

Cisco Spaces API Guide



Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706 USA
<https://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)



Table of Contents

Getting Started.....	3
Cisco Spaces Architecture.....	4
APIs in Cisco Spaces	6
Cisco Spaces Firehose API.....	6
Key Concepts.....	8
Benefits	10
Events supported by Firehose API.....	11
Cisco Spaces REST APIs	19
Detect and Locate REST API.....	19
Asset Locator REST API.....	20
Cisco Spaces Connector APIs	22
Firehose supporting REST APIs	30
Create your first application and view the Firehose API data.....	31
Best Practices for using Firehose API.....	43
Troubleshooting and Frequently Asked Questions	44
Example Script to use Firehose API.....	47
References	47



Getting Started

Cisco Spaces is the next generation outcomes cloud from Cisco, which builds on top of years of industry leading indoor location solutions and adds in multiple new components to deliver an end-to-end solution to solve business use cases through a cloud and subscription-based delivery model. It provides a single point of entry for all location technology and intelligence through a single dashboard interface. Cisco Spaces delivers the industry's most scalable location-based services platform, while being compatible across existing Cisco Aironet® wireless LAN Controllers, Cisco Catalyst® 9800 series wireless LAN Controllers, select Cisco Catalyst® 9000 series of switches, Cisco Meraki® infrastructure (including MV Cameras), as well as select Cisco Collaboration endpoints and support for a wide range of deployment options. In addition, Cisco Spaces can consume and control the IoT radios on select Wireless Access Points to deliver new IoT use cases and enable true Smart Building Solutions. For more details on Cisco Smart Building solutions.

In addition to consuming data from the network infrastructure, Cisco Spaces processes, filters and cleanses the data, provides toolkits to act on this data and makes this data accessible to partners - Independent Software vendors, enterprise software as well as solution partners for delivering business outcomes. There is also an entire ecosystem of IoT devices which can be controlled by IoT gateways which are installed on Wireless Access Points and Catalyst switches via the Cisco Spaces dashboard. Cisco Spaces also enables customers to send data to multiple software partners which are available in the Cisco Spaces Partner App Center to integrate the Cisco solution with a different third-party solution which can be focused on verticals like Healthcare, Manufacturing, Retail, Higher Education or could deliver dedicated use cases like High Value Asset Tracking, Patient and Staff Safety, Employee Experience and Customer Management, Space Utilization, Environmental Monitoring and Compliance etc. By integrating with the Cisco Spaces solution, the possibilities are truly endless. We are excited to have you as a developer look at understanding how to do just that and cannot wait to see what new solutions you would create by leveraging the Cisco Spaces platform.



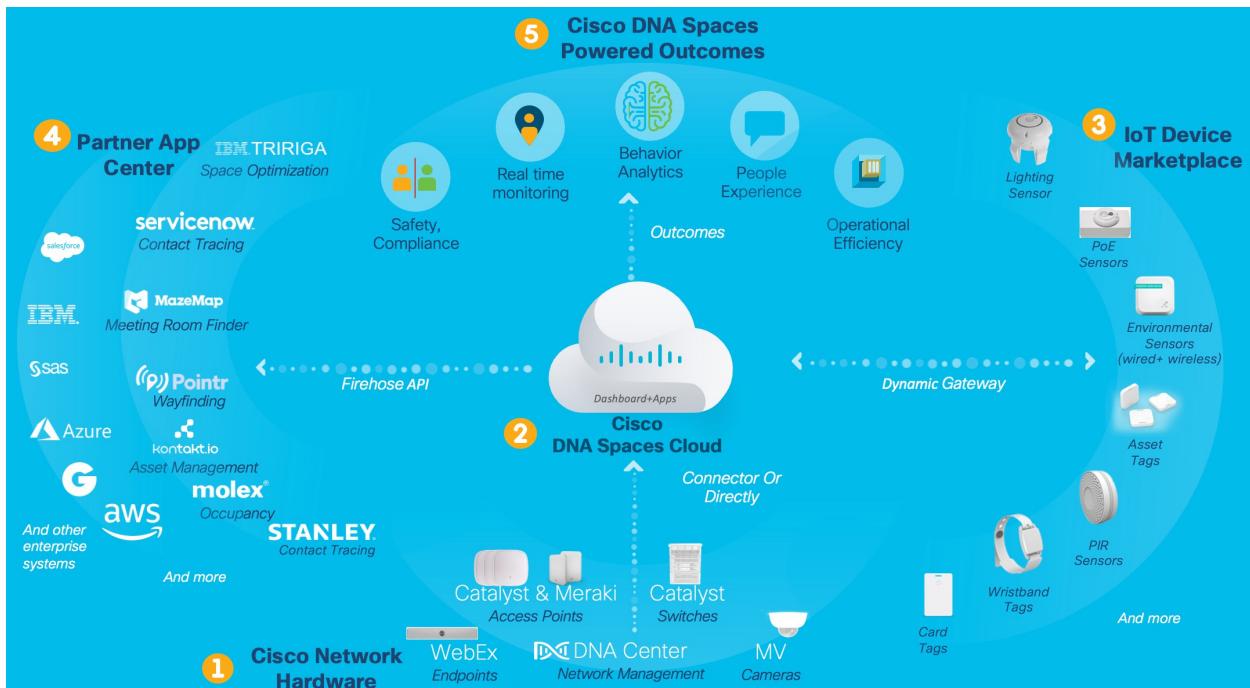
In this guide, we will be going over the high-level architecture to understand how data is coming into the Cisco Spaces cloud from different components, the types of APIs available in the Cisco Spaces solution, API explanations and sample workflows to use for creating your own solutions and applications.

Cisco Spaces Architecture

Before going into the specifics of different API types and models, it is important for application developers to understand basic architecture for Cisco Spaces and understand the different components involved.

At a high level, the data flows in from different network components (Cisco wireless LAN Controllers and Access Points, Catalyst switches, Webex devices, Meraki networks) which generally reside on-premises but not mandatory. There are multiple options to get the data from these devices into the Cisco Spaces cloud, which we will cover further in this section. Once the data is received in the Cisco Spaces cloud, it will be processed by different sub-components which collect the data, understand locations like buildings and floors, calculate a X/Y location, process telemetry etc. and is then consumed by different applications within the Cisco Spaces dashboard. In addition, IoT data can also come directly from IoT Gateways residing on specific Access Points and switches. That solution is called the IoT Services solution and you are encouraged to view the specific documentation to understand how to configure and enable

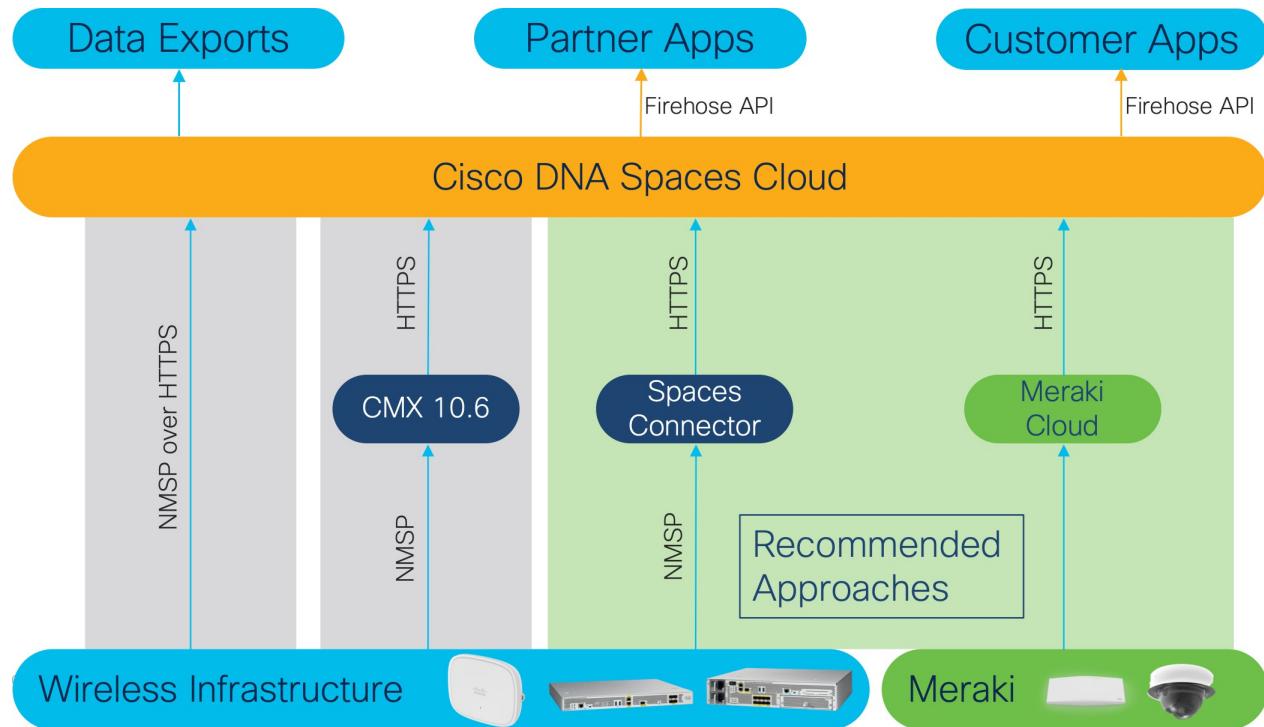
that solution to provide the BLE (Bluetooth Low Energy) data to Cisco Spaces. Finally, the data is sent to different partner applications which are available in the Partner App Center. An end user of the account would choose which application is enabled, what data is sent and from which locations the data is sent. This is a key part of the API enablement of the Cisco Spaces solution and would be covered in more details in upcoming sections.



