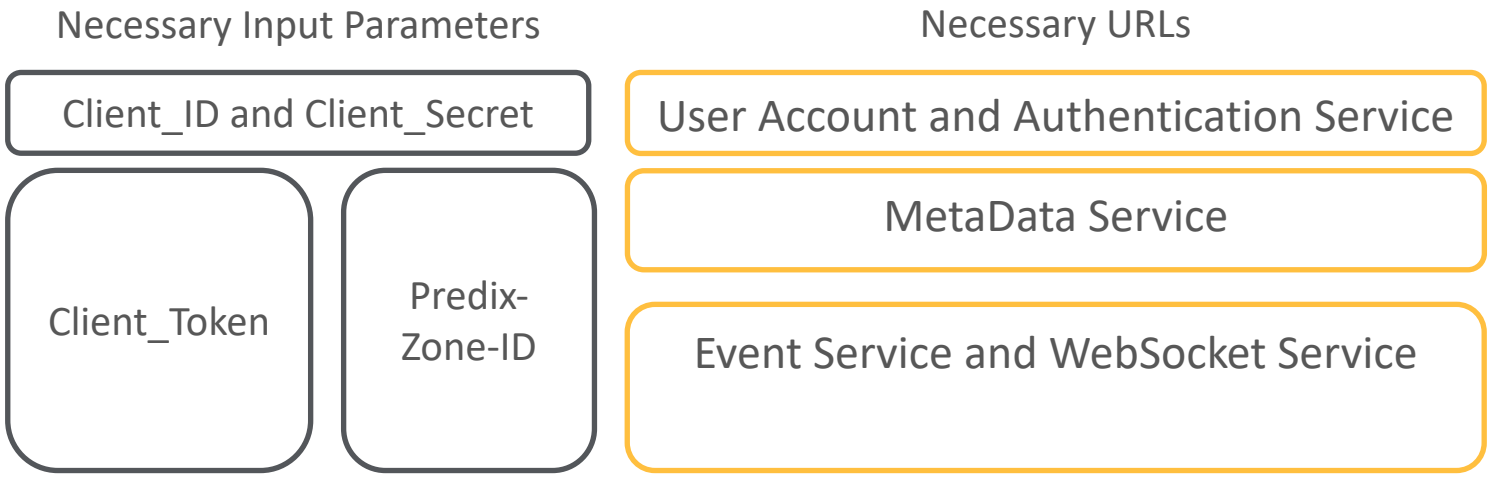
# Metadata Service

Bearer Token Authentication

- Requires BasicAuth Token & Predix-Zone-ID

Static Data

- Asset Details (Device Details)
- Location Details (Coordinates and Types)



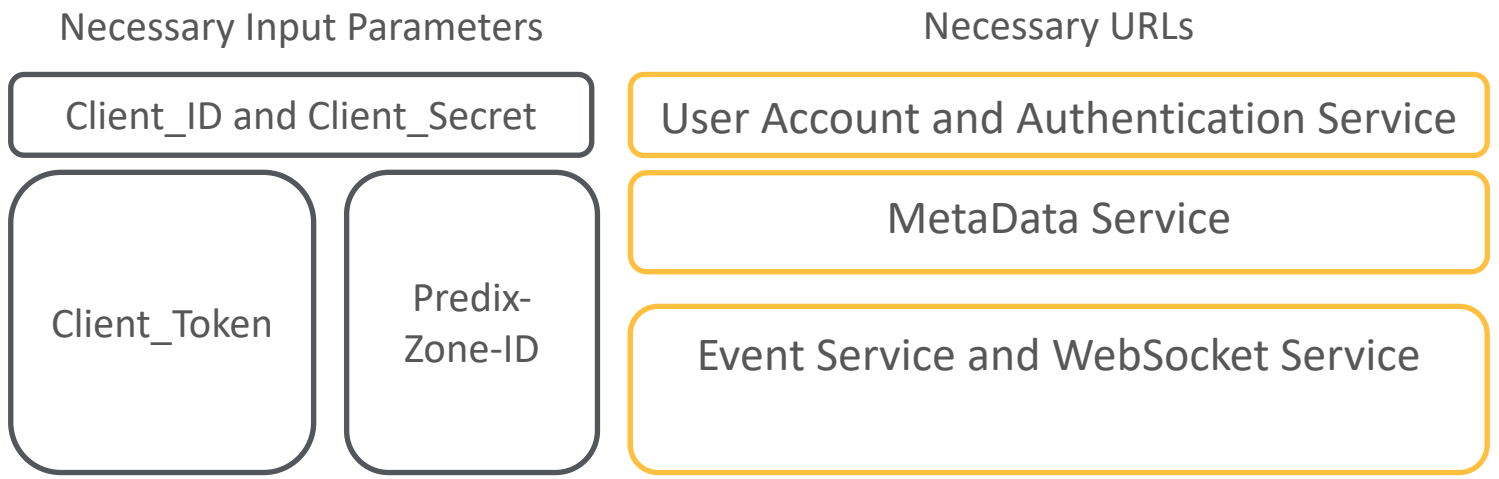
# Event Service and WebSocket Service

Bearer Token Authentication

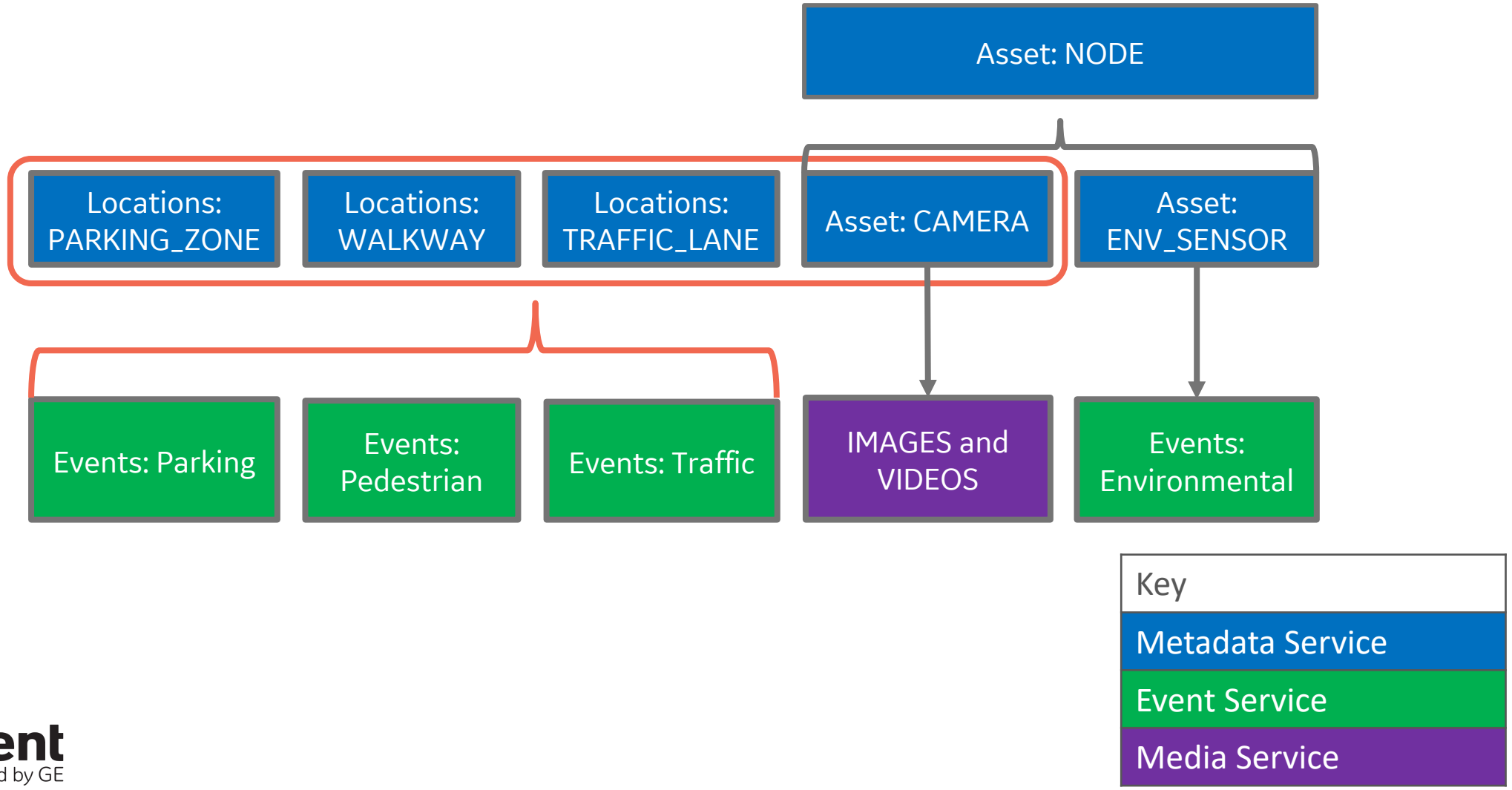
- Requires BasicAuth Token & Predix-Zone-ID

Dynamic Data

- Events recorded with Asset and Location Details



# Asset/Location Event Tree



# Methods of Accessing the API

REST API vs WebSocket
This will clarify the CityIQ Methods of Access to help developers understand implementation options

The following questions will be answered:

1. What is the difference between the REST API and the WebSocket?
2. When do I use the WebSocket?
3. What are the necessary credentials.  To access the REST API? And the WebSocket?
4. Why must I always use the REST API for UAA and Metadata calls?

# Methods of Accessing the API

Main Method

REST API:

- Historical Data Notifications
- Client notifies API when data is needed
- Postman demos the REST API

*Necessary when using UAA and Metadata Services*

Optional Method

WebSocket:

- Near real-time Data Notifications
- Opens a connection
- Custom programming necessary

*Only available when accessing events*

# Accessing Steps

REST API

WebSocket

## Necessary Input Parameters

Client_ID and Client_Secret

Client_Token
- Obtained via UAA URL

*** Always needed**

Predix-Zone-ID
- Subscription ID

*** Always needed**

## Necessary URLs

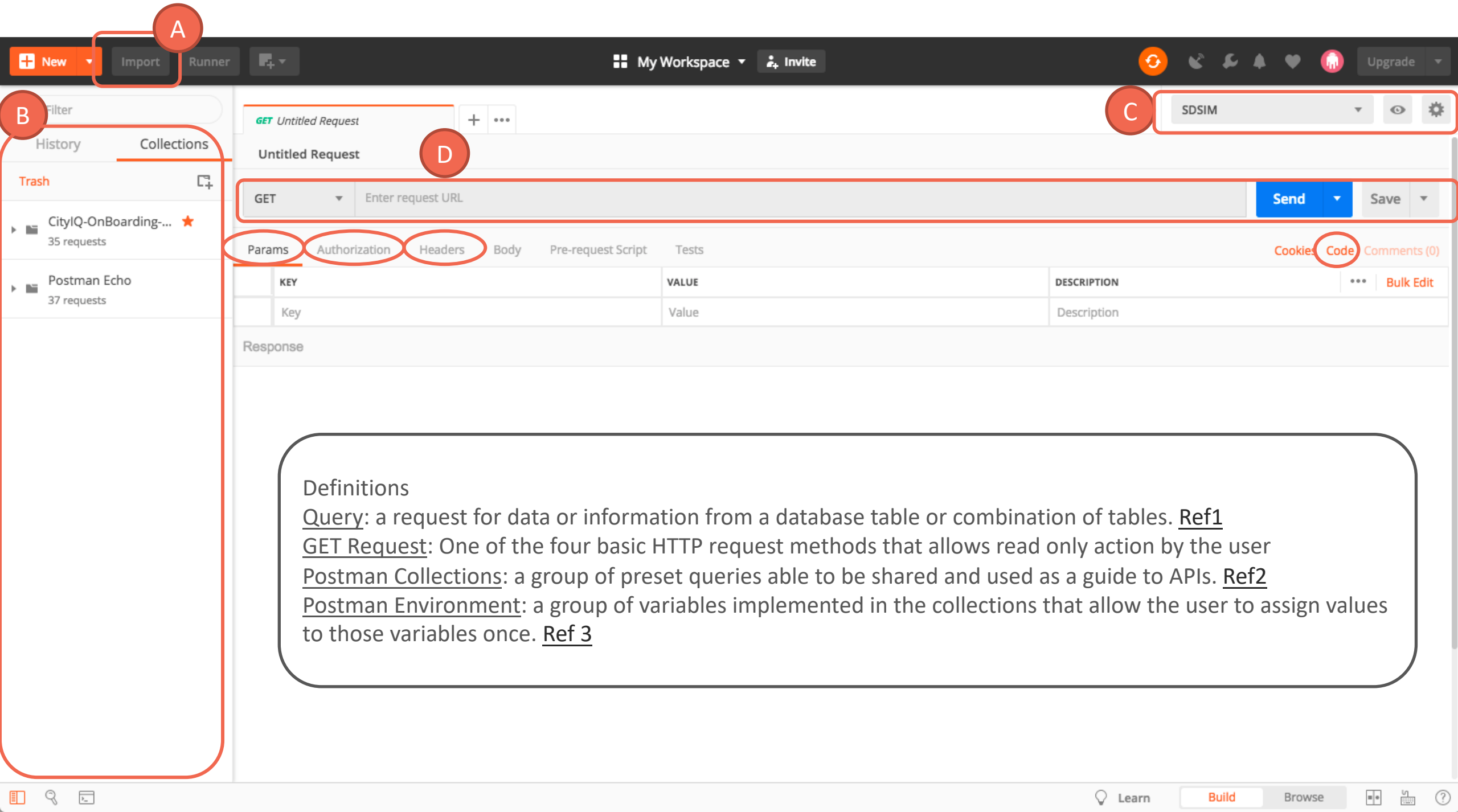
1 1 User Account and Authentication Service
- Overarching URL Service which provides security for the overall API Services

2 2 MetaData Service
- Independent URL Service which provides clients information pertaining to the assets and locations available for monitoring.

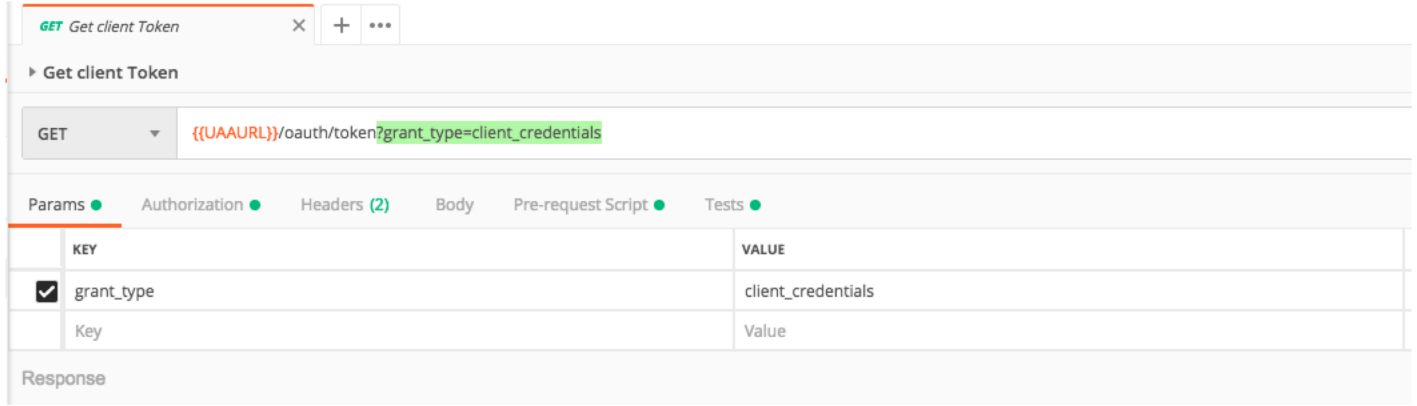
3 Events Service
- URL Service which provides clients with the events recorded at specified time intervals

3 WebSocket Service
- URL Service which provides clients the ability to stream event data

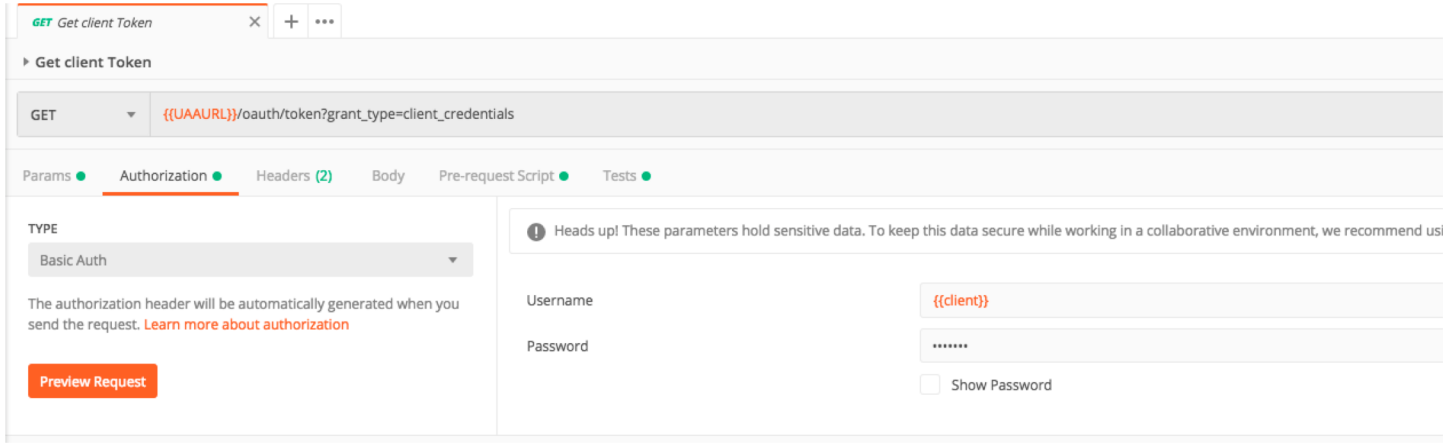
current powered by GE


A
New
Import
Runner
My Workspace
Invite
Upgrade
B
Filter
History
Collections
Trash
CityIQ-OnBoarding-...
35 requests
Postman Echo
37 requests
GET Untitled Request
Untitled Request
C
SDSIM
D
GET
Enter request URL
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Response
Definitions
Query: a request for data or information from a database table or combination of tables. Ref1
GET Request: One of the four basic HTTP request methods that allows read only action by the user
Postman Collections: a group of preset queries able to be shared and used as a guide to APIs. Ref2
Postman Environment: a group of variables implemented in the collections that allow the user to assign values to those variables once. Ref 3
Learn
Build
Browse

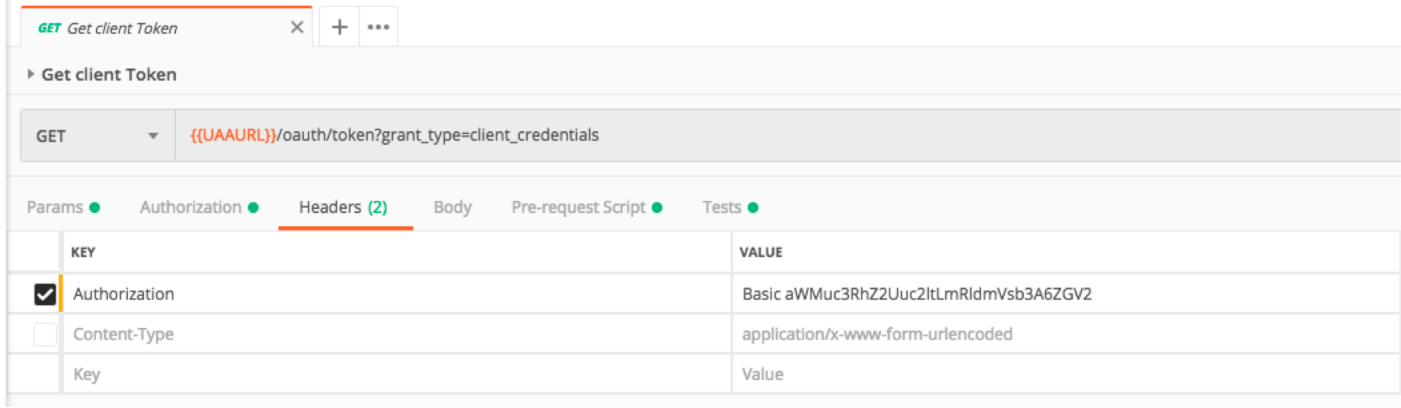
Params:




Authorization:




Headers:




current
powered by GE

# Getting Started

This will clarify the process of getting started using the CityIQ APIs

The following questions will be answered:

1. How do I use the REST API?
2. How do I use the WebSocket?
3. What is the process to obtain CityIQ dynamic data?
4. How do I get a client_token? Is it the same process when using the WebSocket?
5. What queries can I make to get metadata information?
6. What does metadata look like in the real world?
7. Where do I input the Predix-Zone-Id (Subscription-ID)?