As mentioned previously, there are multiple ways to connect network devices like wireless LAN controllers to the Cisco Spaces Cloud. Those options are:

1. Via the Cisco Spaces Connector (Recommended and required for most use cases). Unless there is some major reason to not be able to deploy the Cisco Spaces Connector, all customers should follow this method. It provides the most scale and the most features. Many of the more advanced features of Cisco Spaces requires this approach to be used. More details about the Cisco Spaces connector will be covered in its own dedicated section.
2. Via the WLC (Wireless LAN Controller) Directly.
3. Via a CMX by tethering the CMX with Cisco Spaces cloud. In this approach, all the location calculations are done locally on CMX, and the final locations are sent to the cloud.

For more details about these different approaches and the pros and cons of each approach, please see the appropriate documentation. From an API standpoint, we will be focusing on option 1, as it allows for all the possible use cases to look at.



APIs in Cisco Spaces

Cisco Spaces supports two types of APIs. A traditional REST based set of APIs which can be used for some applications, and which support specific use cases within the solution. And a new, state of the art, cloud based streaming API called the Cisco Spaces Firehose API. For most of the applications, customers are encouraged to use the Firehose API. This document covers both types of APIs in the next section starting with the Firehose API.

Cisco Spaces Firehose API

With the need to address the ever-changing data, continuous streaming data is required to update partner applications. Like a Firehose in the real world, the Firehose API was intended to provide a steady stream of all the available data from a real-time source. It can constantly deliver data to many subscribers or consumers, all at the same time. The stream is constant,

delivering new and updated data in real time. The amount of data in the Firehose may vary with spikes and lows, but the data continuously flows through the Firehose until it is processed. After the data is processed, it can be visualized, published, and graphed to suit your business needs, and all this happens in real-time.

To achieve this, Cisco Spaces has introduced a new API model to consume data from the platform called the Firehose API model. It is a cloud-first, highly scalable and event based streaming API which changes the paradigm of leveraging APIs from a pull-based REST model, where an endpoint would continuously have to query a system (like Cisco Mobility Services Engine (MSE) or Cisco Connected Mobile Experiences (CMX) platforms) to a more streamlined push-based model. Developers need to understand that this is a brand-new type of API structure and older CMX or MSE APIs will not be able to be translated 1:1 to the new API model. It is a new type of consumption model, which would first need to be learnt and customers and developers would need to write new code or edit their applications to leverage this API. It is also expected that developers would receive the data from Cisco Spaces but would locally have to store and process the data as needed by their end goals so they may need to create a mechanism to do so. Some sample applications and a handler for MongoDB are made available in the references section of this guide.

Firehose API

Cloud-first, high performance, and low latency Firehose API

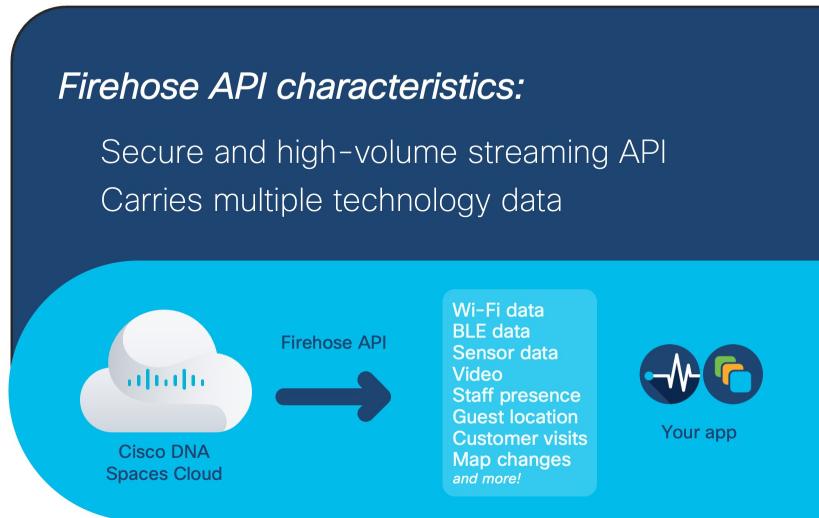
used specifically for developers to integrate their business applications with Cisco DNA Spaces.

Offers several benefits over the traditional approach.

Firehose API characteristics:

Secure and high-volume streaming API

Carries multiple technology data





Key Concepts

Some of the key concepts one needs to understand as they start to look at developing around the Firehose API are:

'Partner' and Cisco Spaces Partner Dashboard

In the context of Cisco Spaces, a 'Partner' can be any person or company which is looking creating a solution by leveraging Cisco Spaces either for their own use or as a provider for others. So independent software vendors (ISVs) as well as Cisco's end customers can both be considered as a 'Partner', but in most situations, a dedicated partner would be a third-party software vendor that is working with Cisco to jointly provide and market a solution for Cisco Spaces customers.

Cisco Spaces also provides customers with an entire Partner App Center where solutions from various third-party providers are made available for end customers to choose from. An end customer who wants to just use the Firehose API to write some in-house solutions would not need to do anything here as they are not looking to push their solution for all Cisco customers. It is important to understand that this guide is created to primarily help the end customers and developers who already own Cisco Spaces licenses and are now looking to get started with doing some in house development around the APIs so the content will be tailored as such. Third party ISVs who are looking to jointly partner with Cisco should reach out to the correct team by filling out the form on <https://partners.dnaspaces.io> to get more information and access to the full Partner API documentation.

In addition, Cisco Spaces also provides a separate dedicated dashboard called the Partner Dashboard which allows you to create, view, update, manage and test the integration of the Firehose API with your account. In this dashboard, you would choose the configuration or what and how to receive the data over the Firehose API. By default, all customers that are hosted on dnaspaces.io can access this dashboard by going to <https://partners.dnaspaces.io> and customers that are hosted on dnaspaces.eu need to request access to this dashboard. Access to the dashboard is a pre-requisite to start leveraging the Firehose API. To request access to the Partner Dashboard, please reach out to your Cisco representative or the Cisco Spaces Support team.

Events

As mentioned before, Firehose API is a streaming API in which all the data will always be sent in real time. All this data is sent in a JSON or Protobuf 'event' message. Every single piece of information is a dedicated event message and there are several types of events that contain distinct types of content. For example, there is an event called 'IoT Telemetry' and an event called 'Device Location Update'. As the name suggests, all the telemetry information like



temperature, humidity, button press etc. would be sent in a message which would be an IoT Telemetry event, whereas all the location information about wireless endpoints would be in a device location update event type message. As a developer, you would be able to choose which types of event data your solution would need, and you can select to only receive and process that type of data. There are multiple event types supported by the Firehose API which are mentioned later in this guide.

Application

In the context of Cisco Spaces, all the use cases are enabled by an application. When you want to leverage the Firehose API to receive the events from the Cisco Spaces cloud, you first create an 'application' in the Partner Dashboard. This 'application' is basically a wrapper around some basic configurations where you define things like a name, description, which events you want to receive, how you want to receive the events (pull vs push) etc. There are other concepts like single tenant or multi-tenant application vs on prem application, but those are primarily meant for ISVs so we will not go into those in this guide. For a customer looking to just receive data from their own Cisco Spaces account, you just need to understand that you need to create an 'Application' in the Partner Dashboard, where you will choose basic configuration settings to start receiving the Firehose data stream. For most of our customers, creating a multi-tenant cloud-based application is going to be adequate, and we will be going over how to create that in this guide.

Activation

Once you have created an application, you need to 'Activate' this application in your Cisco Spaces Dashboard. While the application just has the configuration settings, the data would only start to flow when this application is activated in the account. When you will activate the application, you would be able to choose which locations from the Location Hierarchy you want to send the data from as well as which groups of IoT devices you want to send data from to your application. Once the activation is complete, then the data would automatically start to flow to the endpoint you selected in the Application creation.

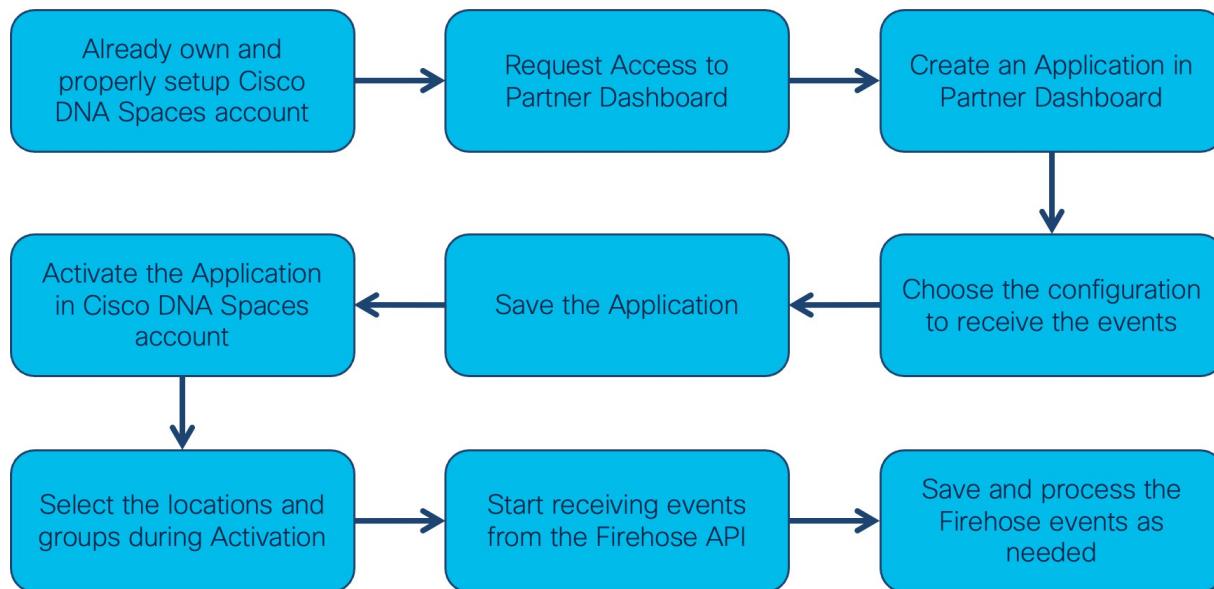
Privacy and Security

All the data which is sent from the Cisco Spaces Cloud is always sent encrypted over HTTPS for both Push and Pull models of Firehose (REST APIs are also always run only on HTTPS for security). So, you could be receiving data directly over the internet into your application or you could be receiving the data into your AWS Kinesis, the data would be encrypted during transit. For third party ISVs, they must integrate OAuth into their application to make sure only authorized users are allowed to activate their application, but since this guide is focused on end customers creating their own application, the OAuth part may not be important. Please refer to the Partner API Documentation for more details.