# San Diego Simulated Data Credentials

Client_id: SanDiego

Client_secret : $aND!3g0

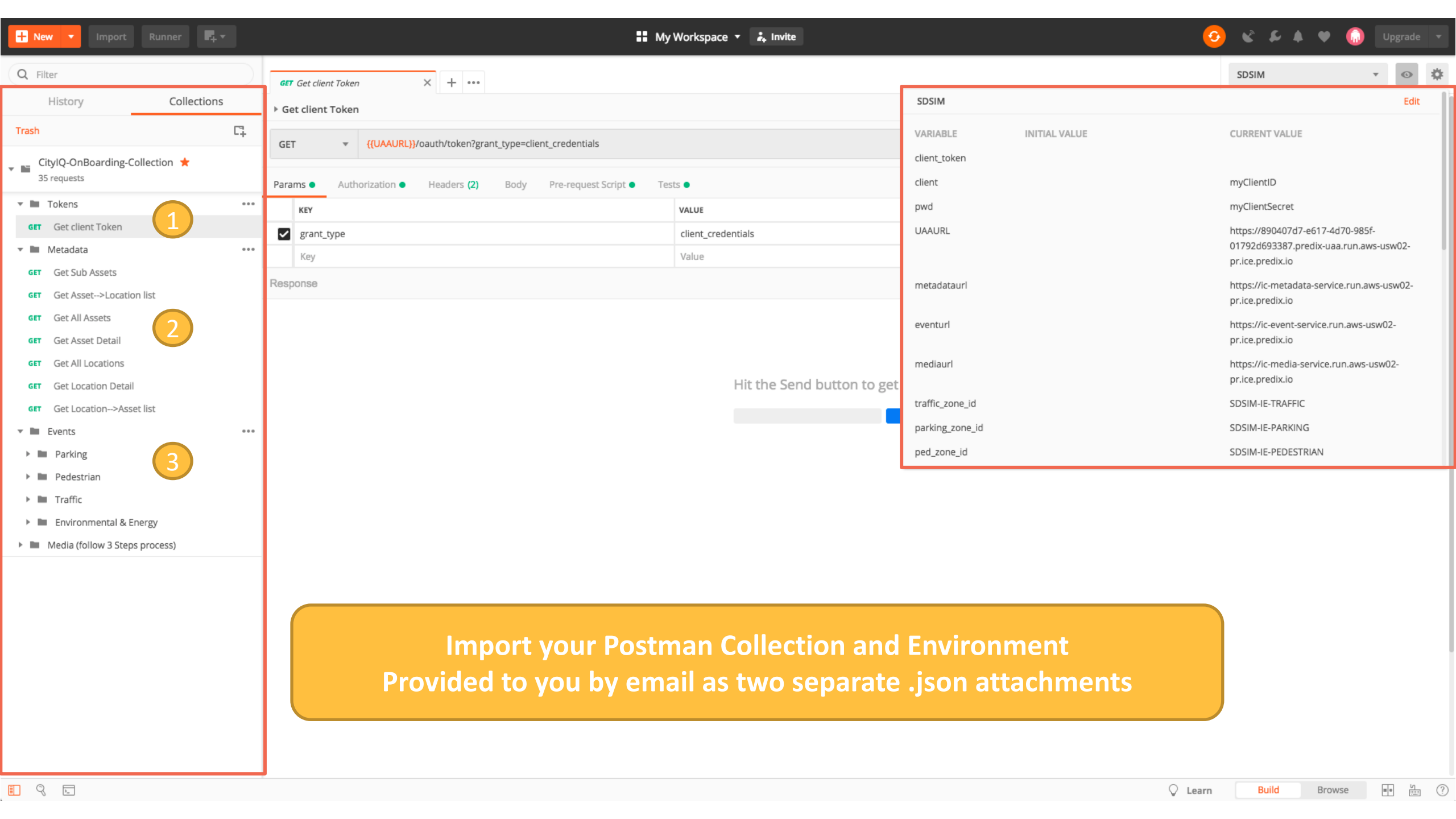
&

Postman Link:

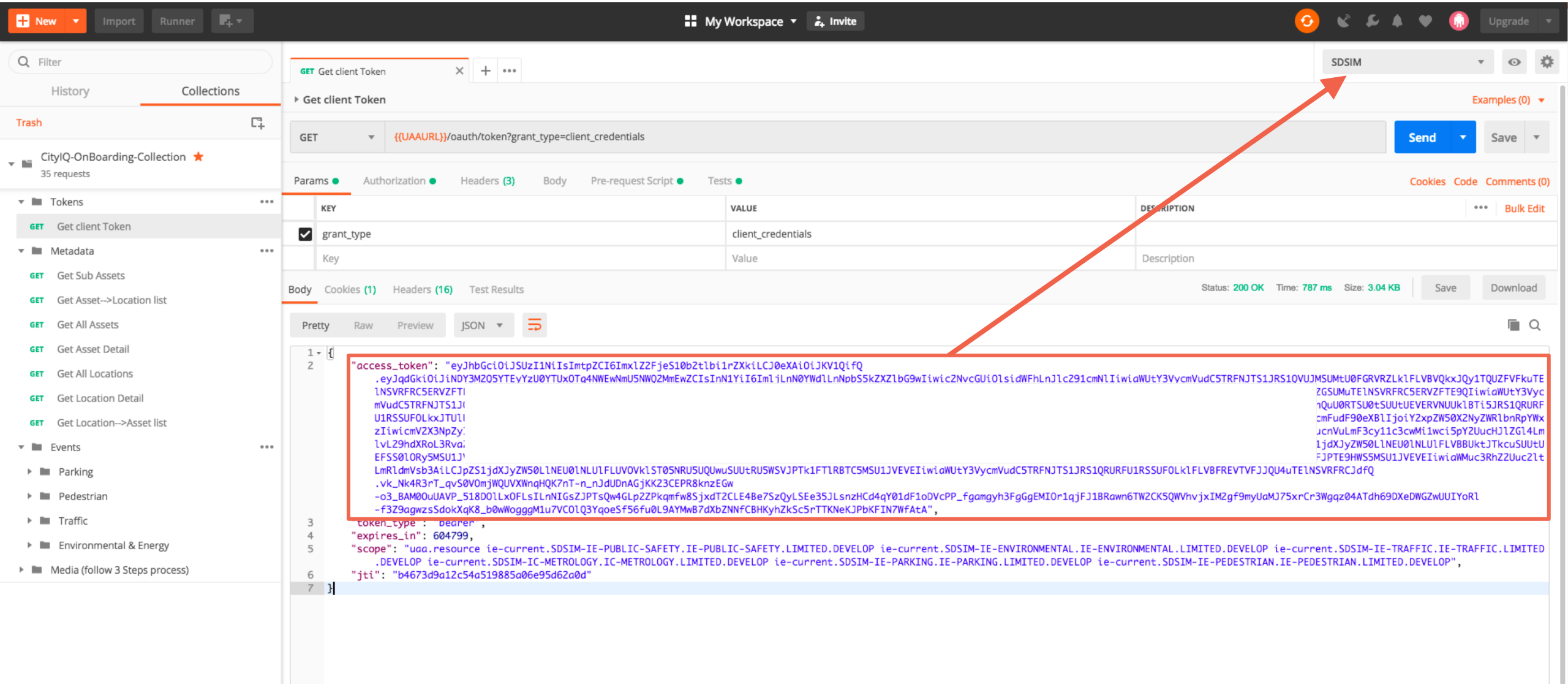
https://app.getpostman.com/join-team?invite_code=cc8b0ffa5ba954fd01d0dac299ac030b&ws=6eece5f6-4729-40bf-8b1c-5dbedc667286

# Getting Started
# 3 Step Process

1. UAA URL – Get Client Token
   - Token expires every 24 hours
   - Must make as REST API Request
   - Requires base 64 encryption of client_id and client_secret
2. Metadata Service URL – Find the Asset or Location you would like event data for
   - Must be done when configuring your app
   - Once assets/locations are known, not necessary to do again until new assets are available
   - Must make as REST API Request
3. Event Service URL – Ask the API for the specific events related to that Asset or Location
   - Can use REST API or WebSocket


New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
Trash
CityIQ-OnBoarding-Collection
35 requests
Tokens
GET Get client Token
Metadata
GET Get Sub Assets
GET Get Asset-->Location list
GET Get All Assets
GET Get Asset Detail
GET Get All Locations
GET Get Location Detail
GET Get Location-->Asset list
Events
Parking
Pedestrian
Traffic
Environmental & Energy
Media (follow 3 Steps process)
1
2
3
GET Get client Token
Get client Token
GET
{{UAAURL}}/oauth/token?grant_type=client_credentials
Params
Authorization
Headers (2)
Body
Pre-request Script
Tests
KEY
VALUE
grant_type
client_credentials
Key
Value
Response
Hit the Send button to get
SDSIM
Edit
VARIABLE
INITIAL VALUE
CURRENT VALUE
client_token
client
myClientID
pwd
myClientSecret
UAAURL
https://890407d7-e617-4d70-985f-01792d693387.predix-uaa.run.aws-usw02-pr.ice.predix.io
metadataurl
https://ic-metadata-service.run.aws-usw02-pr.ice.predix.io
eventurl
https://ic-event-service.run.aws-usw02-pr.ice.predix.io
mediaurl
https://ic-media-service.run.aws-usw02-pr.ice.predix.io
traffic_zone_id
SDSIM-IE-TRAFFIC
parking_zone_id
SDSIM-IE-PARKING
ped_zone_id
SDSIM-IE-PEDESTRIAN
Import your Postman Collection and Environment
Provided to you by email as two separate .json attachments
Learn
Build
Browse

# 1 Get Client Token



# Metadata

Get All Assets:

- Get a list of all the assets within a specific eventType or within a certain geographic region

Get Asset Detail

- Get the asset details to one particular asset

Get Asset → Location List associated with that asset

- Get all the locations within view of one asset

Get SubAssets

- Get all the children assets (sensors) assigned to a node asset

Get All Locations

- Get a list of all the locations within a specific geographic region

Get Location Details

- Get the location details to one particular location

Get Location → Asset List associated with that location

- Get all assets that are currently monitoring that location

# CityIQ Metadata Service

The Metadata Service provides users with static data such as assets and physical locations with their properties and identifiers.

**json responses in blue are obtained using the metadata service url**

Sample Responses are described here in json. Please note, data is only available by permissions granted by your municipality.

## assetType NODE

```
{ assetUid: 'NODE-X',
  parentAssetUid: null,
  eventTypes: [ 'HEALTH_REPORT' ],
  mediaType: null,
  assetType: 'NODE',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {} }
```

## assetType CAMERA

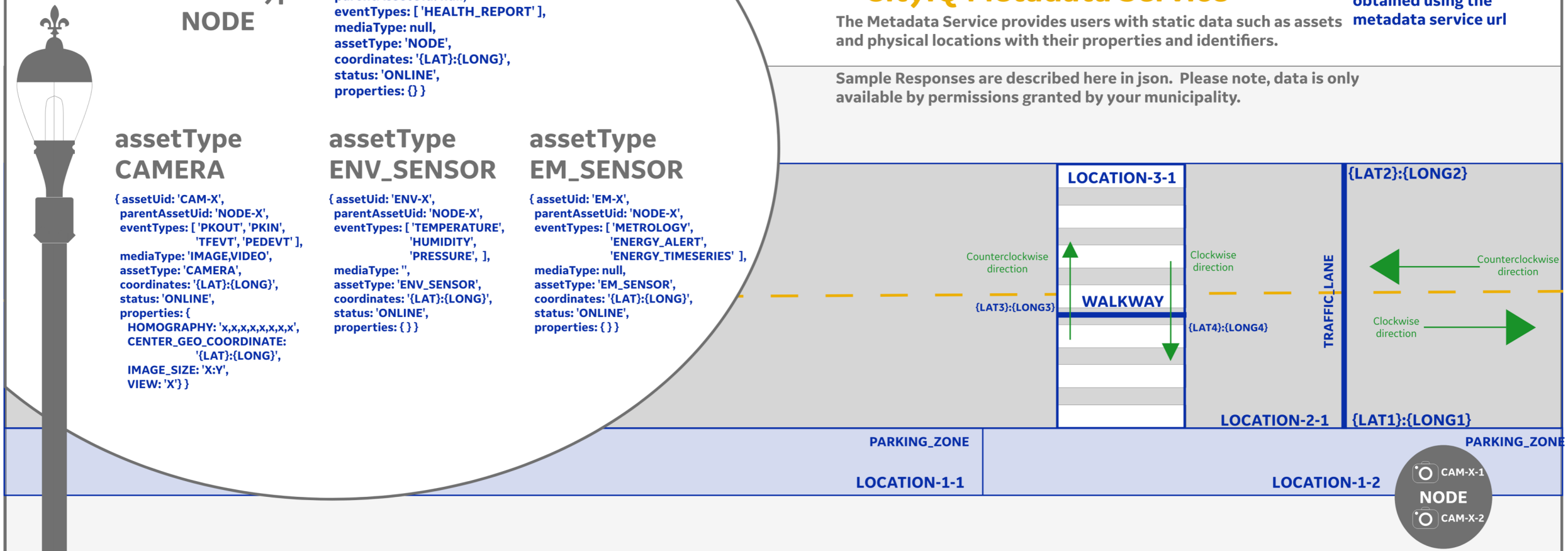
```
{ assetUid: 'CAM-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'PKOUT', 'PKIN',
                'TFEVT', 'PEDEVT' ],
  mediaType: 'IMAGE,VIDEO',
  assetType: 'CAMERA',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {
    HOMOGRAPHY: 'x,x,x,x,x,x,x,x',
    CENTER_GEO_COORDINATE:
                '{LAT}:{LONG}',
    IMAGE_SIZE: 'X:Y',
    VIEW: 'X'} }
```

## assetType ENV_SENSOR

```
{ assetUid: 'ENV-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'TEMPERATURE',
                'HUMIDITY',
                'PRESSURE', ],
  mediaType: '',
  assetType: 'ENV_SENSOR',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: { } }
```

## assetType EM_SENSOR

```
{ assetUid: 'EM-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'METROLOGY',
                'ENERGY_ALERT',
                'ENERGY_TIMESERIES' ],
  mediaType: null,
  assetType: 'EM_SENSOR',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: { } }
```




## locationType PARKING_ZONE

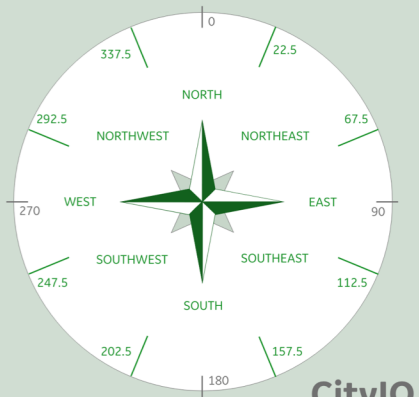
```
{ locationUid: 'LOCATION-1-1',
  locationType: 'PARKING_ZONE',
  parentLocationUid: 'LOCATION-1',
  coordinatesType: 'GEO',
  coordinates: '{LAT}:{LONG},{LAT}:{LONG},
                {LAT}:{LONG},{LAT}:{LONG}',
  name: '1ST-AVE-PARKING-SOUTH',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {} }
```

## locationType TRAFFIC_LANE

```
{ locationUid: 'LOCATION-2-1',
  locationType: 'TRAFFIC_LANE',
  parentLocationUid: 'LOCATION-2',
  coordinatesType: 'GEO',
  coordinates: '{LAT1}:{LONG1},{LAT2}:{LONG2}',
  name: '1ST-AVE',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {
    TFEVT: "TRAFFIC_FLOW"} }
```

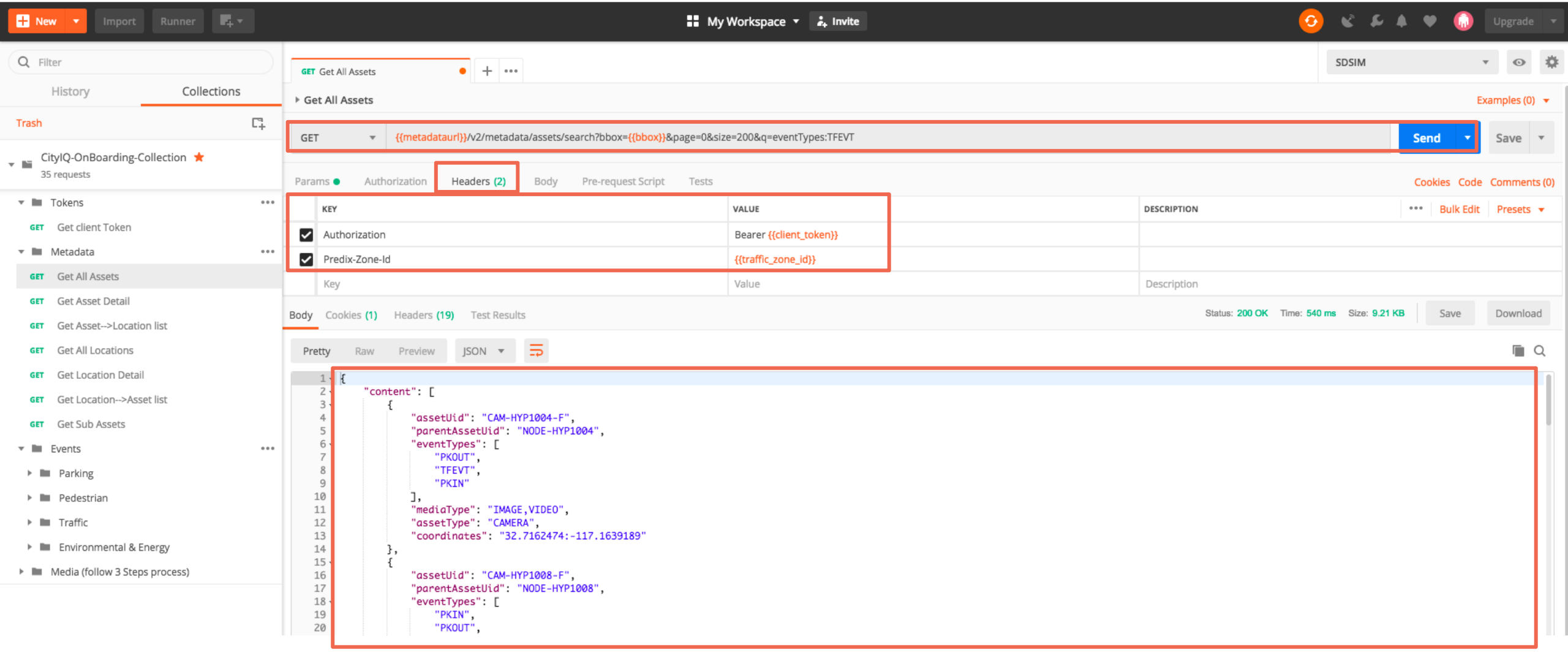
## locationType WALKWAY

```
{ locationUid: 'LOCATION-3-1',
  locationType: 'WALKWAY',
  parentLocationUid: 'LOCATION-3',
  coordinatesType: 'GEO',
  coordinates: '{LAT3}:{LONG3},{LAT4}:{LONG4}',
  name: '1ST-AVE-WALKWAY-SOUTH',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {} }
```



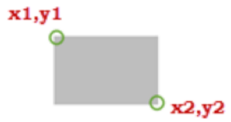

CityIQ API Version 1.0

# 2 Metadata - Get All Assets



# Input Params for Get All Assets

**Query Parameters**

| Parameter | Description | Required? | Filter Values |
|---|---|---|---|
| bbox | The bounded area for your search; establishes the periphery of a searchable area for assets and nodes, identified by GPS coordinates. | Yes | **bbox= <lat1:long1>, <lat2:long2>**<br><br>Replace <lat1:long1>, <lat2:long2> with your actual GPS coordinates, similar to the following:<br><br>**Example**: 32.715675:-117.161230,32.708498:-117.151681<br><br>You can use Google Maps to obtain the latitude and longitude. To locate GPS coordinates for the area and identify the boundaries, place the cursor in the upper left location of the area, then right-click and select **What's here?** to obtain the first set of coordinates. Right-click on the lower-right location and repeat this procedure to obtain the second set of coordinates.<br><br>  |
| q | Identifies a "type" query. | No | Query to search by assetType, mediaType, or eventTypes. |
| assetType | Filter by type of asset.<br><br>**Note:** CAMERA is the only sensor that will generate eventTypes and mediaType. | No | See the enumeration codes for assetType in the Glossary for Intelligent Cities. |
| mediaType | Filter by type of media.<br><br>**Note:** Get Media only works when you have access to the Situational Awareness API. | No | See the enumeration codes for mediaType in the Glossary for Intelligent Cities. |
| eventTypes | Filter by type of event. | No | See the enumeration codes for eventTypes in the Glossary for Intelligent Cities. |
| size | Maximum number of records to return per page; if none specified, the default value of 2 is used automatically. | No | Numerical value, such as 20. |
| page | Indicates the page number; default is 0. | No | Numerical value, such as 1. |

https://ie-cities-docs.run.aws-usw02-pr.ice.predix.io/#r_get_list_of_assets_api.html