In addition, during the activation, a customer can choose data sources and locations would be allowed to send their data to the application. They can also delete the activation to stop the flow of data if needed. This allows the customer to have full visibility and control over where their data is going.

To recap, the following flowchart explains the high-level steps an end customer needs to do to start leveraging the Firehose API for their own accounts.

Steps for end customers



Benefits

The Firehose API model also brings in many benefits like:

- It is designed from ground up for cloud scale, designed for low latency and can be used for supporting real time use cases.
- A single channel supports multiple different types of events. Your application can specify the types of events you are interested in.
- A single channel can carry information from multiple tenant accounts that are sharing data with you.
- It supports sending both JSON data and Protocol Buffers (protobuf) objects and supports HTTP/2 protocol.

- JSON is easy to work with and can be directly consumed by most platforms today.
- Protocol Buffers (developed and open sourced by Google Inc.) is a language neutral, binary format that optimizes the bandwidth used and allows faster processing.
- It supports push channels for ease of consuming the data directly in Amazon AWS, Google Cloud or Microsoft Azure.
- Overall, it is easy to implement and better than a Webhook connection and supports data replay capability so re-send the data if needed. That is particularly helpful if your application needs to be offline for system upgrades or faces a short outage.

Events supported by Firehose API

Cisco Spaces can transmit the following event types to your application:

- Device Entry: This event is sent when a device enters a location.
- Device Exit: This event is sent when a device has exited a location.
- Profile Update: This event is sent when a device profile is updated or changed. For example, this event is sent when an end-user provides information in a captive portal.
- Location Information Change: This event is sent when a location is updated. For example, a location is moved under a group, location is renamed or there is a change to location's meta data.
- Tele Presence: This event is sent when the Tele Presence system encounters a people-count update. If you choose the Tele Presence option, then the app requires data generated by the Tele Presence units. This includes the presence event, and the people count event. Presence is provided by devices connected to the Telepresence unit via ultrasound. People count is provided by computer vision from the Telepresence video camera. Cisco Spaces does not receive any video or photographs data. Cisco Spaces receives the final count of people recognized and all video stays on the Tele Presence unit.
- Device Location Update: This event is sent when a device location is updated.
- Location update events continuously generate an approximate location of the device when connected to the network. The app requires X, Y, and/or the latitude, and longitude level location data and a map of your location. If the Device Location Update event is selected in the Event Types section, then one of the following events is selected. The app requires zone, floor, or site level presence data. These events include Location Updates and Device Entry, Device Exit, and Device Dwell data for the devices

connected to your network. These events are generated every time a device enters or exits a building, a floor, or a zone. The information available in Cisco Spaces about these devices and high-level information about their location is also shared. You can choose to receive Geo coordinates data for the device location update event.

- If the Device Location Update event is not selected, then either Device Entry, or Device Entry, or Device Exit, or Device Presence, or the User Presence event is selected in Event Types section.
- App Activation: This event is sent when a customer activates the application.
- Account Admin Change: This event is sent when an account admin gets added/removed/updated for the partner account.
- Keep Alive: This event is sent when there is no other event sent for a period of 15 seconds. This event is applicable only for HTTP/gRPC channels to prevent your connection from timing out.
- Device Presence: This event is used to track the life cycle of a device at a location. Events are generated at various points such as at device entry, when a device is inactive for 10 minutes, when a device is active after being inactive, or when we determine the device has exited. These events also provide the current count of active and inactive devices at location.
- User Presence: Based on the available authentication in use and the information available from the network, Cisco Spaces can map group or multiple devices owned by a user. Events are generated at various points such as at device entry, when a device is inactive for 10 minutes, when a device is active after being inactive, or when we determine the device has exited. These events also provide the current count of active and inactive devices at location.
- IoT Telemetry: This event is sent when there are telemetry updates from BLE, RFID, and Zigbee IoT devices.
- IoT User Action: This event is sent when user actions are performed on IoT devices.
- Device Count: This event is sent when there is a change in the (count) number of devices at the location.
- Camera Count: This event is sent when there is a change in the aggregated count of people (computed via the Meraki Video Camera) at the location.
- Raw Camera Count: This event is sent when there is a change in the individual camera count (computed via the Meraki Video Camera) at the location.
- Network Telemetry: This event is sent at a periodical interval with health and performance telemetries of the location.
- Location Anchor Update: This event is sent when a new location anchor is added to, updated in, or removed from IoT Services.



Below are some examples of messages of different types of events. For full message format of every event type, please refer to the full API documentation referenced at the end of the guide.

Example 1:

Event Type – IOT Telemetry

Use Case – Read the temperature received from a BLE IOT Device.

In this example, we can see the value of the temperature sent in the event by the Cisco Spaces Firehose API along with other details like X/Y position as processed by Detect and Locate, battery life reported etc.

It is expected for the developers to be able to store the record Uid and timestamp to store relevant information locally and then process it as needed.

Furthermore, the location id and map id would correspond to the exact site in the hierarchy (Campus, Building, Floor, Zone etc.) – the details of which are something developers are expected to be able to map locally in their backend.



```
{
  "recordUid": "event-ff205269",
  "recordTimestamp": 1624406443415,
  "spacesTenantId": "dnaspaces-tenant-464026d0",
  "spacesTenantName": "DNA Spaces LAB",
  "partnerTenantId": "dnaspaceslab",
  "eventType": "IOT_TELEMETRY",
  "iotTelemetry": {
    "deviceInfo": {
      "deviceType": "IOT_BLE_DEVICE",
      "deviceId": "",
      "deviceMacAddress": "XXXXXXXXXXXXXX",
      "group": [],
      "deviceName": "",
      "firmwareVersion": "",
      "rawDeviceId": "",
      "manufacturer": ""
    },
    "detectedPosition": {
      "xPos": 167.5,
      "yPos": 167.8,
      "latitude": 0,
      "longitude": 0,
      "confidenceFactor": 64,
      "mapId": "6568ba2637adb1225ba36f10757d0ab2",
      "locationId": "location-549356d4",
      "lastLocatedTime": 1624406440000
    },
    "location": {
      "locationId": "location-549356d4",
      "name": "DNA Spaces Lab",
      "inferredLocationTypes": [
        "FLOOR"
      ],
      "parent": {
        "locationId": "location-2f64620f",
        "name": "SJC-19",
        "inferredLocationTypes": [
          "NETWORK",
          "BUILDING"
        ],
        "parent": {
          "locationId": "location-36f8282c",
          "name": "DNASpacesLab",
          "inferredLocationTypes": [
            "CAMPUS"
          ],
          "parent": {
            "locationId": "location-65fae68e",
            "name": "DNASpacesLAB",
            "inferredLocationTypes": [
              "ROOT"
            ],
            "sourceLocationId": ""
          },
          "sourceLocationId": "f0918a66-8394-4f3b-ae7e-27fdca30da43"
        },
        "sourceLocationId": "c957ba87-4502-4ae9-81f3-246629a82711"
      },
      "sourceLocationId": "bb49a8cc-069a-4017-a1ce-c1eb9689eaa2",
      "floorNumber": 1
    },
    "temperature": {
      "temperatureInCelsius": 24,
      "rawTemperature": 56
    },
    "accelerometer": {
      "x": -1,
      "y": -1,
      "z": 60,
      "lastMovementTimestamp": 0
    },
    "deviceRtcTime": 1624406375000,
    "battery": {
      "value": 100,
      "unit": "PERCENTAGE",
      "lastRetrieved": 1624406442681
    },
    "rawHeader": 0,
    "rawPayload": "AgEGGRZq/gMJAhD//zz///BgFnedJgZAMF/xg=",
    "sequenceNum": 0
  }
}
```



Example 2:

Event Type – IOT Telemetry for Webex Device for People Count, Air Quality, Presence etc.

Use Case – Read the telemetry about Presence of people and count of people in a room as well as air quality information as detected by a Webex endpoint. (Please note that it is assumed that customers have already setup their Webex devices in Cisco Spaces properly).

In this example, we can see the value of presence field (binary True or False) as well as the people count value as reported by the Webex endpoint with the mac address '08:96:AD:43:33:61'. It is expected for the developers to be able to store the record Uid and timestamp to store relevant information locally and then process it as needed.



```
{  
    "recordUid": "event-affe1f5",  
    "recordTimestamp": 1635458061000,  
    "spacesTenantId": "spaces-tenant-464026d0",  
    "spacesTenantName": "D . . . . b",  
    "partnerTenantId": "d . . . . b",  
    "eventType": "IOT_TELEMETRY",  
    "iotTelemetry": {  
        "deviceInfo": {  
            "deviceType": "IOT_TELE_PRESENCE_DEVICE",  
            "deviceId": "D . . . . b",  
            "deviceMacAddress": "08:96:AD:43:33:61",  
            "group": [],  
            "deviceName": "",  
            "firmwareVersion": "",  
            "rawDeviceId": "",  
            "manufacturer": "",  
            "companyId": "",  
            "serviceUuid": "",  
            "label": ""  
        },  
        "location": { },  
        "temperature": {  
            "temperatureInCelsius": -1,  
            "rawTemperature": -1  
        },  
        "deviceRtcTime": 0,  
        "rawHeader": 0,  
        "rawPayload": "",  
        "sequenceNum": 0,  
        "tpData": {  
            "presence": false,  
            "peopleCount": 0,  
            "standbyState": 1,  
            "ambientNoise": 33,  
            "drynessScore": -1,  
            "activeCalls": 0,  
            "presentationState": 0,  
            "timeStamp": 1635458061000,  
            "airQualityIndex": -1,  
            "temperatureInCelsius": -1,  
            "humidityInPercentage": -1,  
            "soundLevel": 34,  
            "ambientLight": "Unavailable",  
            "reverberationTime": 0,  
            "closeProximity": false,  
            "airQualityStatus": ""  
        },  
        "humidity": {  
            "humidityInPercentage": -1,  
            "rawHumidity": -1  
        },  
        "airQuality": {  
            "airQualityIndex": -1,  
            "airQualityPpb": 0,  
            "airQualityStatus": ""  
        },  
        "maxDetectedRssi": 0  
    }  
}
```



Example 3:

Event Type – Device Location Update

Use Case – Use the location of a wireless endpoint (X/Y on Cisco Spaces or geo coordinates if they are setup in Cisco Spaces Dashboard)

```
{
  "recordUid": "event-463f70a7",
  "recordTimestamp": 1624397842765,
  "spacesTenantId": "-----tenant-464026d0",
  "spacesTenantName": "DNA Spaces LAB",
  "partnerTenantId": "dnaspaceslab",
  "eventType": "DEVICE_LOCATION_UPDATE",
  "deviceLocationUpdate": {
    "device": {
      "deviceId": "device-DVAHtv7DZfiDUOnxaEPl",
      "userId": "",
      "tags": [],
      "mobile": "",
      "email": "",
      "gender": "GENDER_NOT_AVAILABLE",
      "firstName": "",
      "lastName": "",
      "postalCode": "",
      "optIns": [],
      "attributes": [],
      "macAddress": "c4:b3:6a:ca:c2:18",
      "manufacturer": "Cisco Systems, Inc",
      "os": "",
      "osVersion": "",
      "type": "NOT_AVAILABLE",
      "socialNetworkInfo": [],
      "deviceModel": "",
      "dhcpProfileInfo": {
        "dcProfileName": "",
        "dcDeviceAssTag": "",
        "dcCertaintyMetric": "",
        "dcProtocolMap": ""
      }
    },
    "location": {
      "locationId": "location-549356d4",
      "name": "DNA Spaces Lab",
      "inferredLocationTypes": [
        "FLOOR"
      ],
      "parent": {
        "locationId": "location-2f64620f",
        "name": "SJC-19",
        "inferredLocationTypes": [
          "NETWORK",
          "BUILDING"
        ],
        "parent": {
          "locationId": "location-36f8282c",
          "name": "DNASpacesLab",
          "inferredLocationTypes": [
            "CAMPUS"
          ],
          "parent": {
            "locationId": "location-65fae68e",
            "name": "DNASpacesLAB",
            "inferredLocationTypes": [
              "ROOT"
            ],
            "sourceLocationId": ""
          },
          "sourceLocationId": "f0918a66-8394-4f3b-ae7e-27fdca30da43"
        },
        "sourceLocationId": "c957ba87-4502-4ae9-81f3-246629a82711"
      },
      "sourceLocationId": "bb49a8cc-069a-4017-a1ce-c1eb9689eaa2",
      "floorNumber": 1
    },
    "ssid": "",
    "rawUserId": "",
    "visitId": "visit--1",
    "lastSeen": 1624397842000,
    "deviceClassification": "",
    "mapId": "6568ba2637adb1225ba36f10757d0ab2",
    "xPos": 173.7,
    "yPos": 293.2,
    "confidenceFactor": 88,
    "latitude": 0,
    "longitude": 0,
    "unc": 0,
    "maxDetectedRssi": -66,
    "ipv4": "",
    "ipv6": []
  }
}
```



Cisco Spaces REST APIs

In addition to the Firehose API, Cisco Spaces continues to also have some areas where REST APIs can be used. Note that we are not adding functionality or improving these APIs as the Firehose API would be the primary API type.

There are four types of REST APIs supported.

1. Detect and Locate REST APIs
2. Asset Locator REST APIs
3. Connector REST APIs
4. Firehose API Supporting REST APIs

Detect and Locate REST API

The Detect and Locate REST APIs allow you to retrieve, add or modify information in Detect and Locate app. The REST APIs are divided into five categories:

- Active clients' location APIs – APIs to retrieve clients' count and location data.
- Clients' location history APIs – APIs to get the devices MAC address list and the details for a given device.
- Notifications APIs - APIs to create WEBHOOKs for notifications.
- Map APIs – APIs to upload, navigate the maps hierarchy, retrieve, and delete map element etc.
- Access points APIs – APIs to get access points count and list.

To use these REST APIs, you must first generate an API Key from the Cisco Spaces Account. An API key is a JSON Web Token (JWT) which is required in each HTTP request header to authenticate and authorize the user. You can generate an API Key from the Detect and Locate, navigating to Notifications > API Keys and then click **Add**. You can choose the expiration of the API key to be up to 365 days.



API Key

YOUR API KEY

```
eyJhbGciOiJIUzI1NiIsInR5cCl6IkpxVCJ9eyJpYnkiOiJMb2NhdGlvbiIsInR5cGUiOiJCZWF
```

COPY AND STORE YOUR API KEY IN A SAFE PLACE

Dashboard does not store API keys in plain text for security reasons, so this is the only time you will be able to record it. If you lose or forgot your API key, you will have to revoke it and generate a new one.

I have stored my new API key

Done

435393

Each authenticated user can have up to ten keys, out of which only five can be active. Below is an example from the POSTMAN client, where the API key has been used as an Authorization header.

The screenshot shows a POSTMAN collection named "get client". A GET request is defined with the URL <https://dnaspaces.io/api/location/v1/history/clients>. In the Headers tab, two headers are present: "Content-Type" set to "application/json" and "Authorization" set to "Bearer eyJhbGciOiJIUzI1NiIsInR5cCl6IkpxVCJ9eyJpYnkiOiJMb2NhdGlvbiIsInR5cGUiOiJC...". The Body tab displays a JSON response with 10 objects, each containing a "macAddress" field. The response status is 200 OK, time 350 ms, and size 52.96 KB.

For all the URI endpoints for the Detect and Locate REST API, please refer to the Location API Documentation at <https://developer.cisco.com/docs/dna-spaces/#Idna-spaces-location-cloud-api>

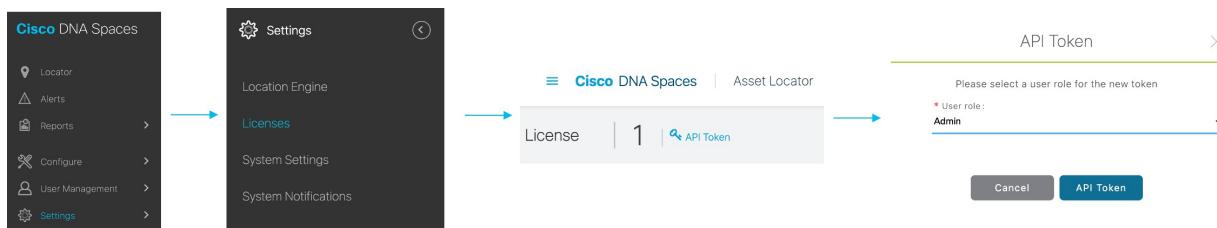
Asset Locator REST API

Asset Locator is Cisco Spaces application for managing, monitoring, and optimizing assets, Internet of Things (IOT) devices, alerting systems, and operational workflows. Using a technology-agnostic approach, the solution can use a wide range of tags, sensors, including Wi-Fi, RFID, and environmental monitors, to continually integrate, monitor, and manage connected

operations. The Asset Locator REST APIs allow customers to perform wide operations to retrieve, add or modify information in Asset Locator app.

To run the Asset Locator REST APIs, customers need to first obtain an access token, which would need to be done via a two-step process.

In the first step, customers would need to create a permanent API token for the Asset Locator app. To do that, in the Cisco Spaces Dashboard, navigate to the Asset Locator application > Settings > Licenses > API Token > Generate an API token for Admin purposes.



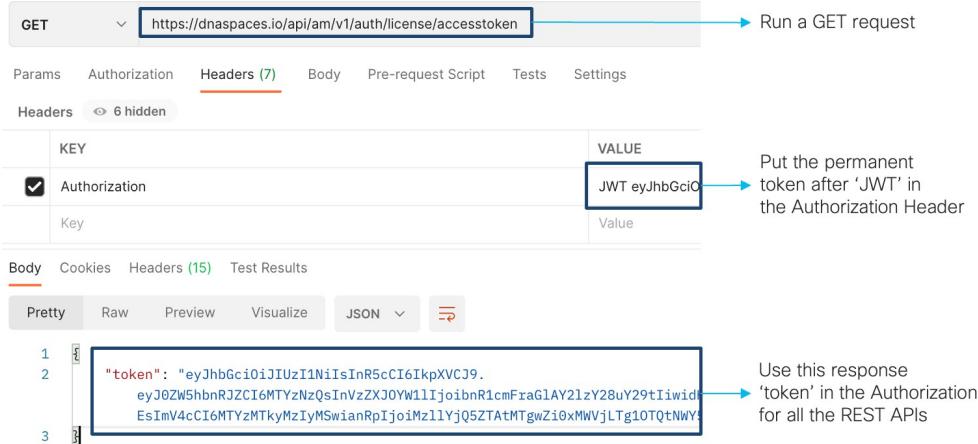
Note: This unique API token should be kept private. It does not expire and can be used to inject malicious data into your account if stolen or exposed resulting in potential service disruption. The user assumes full responsibility and liability for the management of this key. Please notify Cisco immediately if an API token is lost, stolen, or compromised.