# Output Response Parameters for Get All Assets

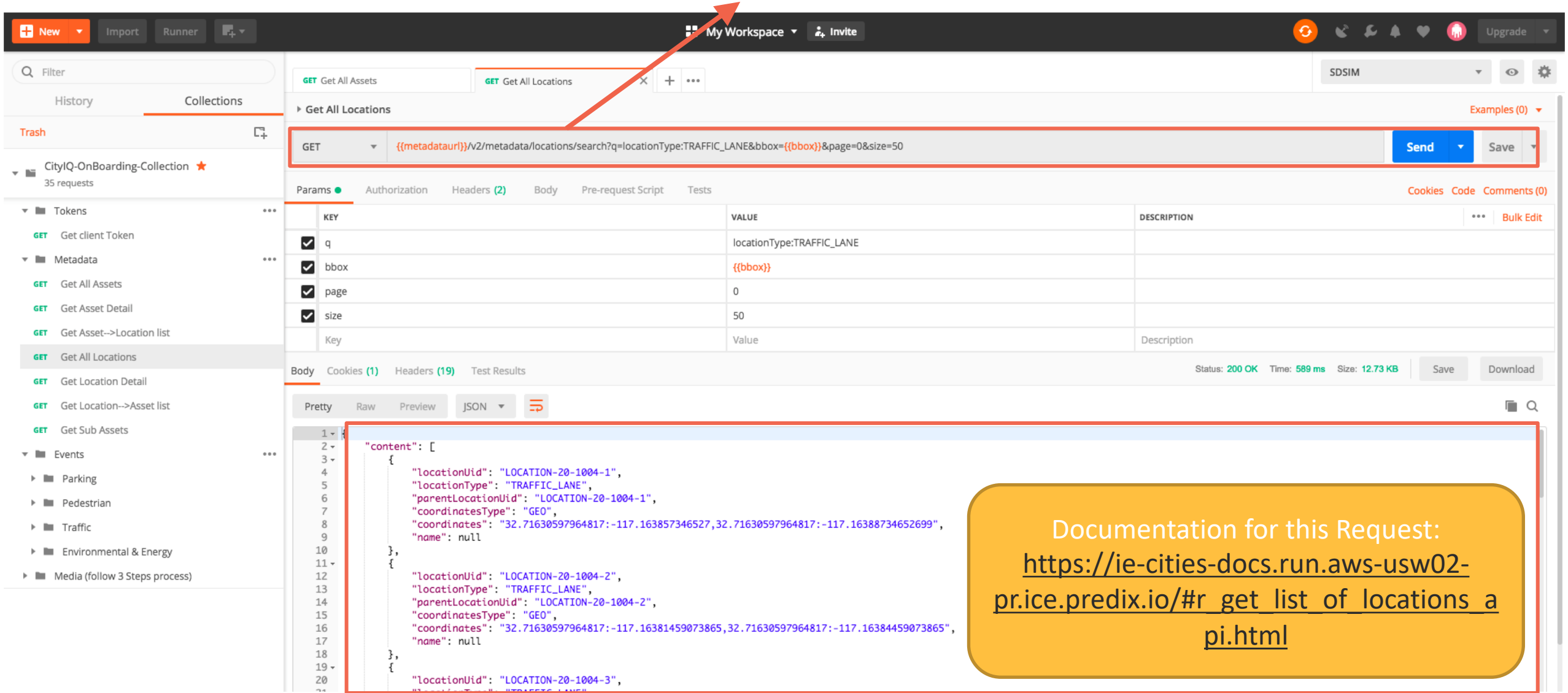
**Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| **assetUid** | String | A unique identifier established by a customer or external resource. For example, CAMERA-STG-HYP1042-CAM-L to identify a camera. |
| **parentAssetUid** | String | A unique identifier assigned to the asset at the top of a hierarchical set of assets, in other words, the parent of a child asset. For example, a node is a parent asset, comprising child assets such as cameras or microphones. |
| **eventTypes** | String | The event type that was recorded. In the sample response data, the camera on the specified node is collecting data on parking instances (vehicle in, vehicle out) and traffic flow in the parking area.<br><br>See the enumeration codes for eventTypes in the Glossary for Intelligent Cities. |
| **assetType** | String | Type of asset that records the events.<br><br>See the enumeration codes for assetType in the Glossary for Intelligent Cities. |
| **mediaType** | String | Media output. In the sample response data, the camera on the specified node is collecting video to record when vehicles enter and exit a parking space.<br><br>See the enumeration codes for mediaType in the Glossary for Intelligent Cities. |
| **coordinates** | String | The GPS coordinates (latitude, longitude) for the referenced asset (identified by assetUid), such as **32.711653,-117.157314** to identify where the camera is installed. |

https://ie-cities-docs.run.aws-usw02-pr.ice.predix.io/#r_get_list_of_assets_api.html

# 2 Metadata - Get All Locations

Get All Locations Query is
{{metadataurl}}/v2/metadata/locations/search?{{inputparams}}




Documentation for this Request:
https://ie-cities-docs.run.aws-usw02-pr.ice.predix.io/#r_get_list_of_locations_api.html

# Getting Started Summary

Always need REST API for UAA URL Queries and Metadata Queries

- UAA provides access_token = client_token necessary for future calls
- Metadata provides access to static data such as assets and locations
- Events are obtained by REST API  - or - WebSocket

# Parking Planning API

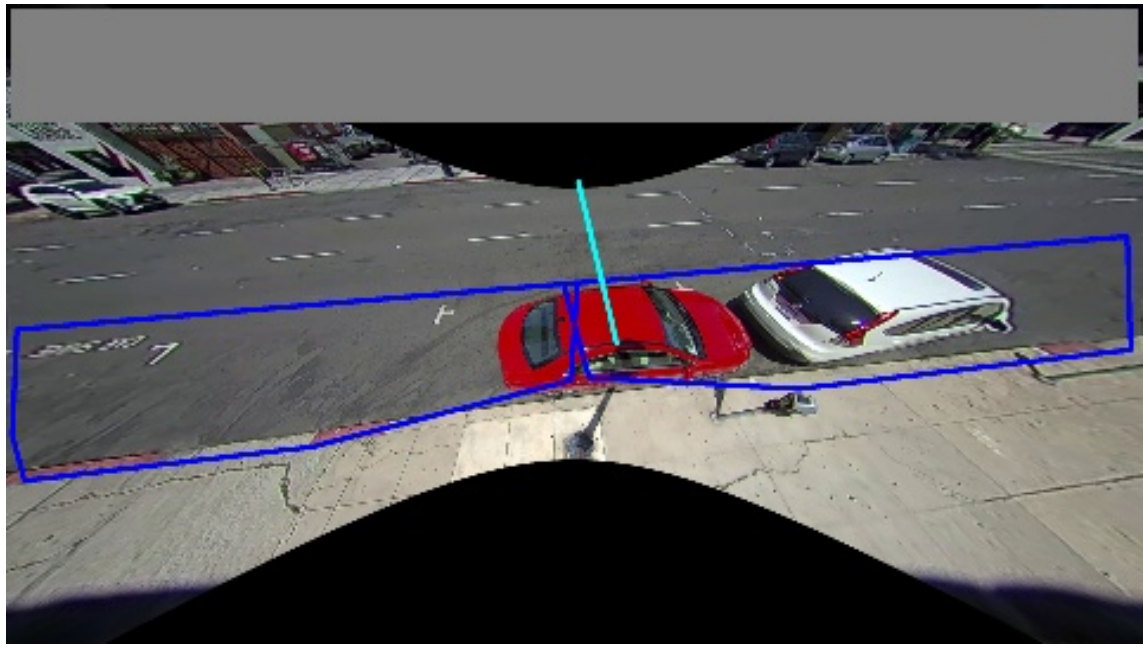
This section clarify the process required to access Parking specific events:

1. What is the Parking Planning API?
2. What data does the Parking Planning API report?
3. What can I do with this data?
4. How do I access the Parking Planning API? And what URLs do I need to access it?

# Parking Planning API

Allows clients access to data related to Parking In (PKIN) and Parking Out (PKOUT) Events

- PKIN: Parking Space Occupied
- PKOUT: Parking Space Vacated
- Returns coordinates of vehicle
- Monitors events occurring within Parking_Zone predetermined by configurator



# CityIQ Parking Planning API

The Parking API uses the camera asset and performs CityIQ Analytics on the node and returns Parking In (PKIN) and Parking Out (PKOUT) events

Sample Responses are described here in json. Please note, the Parking Predix Zone ID is a necessary header input to access this API.

json responses in blue are obtained using the metadata service url

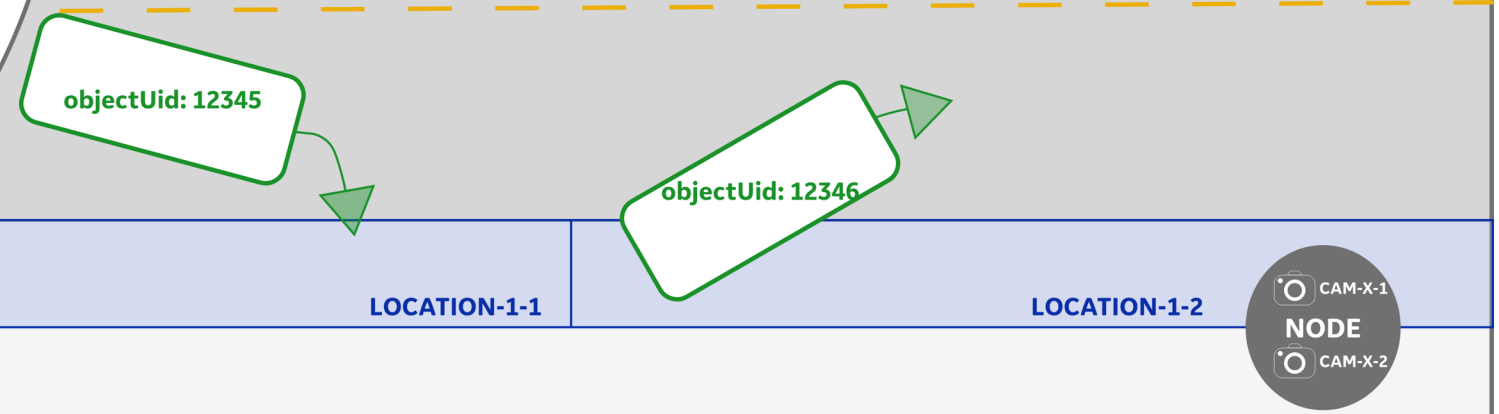
json responses in green are obtained using the events service url

## assetType NODE

```
{ assetUid: 'NODE-X',
  parentAssetUid: null,
  eventTypes: [ 'HEALTH_REPORT' ],
  mediaType: null,
  assetType: 'NODE',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {} }
```

## assetType CAMERA

```
{ assetUid: 'CAM-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'PKOUT', 'PKIN', 'TFEVT', 'PEDEVT' ],
  mediaType: 'IMAGE,VIDEO',
  assetType: 'CAMERA',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {
    HOMOGRAPHY: 'x,x,x,x,x,x,x,x',
    CENTER_GEO_COORDINATE: '{LAT}:{LONG}',
    IMAGE_SIZE: 'X:Y',
    VIEW: 'X'} }
```




## locationType PARKING_ZONE

```
{ locationUid: 'LOCATION-1-1',
  locationType: 'PARKING_ZONE',
  parentLocationUid: 'LOCATION-1',
  coordinatesType: 'GEO',
  coordinates: '{LAT}:{LONG},{LAT}:{LONG},{LAT}:{LONG},{LAT}:{LONG}',
  name: '1ST-AVE-PARKING-SOUTH',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {} }
```

## eventType PKIN

```
{ locationUid: 'LOCATION-1-1',
  assetUid: 'CAM-X',
  eventType: 'PKIN',
  timestamp: 1539090000000,
  properties: {
    orgPixelCoordinates: 'X:Y,X:Y,X:Y,X:Y',
    pixelCoordinates: 'A:B,A:B,A:B,A:B',
    objectUid: '12345',
    geoCoordinates: '{LAT}:{LONG},{LAT}:{LONG}',
    imageAssetUid: 'CAM-X' },
  measures: {} }
```

## eventType PKOUT

```
{ locationUid: 'LOCATION-1-2',
  assetUid: 'CAM-X',
  eventType: 'PKOUT',
  timestamp: 1539090011000,
  properties: {
    orgPixelCoordinates: 'X:Y,X:Y,X:Y,X:Y',
    pixelCoordinates: 'A:B,A:B,A:B,A:B',
    objectUid: '12346',
    geoCoordinates: '{LAT}:{LONG},{LAT}:{LONG}',
    imageAssetUid: 'CAM-X' },
  measures: {} }
```

Parking Planning Version 1.0

# Accessing the Parking Planning API

## REST API

- Requires: Event Service URL
- Historical Events
- One-time request for events specified within a time frame
- Requires use of Unix Epoch https://currentmillis.com/

## WebSocket

- Requires: WebSocket URL
- Near-real-time Events
- Open connection that updates client when events occur with the event details

Regardless of Method Chosen, UAA and Metadata need to be accessed via REST API

# How to Get Parking Planning Events (REST API)
# 3 Step Process

1. UAA URL – Get Client Token (Skip if token already obtained)
2. Metadata Service URL – Find the Asset or Location you would like event data for
    - Example: Get all parking locations on E street between 7th and 8th Ave
    - Q=locationType:PARKING_ZONE
    - Bbox= 32.714858:-117.158431,32.714562:-117.157248
3. Event Service URL – Ask the API for the specific events related to that Asset or Location
    - Example: Get all the events at locationUid (found in step 2) and return all PKIN events between 9am PST and 10am PST
    - locationUid: dvi25bk0tcfjk01npf9
    - eventType=PKIN
    - startTime=1548867600000
    - endTime= 1548871200000

current powered by GE

Note: Each of these query steps require necessary Authorization and Header Inputs

# Sample Response


```
{
   "content": [
      {
         "locationUid": "dvi25bk0tcfjk01npf9",
         "assetUid": "daab92c2-ed35-4171-85fb-feb1de55efde",
         "eventType": "PKIN",
         "timestamp": 1548082737314,
         "properties": {
            "orgPixelCoordinates": "267.0:96.0,267.0:139.0,331.0:139.0,331.0:96.0",
            "pixelCoordinates": "174.0:37.0,157.0:75.0,190.0:49.0,208.0:9.0",
            "objectUid": "1548063154",
            "geoCoordinates": "32.71464384735591:-117.15782474409559,
                               32.71465410670605:-117.1577391696346,
                               32.71468110671108:-117.15774374222988,
                               32.71467084736094:-117.15782931669035",
            "imageAssetUid": "ac8c83dc-818a-4be2-98b3-aa178223e7b5"
         },
         "measures": {}
      }, ...
```


- **Location Name**
- **Camera observing Event**
- **Event that occurred**
- **When it happened.**

- Camera Specific Details
- Camera Specific Details
- **Event Identifier**
- Geographic Coordinates of Vehicle Parked
- Image Reference Number (for Media API Clients only)

current powered by GE

# Traffic Planning API

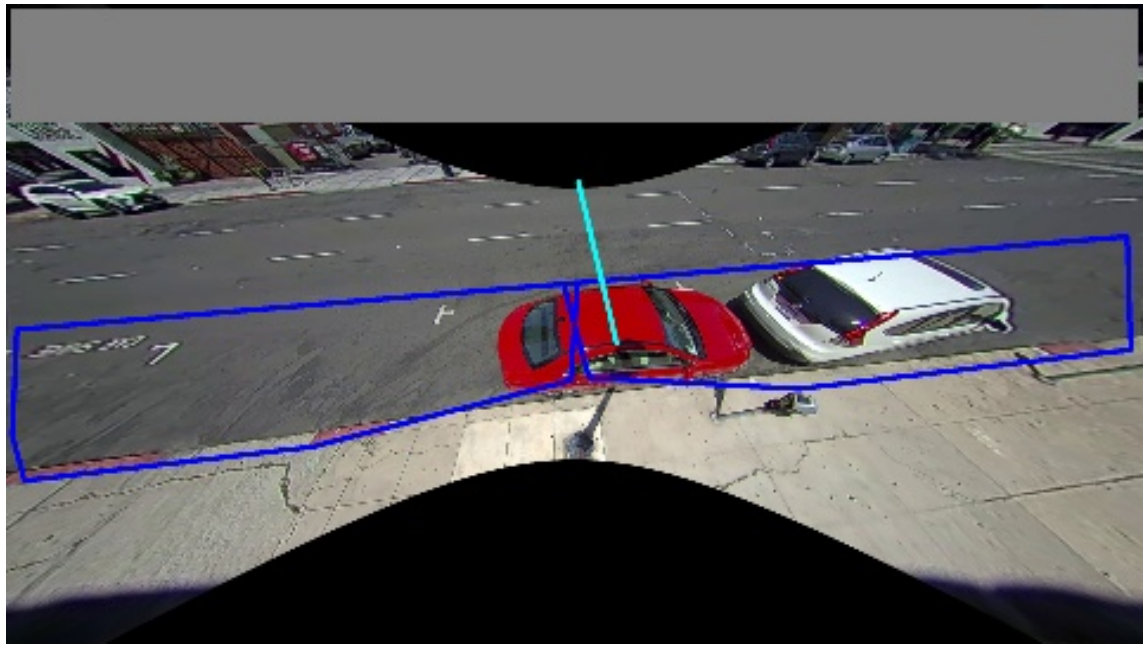
This section clarify the process required to access Traffic specific events:

1. What is the Traffic Planning API?
2. What data does the Traffic Planning API report?
3. What can I do with this data?
4. How do I access the Traffic Planning API? And what URLs do I need to access it?

# Traffic Planning API

Allows clients access to data related to Traffic Group movements as individual events

- TFEVT: Summary of vehicles crossing the trip wire with direction, velocity and count.
  - Traffic 2.0 will report bicycles, small vehicles, large vehicles, etc.
- Location is indicated with a trip wire (two geographic coordinates)



# CityIQ Traffic Planning API

The Traffic API uses the camera asset and performs CityIQ Analytics on the node to return Traffic Events (TFEVT) which register vehicles passing the tripwire in blue.

Sample Responses are described here in json. Please note, the Traffic Predix Zone ID is a necessary header input to access this API.

**json responses in blue are obtained using the metadata service url**

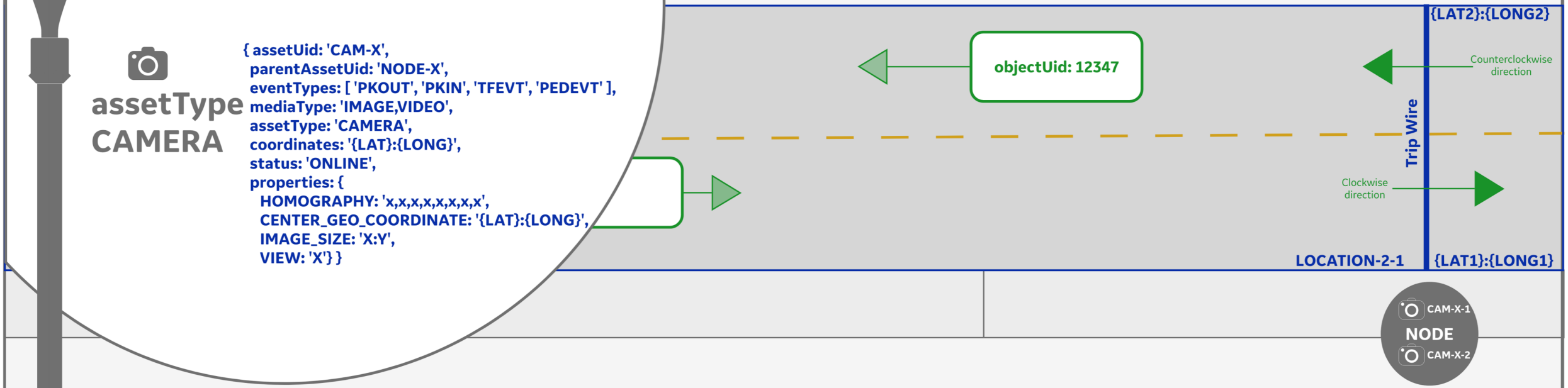
**json responses in green are obtained using the events service url**

## assetType NODE

```
{ assetUid: 'NODE-X',
  parentAssetUid: null,
  eventTypes: [ 'HEALTH_REPORT' ],
  mediaType: null,
  assetType: 'NODE',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {} }
```

## assetType CAMERA

```
{ assetUid: 'CAM-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'PKOUT', 'PKIN', 'TFEVT', 'PEDEVT' ],
  mediaType: 'IMAGE,VIDEO',
  assetType: 'CAMERA',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {
    HOMOGRAPHY: 'x,x,x,x,x,x,x,x,x',
    CENTER_GEO_COORDINATE: '{LAT}:{LONG}',
    IMAGE_SIZE: 'X:Y',
    VIEW: 'X'} }
```