Copy this permanent token and keep it private. It will be needed in the second step.

The second step is to get a temporary access token which will be used to make the REST API calls. To retrieve this token, make a GET request to <https://dnaspaces.io/api/am/v1/auth/license/accesstoken>.

Note: For EU customers, please change the URL to dnaspaces.eu.

Attach the permanent API token from the first step along with the prefix "JWT " in the request header under key "Authorization". This API will return an object where you can find the API access token under key "token". This 'token' will be used in the authorization for all Asset Locator REST APIs. Attach it along with the prefix "JWT " in the request header under key "Authorization" to the APIs you wish to execute. The token has an expiry time of 30 minutes.



Run a GET request

Headers (7)

KEY	VALUE
<input checked="" type="checkbox"/> Authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5hbnRJZC16MTYzMzQsInVzZXJOYW1lIjoibnR1cmFraGlAY2lzY28uY29tIiwidEsImV4cCI6MTYzMzIyMSwianRpIjoimz1lYjQ5ZTAtMTgwZi0xMWVjLTg1OTQtNWY...
Key	Value

Put the permanent token after 'JWT' in the Authorization Header

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON

```

1 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
2 eyJ0ZW5hbnRJZC16MTYzMzQsInVzZXJOYW1lIjoibnR1cmFraGlAY2lzY28uY29tIiwidEsImV4cCI6MTYzMzIyMSwianRpIjoimz1lYjQ5ZTAtMTgwZi0xMWVjLTg1OTQtNWY...
3"

```

Use this response 'token' in the Authorization for all the REST APIs

Once this temporary token is available, you can use it to run the Asset Locator REST APIs. For all the URI endpoints for the Detect and Locate REST API, please refer to the Location API Documentation at <https://www.cisco.com/c/dam/en/us/td/docs/wireless/cisco-dna-spaces/tech-notes/asset-locator-api-ref.pdf>

Cisco Spaces Connector APIs

Cisco Spaces Connector 2.3.2 REST API

The Cisco Spaces Connector (aka ‘Connector’) enables Cisco Spaces to communicate with multiple controllers and switches efficiently, by allowing each controller to transmit client data without missing any client information.

The Connector gathers and aggregates data from controllers, access points (APs), and switches efficiently and sends aggregated data to Cisco Spaces. The Connector architecture allows multiple controllers, APs, and switches to connect to Cisco Spaces through a single point and a single connector can connect to a multiple wireless LAN Controllers (Both Catalyst and AireOS based) and Cisco Catalyst 9300/9400 Series Switches at the same time.

Starting with Cisco Spaces Connector version 2.3.2, customers can run local REST APIs against the connector for configuration, monitoring and automation.



Secure

APIs do not use Basic Auth but rather multi step Token based authentication



Monitor

Create tools to monitor connectors by receiving stats and connector data locally via API



Automate

Use APIs to automate configuration of privacy configurations and rotate hash and SALTs

Pre-requisites for using REST APIs:

APIs are receiving data directly from the connector docker. So:

- Docker container should be up, and
- The connector needs to be added to a DNA Spaces Account

The connector supports four REST API endpoints

- Getting of the Access Token to run APIs
- Configure the hashing algorithm used by the connector. The connector can either be configured to not do hashing. And if it is, it can use either SHA1 or SHA512 algorithm to hash PII (Personally Identifiable Information) data.
- Configure the privacy settings like SALT used to hash data and prevent IP addresses from going to the Cisco Spaces Cloud.
- Read data and stats from the connector locally. This is helpful for doing any automated monitoring locally on the connector to make sure the health of the system is good.

Base URL – <https://<connector-ip>/api/connector>

GET

/accesstoken

This API needs to be run first to get access token to run any other APIs below. Access token is valid for 24h

GET

POST

/hashingalgorithm

Used to read or configure the hashing algorithm used for the connector (Either SHA-1 or SHA-512)

GET

POST

/privacysettings

Used to read or configure privacy settings (mac / username SALT, hide or show IP address)

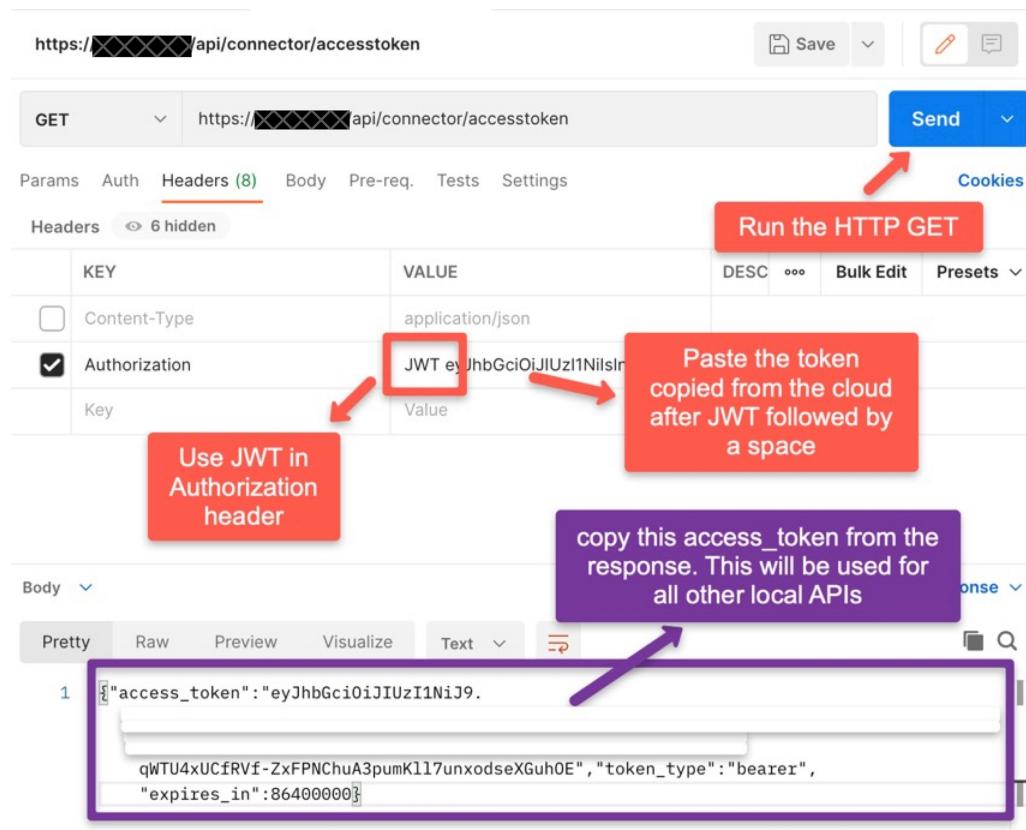
GET

/monitoring/data

Used to read data and stats from the connector locally. Useful to feed information into local monitoring tools.

Getting the Access Token

To get the access token to run REST APIs on the connector locally, customers need to first use the connector's token from the Cisco Spaces Dashboard. This is the same token that was pasted in the connector to set it up in the account on Day 1. If needed, it can be copied from the Dashboard again by navigating to Setup > Wireless > Connectors > Hover over the three dots > View Token > Copy. Once this cloud token is available, run a GET on <https://<ip-of-connector>/api/connector/accesstoken> and put this cloud token after 'JWT' in the Authorization Header. This would provide an 'access_token' which can then be used to run the subsequent REST APIs against the connector.



The screenshot shows the Postman API client interface. The URL in the header is <https://<ip-of-connector>/api/connector/accesstoken>. The method is set to GET. The Headers tab is selected, showing the following configuration:

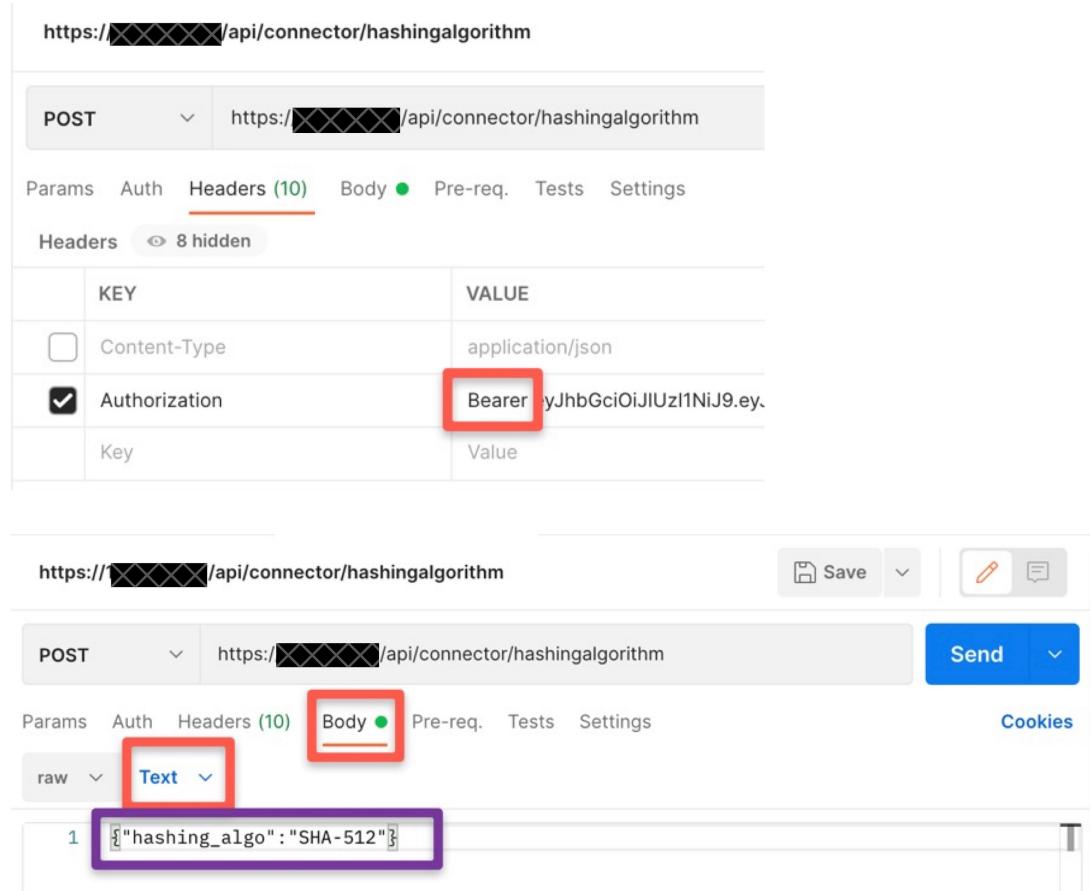
KEY	VALUE
Content-Type	application/json
Authorization	JWT eyJhbGciOiJIUzI1NiJ9.
Key	Value