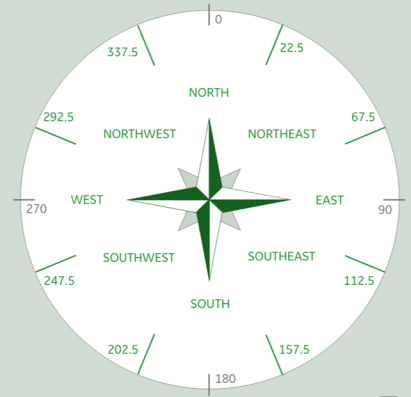


## locationType TRAFFIC_LANE

```
{ locationUid: 'LOCATION-2-1',
  locationType: 'TRAFFIC_LANE',
  parentLocationUid: 'LOCATION-2',
  coordinatesType: 'GEO',
  coordinates: '{LAT1}:{LONG1},{LAT2}:{LONG2}',
  name: '1ST-AVE',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {
    TFEVT: "TRAFFIC_FLOW"} }
```

## eventType TFEVT

```
{ locationUid: 'LOCATION-2-1',
  assetUid: 'CAM-X',
  eventType: 'TFEVT',
  timestamp: 1539090011000,
  properties: {
    speedUnit: 'METERS_PER_SEC',
    eventUid: '12347',
    directionUnit: 'DEGREE',
    counter_direction_vehicleType: 'vehicle',
    vehicleType: '' },
  measures: {
    counter_direction: "270",
    counter_direction_speed : "20",
    counter_direction_vehicleCount: '1',
    diretion: "90",
    speed: "0",
    vehicleCount: "1" } }
```




Traffic Planning Version 1.0

# Accessing the Traffic Planning API

## REST API

- Requires: Event Service URL
- Historical Events
- One-time request for events specified within a time frame
- Requires use of Unix Epoch https://currentmillis.com/

## WebSocket

- Requires: WebSocket URL
- Near-real-time Events
- Open connection that updates client when events occur with the event details

Regardless of Method Chosen, UAA and Metadata need to be accessed via REST API

# How to Get Traffic Planning Events (REST API)
# 3 Step Process

1. UAA URL – Get Client Token (Skip if token already obtained)
2. Metadata Service URL – Find the Asset or Location you would like event data for
    - Example: Get all locations that are TRAFFIC_LANE(s) on E street between 7th and 8th Ave
    - Q=locationType:TRAFFIC_LANE
    - Bbox= 32.714858:-117.158431,32.714562:-117.157248
3. Event Service URL – Ask the API for the specific events related to that Asset or Location
    - Example: Get all the TFEVTs that occur at location 25c2e11b (found in step 2) and return all TFEVTs events between 9am PST and 10am PST
    - assetUid:25c2e11b
    - eventType=TFEVT
    - startTime=1548867600000
    - endTime= 1548871200000

current powered by GE

Note: Each of these query steps require necessary Authorization and Header Inputs

# Sample Response


```
{"content": [
    {
      "locationUid": "25c2e11b",
      "assetUid": "6c0a042a-bd4d-4b33-90fa-ab09022c078c",
      "eventType": "TFEVT",
      "timestamp": 1548048753023,
      "properties": {
        "speedUnit": "METERS_PER_SEC",
        "eventUid": "NmMwYTA0MmEtYmQ0ZC00YjMzLTkwZmEtYWIwOTAyMmMwNzhjLDE1NDgwNDg3NTMwMjM=",
        "directionUnit": "DEGREE",
        "counter_direction_vehicleType": "vehicle",
        "vehicleType": "vehicle"
      },
      "measures": {
        "counter_direction_speed": 0,
        "vehicleCount": 3,
        "counter_direction_vehicleCount": 0,
        "counter_direction": 183,
        "speed": 18.399834,
        "direction": 3
      }
    },
```


- **Location Name**
- **Camera observing Event**
- **Event that occurred**
- **When it happened.**

- Measurement Properties
- **Event Identifier**

- **Event Measurementes for both directions**

current
powered by GE

# Pedestrian Planning API

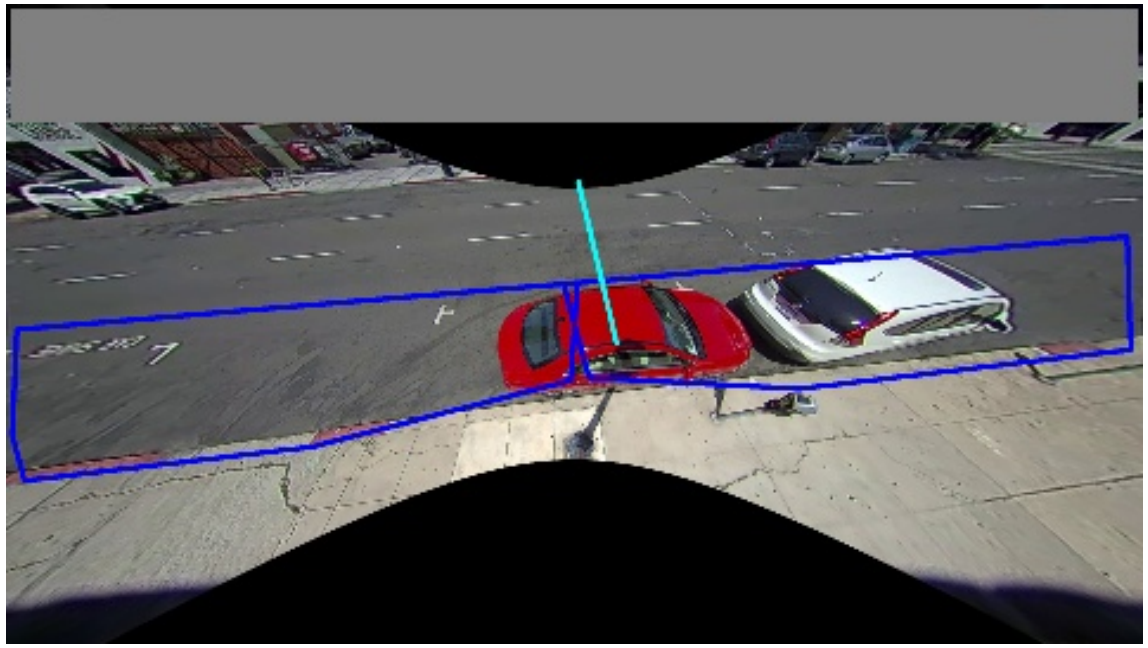
This section clarify the process required to access Pedestrian specific events:

1. What is the Pedestrian Planning API?
2. What data does the Pedestrian Planning API report?
3. What can I do with this data?
4. How do I access the Pedestrian Planning API? And what URLs do I need to access it?

# Pedestrian Planning API

Allows clients access to data related to Pedestrian Group movements as individual events

- PEDEVT: Summary of pedestrians crossing the trip wire with direction, velocity and count.
- Location is indicated with a trip wire (two geographic coordinates)



current
powered by GE

# CityIQ Pedestrian Planning API

The Pedestrian API uses the camera asset and performs CityIQ Analytics on the node to return Pedestrian Events (PEDEVT)

Sample Responses are described here in json. Please note, the Pedestrian Predix Zone ID is a necessary header input to access this API.

**json responses in blue are obtained using the metadata service url**

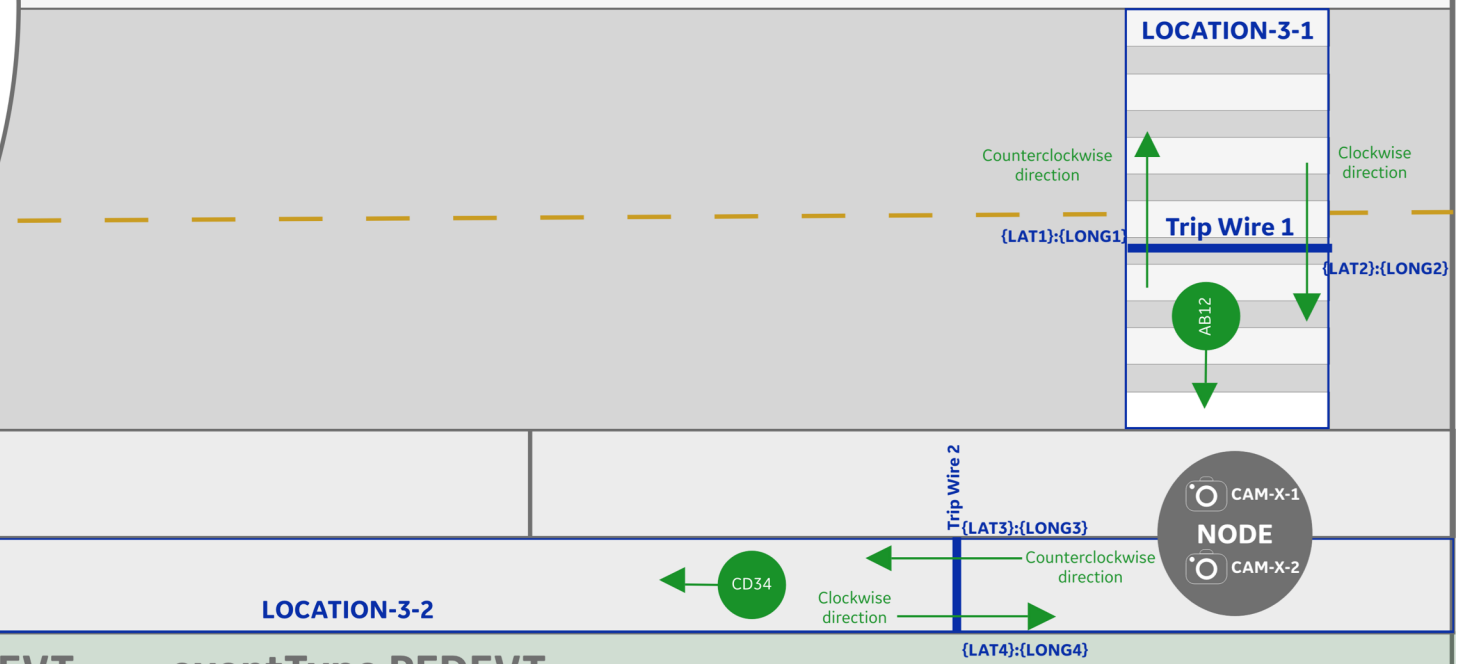
**json responses in green are obtained using the events service url**

## assetType NODE

```
{ assetUid: 'NODE-X',
  parentAssetUid: null,
  eventTypes: [ 'HEALTH_REPORT' ],
  mediaType: null,
  assetType: 'NODE',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {} }
```

## assetType CAMERA

```
{ assetUid: 'CAM-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'PKOUT', 'PKIN', 'TFEVT', 'PEDEVT' ],
  mediaType: 'IMAGE,VIDEO',
  assetType: 'CAMERA',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {
    HOMOGRAPHY: 'x,x,x,x,x,x,x,x,x',
    CENTER_GEO_COORDINATE: '{LAT}:{LONG}',
    IMAGE_SIZE: 'X:Y',
    VIEW: 'X'} }
```