A red box highlights the 'Authorization' key-value pair, with an annotation: "Use JWT in Authorization header". Another red box highlights the value 'JWT eyJhbGciOiJIUzI1NiJ9.', with an annotation: "Paste the token copied from the cloud after JWT followed by a space". A purple box highlights the response body, containing the JSON object: {"access_token": "eyJhbGciOiJIUzI1NiJ9.qWTU4xUCfRVf-ZxFPNChuA3pumK117unxdseXGuh0E", "token_type": "bearer", "expires_in": 86400000}. An annotation above the response body says: "copy this access_token from the response. This will be used for all other local APIs".

Configure the hashing algorithm

Once the access_token is available, paste that after 'Bearer' in the Authorization header and run GET on <https://<ip-of-connector>/api/connector/hashingalgorithm>

The response will show the hashing algorithm as either 'SHA-1' or 'SHA-512'. To change the configuration, run a POST on the same URL but in the body of the request, provide the value as either {"hashing_algo": "SHA-1"} or {"hashing_algo": "SHA-512"}



https://<ip-of-connector>/api/connector/hashingalgorithm

POST https://<ip-of-connector>/api/connector/hashingalgorithm

Headers (10) Body Pre-req. Tests Settings

Headers 8 hidden

KEY	VALUE
<input type="checkbox"/> Content-Type	application/json
<input checked="" type="checkbox"/> Authorization	Bearer <access_token>
Key	Value

https://<ip-of-connector>/api/connector/hashingalgorithm

POST https://<ip-of-connector>/api/connector/hashingalgorithm Send

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

raw Text

```
1 {"hashing_algo": "SHA-512"}
```

Configure the privacy settings

Once the access_token is available, paste that after 'Bearer' in the Authorization header and run GET on <https://<ip-of-connector>/api/connector/privacysettings>

The response will show the values of 'hide_ip', 'username_salt', 'mac_salt'.

The hide IP value is a binary True or False to show whether the connector will send IP addresses to the Cisco Spaces cloud or not.

The username SALT is a string showing the SALT to be used to hash the usernames.

The mac SALT is a string showing the SALT to be used to hash the mac address.

To change the configuration, run a POST on the same URL but in the body of the request, provide the value as

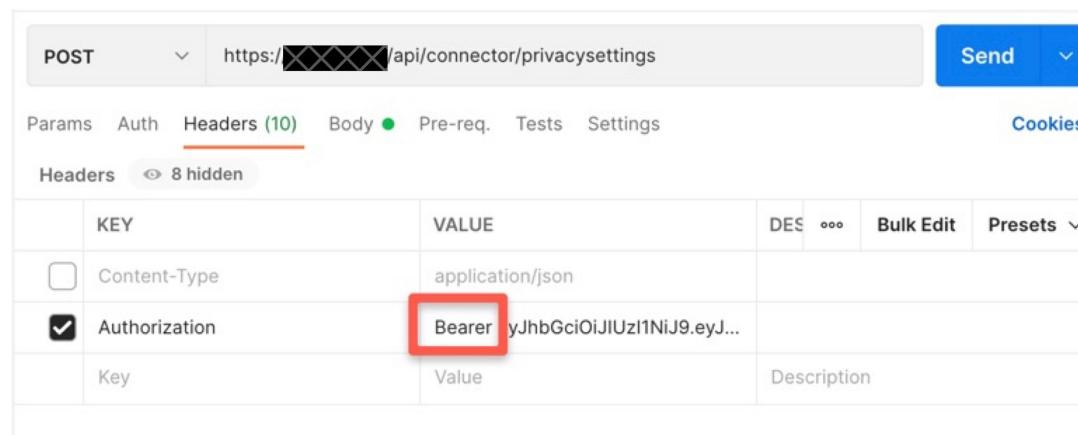
```
{"hide_ip":x, "username_salt":y, "mac_salt":z}
```

where,

x would be true or false

y would be the username SALT string or empty

z would be the mac SALT string or empty

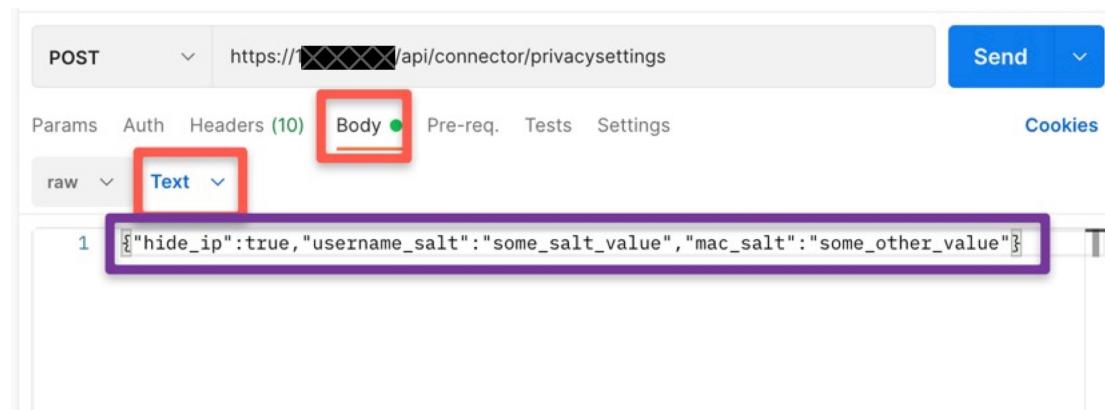


POST https://████████/api/connector/privacysettings Send

Headers (10) Params Auth Body Pre-req. Tests Settings Cookies

Headers (8 hidden)

KEY	VALUE	DES	Bulk Edit	Presets
<input type="checkbox"/> Content-Type	application/json	ooo		
<input checked="" type="checkbox"/> Authorization	Bearer yJhbGciOiJIUzI1NiJ9eyJ...			
Key	Value	Description		



POST https://████████/api/connector/privacysettings Send

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

raw Text

```
1 {"hide_ip":true, "username_salt":"some_salt_value", "mac_salt":"some_other_value"}
```

Note: Any values configured via this API would always take precedence over GUI (Graphical User Interface) based values. For example:



- If there is a value for mac salt configured via API, then it does not matter if a different value is entered in GUI or even if GUI is left blank, connector will still use the value entered via API.
- If the API is used to enter a blank value, then if there is a value entered in the GUI, then the GUI value would automatically be taken
- If the API is used to enter a blank value, and if GUI is also kept blank, only then will the connector assume no configuration.

Read data and stats from the connector

Once the access_token is available, paste that after 'Bearer' in the Authorization header and run GET on <https://<ip-of-connector>/api/connector/monitoring/data>

The response will show the various stats and details from the connector like OS Detail, controller(s) status, connectivity status, network and memory metrics, incoming and outgoing message rates etc.

Customers can use this API to monitor the health of the connector and can create automation around it to alert the admin in case of the connector running into any issues like running out of memory, losing connectivity with Cisco Spaces cloud, losing a connector etc.



GET https://XXXXXX/api/connector/monitoring/data Send

Params Auth Headers (8) Body Pre-req. Tests Settings

Headers 6 hidden

KEY	VALUE	DE:	...	Bulk Edit	Presets
Content-Type	application/json				
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9eyJ...				

The header is a Bearer token

Paste the access_token copied from the previous API

Body

Pretty

```
1 "currentTime": 1629592280972,  
2 "timezone": "Pacific Standard Time",  
3 "uptime": 15207221,  
4 "osArch": "amd64",  
5 "osName": "Linux",  
6 "osVersion": "5.13.2-1.el7.elrepo.x86_64",  
7 "ipAddress": "XXXXXX",  
8 "currentVersion": "v2.0.555",  
9 "goldVersion": "v2.0.153",  
10 "latency": 81,  
11 "controllers": [],  
12 "controlChannelConnectionStatus": "Connected",  
13 "dataChannelConnectionStatus": "Connected",  
14 "metrics": {  
15     "map": {  
16         "outgoing-messsage-rate": {  
17             "map": {  
18                 "mean_rate": 0.0,  
19                 "count": 0,  
20                 "units": "events/second",  
21                 "m15_rate": 0.0,  
22                 "m1_rate": 0.0  
23             }  
24         }  
25     }  
26 }
```

Cisco Spaces Connector 3.0 REST API

Monitoring API

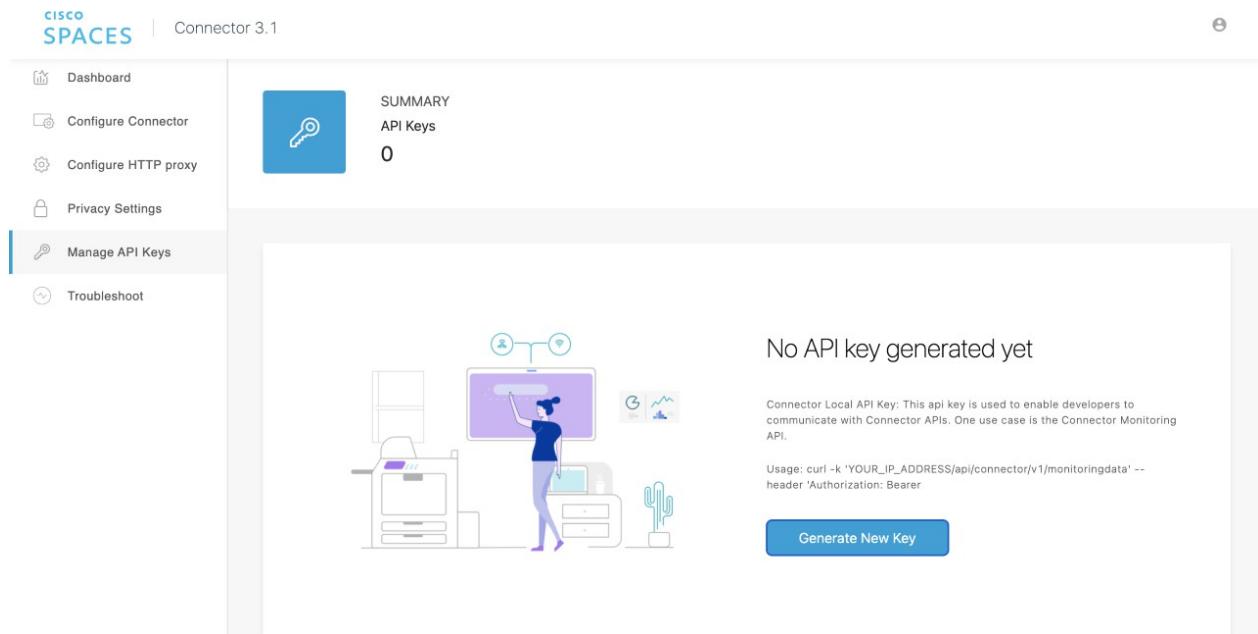
For monitoring the connector and the services from the external monitoring stations and tools we implemented an API to fetch all the monitoring data like

- Connector system monitoring params like CPU
 - Connector Services monitoring params

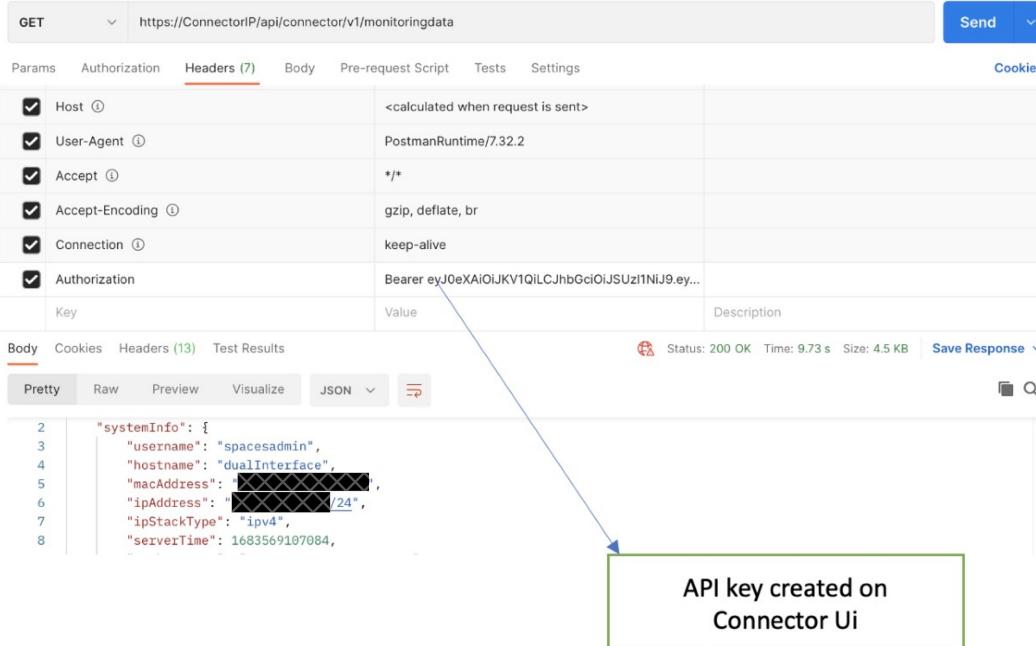
Monitoring API Access

- Create an API access Key from the Cisco Spaces Connector UI and copy the key and save it for future use.

Note: If you don't copy and keep your API key, you will need to create another one.



- Now you can use curl command or any clients like Postman, add the Header as Authorization[key] and value as Bearer followed by the API key created and run the GET on following end point:
<https://<ConnectorIP>/api/connector/v1/monitoringdata>



The screenshot shows a Postman interface with a GET request to `https://ConnectorIP/api/connector/v1/monitoringdata`. In the Headers section, the `Authorization` header is set to `Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ...`. The response body is displayed in JSON format:

```

2   "systemInfo": {
3     "username": "spacesadmin",
4     "hostname": "dualInterface",
5     "macAddress": "[REDACTED]",
6     "ipAddress": "[REDACTED]/24",
7     "ipStackType": "ipv4",
8     "serverTime": 1683569107084,

```

A callout box points from the API key in the response body to the text "API key created on Connector Ui".

- The monitoring API should return all the health information about the service-manager and the services that was spawned through the service-manager.

Configure Privacy Settings

Once you obtain the access_token, paste it after 'Bearer' in the Authorization header and run a GET request on <https://<connector-ip>/api/connector/v1/privacy>. The server returns the values for:

- `hide_ip`: A binary **True** or **False** whether the connector will transmit IP addresses to the Cisco Spaces cloud.
- `username_salt`: A string that represents the SALT for hashing usernames.
- `mac_salt`: A string that represents the SALT for hashing MAC addresses

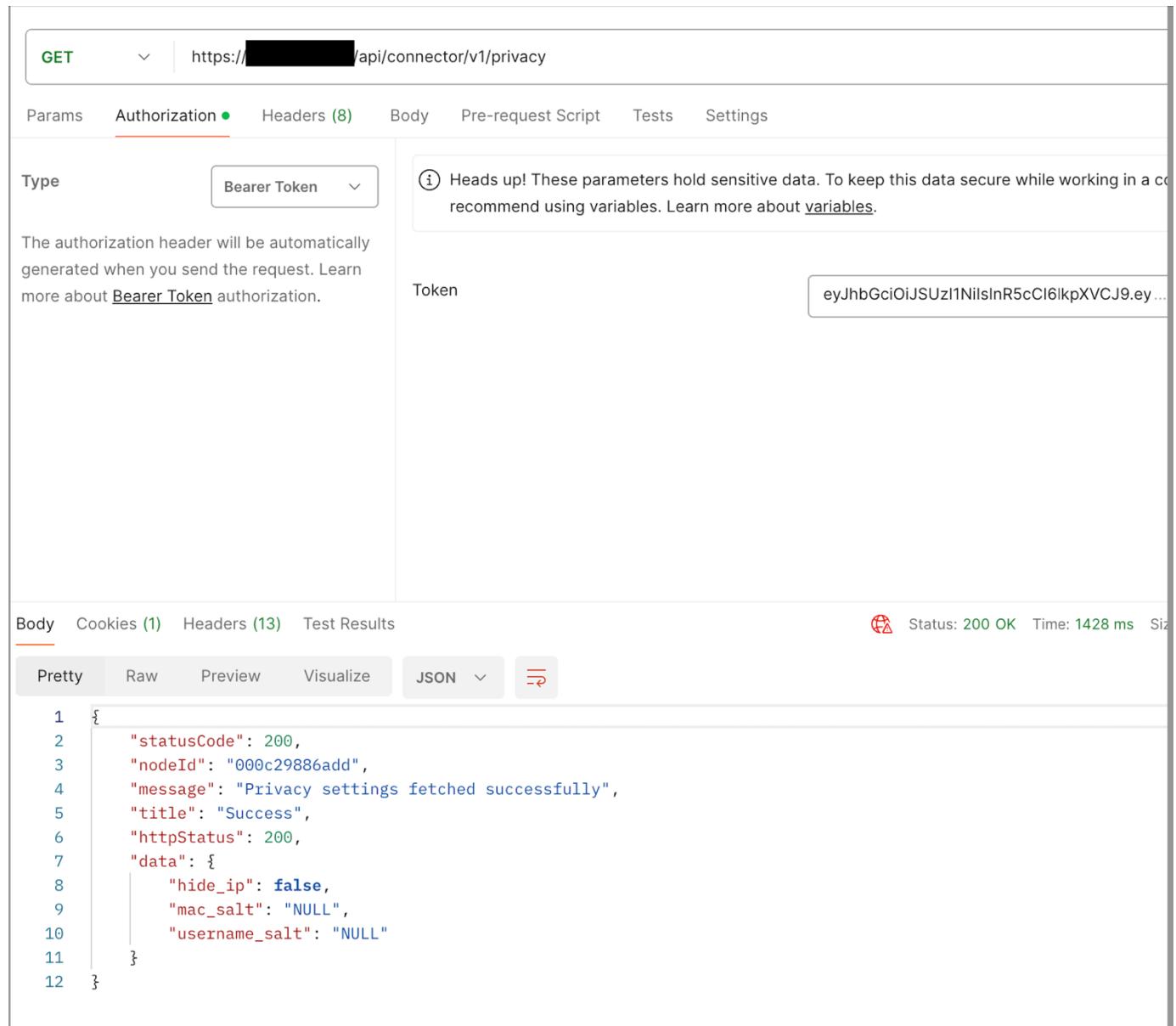
To modify the configuration, perform a POST request to the same URL and include in the request body the values as `{"hide_ip": x, "username_salt": "y", "mac_salt": "z"}`, where x is either true or false, y is the username SALT string or empty, and z is the mac SALT string or empty.