## locationType WALKWAY

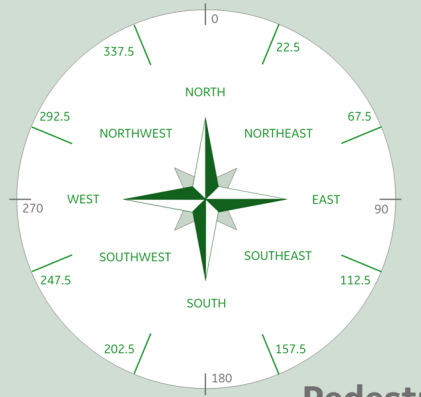
```
{ locationUid: 'LOCATION-3-1',
  locationType: 'WALKWAY',
  parentLocationUid: 'LOCATION-3',
  coordinatesType: 'GEO',
  coordinates: '{LAT1}:{LONG1},{LAT2}:{LONG2}',
  name: '1ST-AVE-WALKWAY-SOUTH',
  city: 'SPRINGFIELD',
  state: 'ILLINOIS',
  country: 'USA',
  zipcode: '12345',
  timezone: 'CDT',
  properties: {
    CD: 'X',
    CCD: 'Y'},
  address: null,
  analyticCategory: {} }
```

## eventType PEDEVT

```
{ locationUid: 'LOCATION-3-1',
  assetUid: 'CAM-X-1',
  eventType: 'PEDEVT',
  timestamp: 1539090011000,
  properties: {
    directionUnit: 'DEGREE',
    speedUnit: 'METERS_PER_SEC',
    eventUid: 'ABC123' },
  measures: {
    counter_direction_speed: '0',
    counter_direction_pedestrianCount: '0',
    pedestrianCount: '1',
    counter_direction: '180',
    speed: '0.6',
    direction: '180' } }
```

## eventType PEDEVT

```
{ locationUid: 'LOCATION-3-2',
  assetUid: 'CAM-X-2',
  eventType: 'PEDEVT',
  timestamp: 1539090011000,
  properties: {
    directionUnit: 'DEGREE',
    speedUnit: 'METERS_PER_SEC',
    eventUid: 'CD34' },
  measures: {
    counter_direction_speed: '0.5',
    counter_direction_pedestrianCount: '1',
    pedestrianCount: '1',
    counter_direction: '270',
    speed: '0',
    direction: '90' } }
```




Pedestrian Planning Version 1.0

# Accessing the Pedestrian Planning API

## REST API

- Requires: Event Service URL
- Historical Events
- One-time request for events specified within a time frame
- Requires use of Unix Epoch https://currentmillis.com/

## WebSocket

- Requires: WebSocket URL
- Near-real-time Events
- Open connection that updates client when events occur with the event details

Regardless of Method Chosen, UAA and Metadata need to be accessed via REST API

# How to Get Pedestrian Planning Events (REST API)
# 3 Step Process

1. UAA URL – Get Client Token (Skip if token already obtained)
2. Metadata Service URL – Find the Asset or Location you would like event data for
    - Example: Get all assets that register pedestrian events on E street between 7th and 8th Ave
    - Q=eventTypes:PEDEVT
    - Bbox= 32.714858:-117.158431,32.714562:-117.157248
3. Event Service URL – Ask the API for the specific events related to that Asset or Location
    - Example: Get all the PEDEVTs within the view of camera 01a6769a-ccb0-41ee-af4c-b48d7265b03f (found in step 2) and return all PEDEVTs events between 9am PST and 10am PST
    - assetUid:01a6769a-ccb0-41ee-af4c-b48d7265b03f
    - eventType=PEDEVT
    - startTime=1548867600000
    - endTime= 1548871200000

current powered by GE

Note: Each of these query steps require necessary Authorization and Header Inputs

# Sample Response


```
{"content": [
    {
      "locationUid": "3f5e9490",
      "assetUid": "01a6769a-ccb0-41ee-af4c-b48d7265b03f",
      "eventType": "PEDEVT",
      "timestamp": 1548045500020,
      "properties": {
        "directionUnit": "DEGREE",
        "speedUnit": "METERS_PER_SEC",
        "eventUid": "MDFhNjc2OWEtY2NiMC00MWVlLWFmNGMtYjQ4ZDcyNjViMDNmLDE1NDgwNDU1MDAwMjA="
      },
      "measures": {
        "counter_direction_speed": 0,
        "counter_direction_pedestrianCount": 0,
        "pedestrianCount": 1,
        "counter_direction": 270,
        "speed": 2.641,
        "direction": 90
      }
    },...
]}
```


- **Location Name**
- **Camera observing Event**
- **Event that occurred**
- **When it happened.**

- Measurement Properties
- Measurement Properties
- **Event Identifier**

- **Event Measurementes for both directions**

current
powered by GE

# Environmental Planning API

Allows clients access to data reporting Temperature, Humidity and Pressure

- TEMPERATURE:
- HUMIDITY:
- PRESSURE:
- Returns sensor measurements

Must Use Asset (Node and Sensor) Metadata to obtain Environmental Events

# CityIQ Environmental Planning API

The Environmental API uses the sensor assets and reports measurements periodically in the form of Temperature, Humidity, and Pressure events.

Sample Responses are described here in json. Please note, the Environmental Predix Zone ID is a necessary header input to access this API.

json responses in blue are obtained using the metadata service url

json responses in green are obtained using the events service url

## assetType NODE

```
{ assetUid: 'NODE-X',
  parentAssetUid: null,
  eventTypes: [ 'HEALTH_REPORT' ],
  mediaType: null,
  assetType: 'NODE',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: {} }
```

## assetType ENV_SENSOR

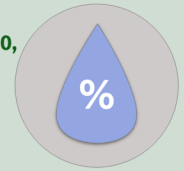
```
{ assetUid: 'ENV-X',
  parentAssetUid: 'NODE-X',
  eventTypes: [ 'TEMPERATURE',
                'HUMIDITY',
                'PRESSURE', ],
  mediaType: '',
  assetType: 'ENV_SENSOR',
  coordinates: '{LAT}:{LONG}',
  status: 'ONLINE',
  properties: { } }
```

## eventType TEMPERATURE

```
{ locationUid: 'ENV_NODE_LOCATION',
  assetUid: 'ENV-X',
  eventType: 'TEMPERATURE',
  timestamp: 1539090011000,
  properties: {
    unit: 'KELVIN',
    powerOf10: '-1' },
 measures: {
   min: '2880',
   median: '2890'
   max: '2930',
   mean: '2900' } }
```

## eventType HUMIDITY

```
{ locationUid: 'ENV_NODE_LOCATION',
  assetUid: 'ENV-X',
  eventType: 'HUMIDITY',
  timestamp: 1539090011000,
  properties: {
    unit: 'PERCENT',
    powerOf10: '-2' },
 measures: {
   min: '5416',
   median: '5420'
   max: '5421',
   mean: '5419' } }
```




## eventType PRESSURE

```
{ locationUid: 'ENV_NODE_LOCATION',
  assetUid: 'ENV-X',
  eventType: 'PRESSURE',
  timestamp: 1539090011000,
  properties: {
    unit: 'PASCALS',
    powerOf10: '0' },
 measures: {
   min: '100711',
   median: '100717'
   max: '100722',
   mean: '100717' } }
```

**Sample Calculation for Temperature:**
**Temp in Celsius = 2880 K * $10^{-1}$ - 273 = 15°C**

# Accessing the Environmental Planning API

## REST API

- Requires: Event Service URL
- Historical Events
- One-time request for events specified within a time frame
- Requires use of Unix Epoch https://currentmillis.com/

## WebSocket

- Requires: WebSocket URL
- Near-real-time Events
- Open connection that updates client when events occur with the event details

Regardless of Method Chosen, UAA and Metadata need to be accessed via REST API

# How to Get Environmental Planning Events (REST API)
# 3 Step Process

1. UAA URL – Get Client Token (Skip if token already obtained)
2. Metadata Service URL – Find the Asset you would like event data for
    - Example: Get all parking locations on E street between 7th and 8th Ave
    - Q=eventTypes:TEMPERATURE
    - Bbox= 32.714858:-117.158431,32.714562:-117.157248
3. Event Service URL – Ask the API for the specific events related to that Asset
    - Example: Get all the events at assetUid 1ec8a25b-fb17-4242-ab6c-1aeed5c32e59 (found in step 2) and return all TEMPERATURE events between 9am PST and 10am PST
    - assetUid: 1ec8a25b-fb17-4242-ab6c-1aeed5c32e59
    - eventType=TEMPERATURE
    - startTime=1548867600000
    - endTime= 1548871200000

current
powered by GE

Note: Each of these query steps require necessary Authorization and Header Inputs

# Sample Response


```
{
  "content": [
    {
      "locationUid": "ENVIRONMENTAL_NODE_LOCATION",
      "assetUid": "1ec8a25b-fb17-4242-ab6c-1aeed5c32e59",
      "eventType": "TEMPERATURE",
      "timestamp": 1548050520023,
      "properties": {
        "unit": "KELVIN",
        "powerOf10": "-1"
      },
      "measures": {
        "min": 2896,
        "median": 2896,
        "max": 2896,
        "mean": 2896
      }
    },
```


- **Location Name**
- **Camera observing Event**
- **Event that occurred**
- **When it happened.**

- Measurement unit
- Measurement multiplier

- Measurements that need to be adjusted for local units

current
powered by GE

# Resources

API Documentation: https://ie-cities-docs.run.aws-usw02-pr.ice.predix.io/#c_overview_of_intelligent_cities_apis.html

Developer Portal: https://developer.currentbyge.com/cityiq

Email: support.cityiq@ge.com

CityIQ Starter Code: https://github.com/jethompson911/CityIQ-Starter-Code

# Sample Query Formats

| | Sample Query |
|---|---|
| UAA Service | **[UAA_URL]**/oauth/token?grant_credentials<br><br>Authorization : Basic = btoa(**[client_id]**:**[client_secret]**) |
| Metadata Service | **[metadata_url]**/v2/metadata/assets/**[known_asset]**<br>Or<br>**[metadata_url]**/v2/metadata/locations/**[known_location]** |
| Event Service | **[events_url]**/v2/assets/**[known_asset]**/events?**[search_parameters]**<br>OR<br>**[events_url]**/v2/locations/**[known_location]**/events**?[search_parameters]** |
| Media Service | **[media_url]**/v2/mediastore/ondemand/assets/**[camera_asset]**/media?**[search_parameters]** |
| WebSocket | **[websocket_url]**/v2/events |
| Predix-Zone-ID | Input to Headers<br>Predix-Zone-Id:**[parking_zone_id]** |