Note that any values that are set using this API will consistently take precedence over those that are set using the Graphical User Interface (GUI).



For instance:

- If you configure a mac salt value through the API, the connector will use this value regardless of any different value entered in the GUI or even if the GUI is left blank.
- If you use the API to enter a blank value for any setting, and there is a value entered in the GUI, the connector will default to using the GUI value.
- If you enter a blank value through the API and the GUI is also left blank, the connector will then assume there is no configuration set.



The screenshot shows a POST request in Postman to the URL `https://[REDACTED]/api/connector/v1/privacy`. The 'Authorization' tab is selected, showing a 'Bearer Token' dropdown and a note about sensitive data. The 'Body' tab displays a JSON response with status code 200, node ID, message, title, http status, and data fields. The 'Pretty' tab shows the JSON response in a readable format.

GET https://[REDACTED]/api/connector/v1/privacy

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Type Bearer Token Heads up! These parameters hold sensitive data. To keep this data secure while working in a public browser, we recommend using variables. Learn more about [variables](#).

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token eyJhbGciOiJSUzI1NjlsInR5cCI6IkpxVCJ9eyJ...

Body Cookies (1) Headers (13) Test Results Status: 200 OK Time: 1428 ms Size: 0

Pretty Raw Preview Visualize JSON

```
1 {  
2   "statusCode": 200,  
3   "nodeId": "000c29886add",  
4   "message": "Privacy settings fetched successfully",  
5   "title": "Success",  
6   "httpStatus": 200,  
7   "data": {  
8     "hide_ip": false,  
9     "mac_salt": "NULL",  
10    "username_salt": "NULL"  
11  }  
12}
```



POST [https://\[REDACTED\]/api/connector/v1/privacy](https://[REDACTED]/api/connector/v1/privacy)

Params Authorization ● Headers (10) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ▾

```
1 {
2   "hide_ip": true,
3   "mac_salt": "",
4   "username_salt": "test123"
5 }
```

Body Cookies (1) Headers (13) Test Results 🕒 Status: 200 OK Time: 527 ms Size: 748 B

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   "statusCode": 200,
3   "nodeId": "000c29886add",
4   "message": "Privacy settings saved successfully",
5   "title": "Success",
6   "httpStatus": 200,
7   "data": {
8     "mac_salt": {
9       "flag": "NOT_SET",
10      "value": "NULL"
11    },
12    "username_salt": {
13      "flag": "SET",
14      "value": "*****"
15    },
16    "ip_data": {
17      "flag": "SET",
18      "value": "NOT_VISIBLE"
19    }
20  }
21}
```



Firehose supporting REST APIs

The final type of REST APIs that are available in Cisco Spaces are around the Firehose. Customers who are using the firehose API in their solutions to consume various event types and event data can leverage these supporting APIs to get data for specific entities such as location, map, device profile configuration etc. All these APIs require the X-API-KEY which is generated in the Partner Dashboard when the customer has created their application. The different supporting APIs are:

- Location Info – Get location information by passing a location ID
- Map Image Info – Get map image information by passing map ID
- Map Image – Get map image by passing map ID
- Devices Lookup - Get Device ID by passing remote IP and client IP
- Create or Update MAC Filters - create or update MAC filters to be used for a tenant
- Get MAC Filters – to get the MAC filters used in a tenant
- Update Device Profile - To update device profile data
- Get Device Profile Data - To get device profile data
- Healthcheck Info - To get firehose health info

Providing all the details for these REST APIs is out of scope for this guide. For more details, please refer to the full Cisco Spaces Partner API Documentation mentioned in the reference section.

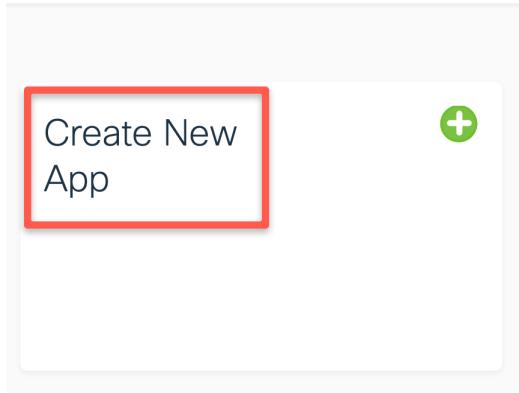
Create your first application and view the Firehose API data

Let us look at the process to get started on creating an application in the Partner Dashboard and use it to start receiving events from the Firehose API. This sample application creation is meant to help developers understand how they can start to leverage the API and then use the data from the platform to create their own solutions. It is assumed that you have access to the Cisco Spaces Partner Dashboard already. If not, please reach out to your Cisco representative or the Cisco Spaces Support team.

Step 1: Login into the Partner Dashboard (<https://partners.dnaspaces.io>) or (<https://partners.dnaspaces.eu>). In this dashboard, click 'Create New App'. This would be the start of an 'Application' where you would choose the different configuration settings for receiving the Firehose API events.



≡ [Cisco DNA Spaces](#) | Partners



Step 2: Choose an App Type. You will see three options – Multi Tenant Cloud, Single Tenant Cloud and On-Prem. These are more advanced options fit for ISV (Independent Software Vendors) partners who may have specific needs. But for the purposes of this guide, we are focusing on enabling end customers. For ease of use and the most options to receive the

events, we recommend choosing the multi-tenant cloud option. Please choose that selection and hit Create.

Choose App Type

Multi Tenant Cloud

- Multi Tenant Cloud apps are available to all users via the DNA Space App Center.

Single Tenant Cloud

- Single Tenant Cloud apps are available specific to partners who would like to build solutions.

On-Prem

- On Prem apps are hosted on specific customers infrastructure and are specific to partners who have existing relationships with customers and would like to build solutions for them.

[Cancel](#) [Create](#)

Step 3: You will now see multiple options in different tabs, namely App Center, Use Cases, App Tile, Behaviors, Events, and Integration Details. We will go over each of the steps but not every field is mandatory. In the 'App Center' Tab, please fill in the mandatory items like region, name, a tagline, description, image etc. This is how the application would show when you go to activate it. Since you are primarily looking to only enable the events from your own account, please fill in the details that make sense to you and you can differentiate between different apps you may create. Then go the next tab of Use Cases.



≡ Cisco DNA Spaces | Partners

My Test App CLOUD

App Center Use Cases App Tile Behaviors Events Integration Details

App Information

Fields marked as * are mandatory

Choose the region *

Based on the region you select, the app will be published in the respective region for customer activation.

NOTE: Looks like you don't have an account in the EU region. Please contact support to setup an account in the EU region if you want to test or publish apps to the Europe region.

Europe Region Rest of the World (except Europe region)

APP Name *

My Test App

APP Tagline *

My App Tagline

APP Icon *

Choose File No file chosen

APP Description *

Some Description

Primary Industry *

Hospitality

Step 4: In the Use Cases tab, you can add some use cases to be shown when you activate your app. Since you do not really need to do this for your own account, you can skip adding any use cases. It is not mandatory.

Step 5: In the App Tile tab, please add a Tile label and a tagline. This is also how your app will be visible during activation. You can add what makes sense to you. Please keep the OAuth option checked and you should see some OAuth URL configurations pre-filled. You can leave them as default if you do not want to use your own OAuth settings. If you need, you can make appropriate changes to the OAuth configurations to validate any activation against your organization's authentication system. For most end customers, leaving it as default would be fine.



≡ Cisco DNA Spaces | Partners

My Test App CLOUD

App Center Use Cases App Tile Behaviors Events

Fields marked as * are mandatory

APP Tile Label *
My App Tile Label

App Tile Tagline *
Some App Tile Tagline

App Activation

OAuth *
Checking this box enables OAuth

Client ID *
dnaspaces

Client Secret *
239337f245eb497880e94bd57200b13c4bdc2f8a

Redirect URI
https://partners.dnaspaces.io/partner/OAuthValidation

OAuth URL Configurations

App Dashboard URL *
Note: Please ensure this URL starts with - https://
<https://trigue.dnaspaces.io/auth/appLogin>

OAuth Login URL *
Note: Please ensure this URL starts with - https://
<https://trigue.dnaspaces.io/auth/login>

OAuth Token URL *
Note: Please ensure this URL starts with - https://
<https://trigue.dnaspaces.io/auth/token>

App Info URL *
Note: Please ensure this URL starts with - https://
<https://trigue.dnaspaces.io/appinfo>

Step 6: In the Behaviors tab, you can choose how to support automated sign ups which is something more relevant to ISV Partners. You can leave it as empty.

Step 7: In the Events tab, you would choose the type of events which you would receive upon the activation of your app in your account. The selection of these events should be based on what your end goal is. You can choose a single event, or all events and you can also edit these selections later. Descriptions of the different event types were already covered earlier in this guide.



≡ Cisco DNA Spaces | Partners

My Test App CLOUD

App Center Use Cases App Tile Behaviors Events Integration Details

Event Types

Device Entry

This event is sent when a device enters a location.

Device Exit

This event is sent when a device has exited a location.

Profile Update

This event is sent when a device profile is updated. For example, this event is sent when an end-user provides information in a captive portal.

Location Information Change

This event is sent when a location is updated or changed. For example, a location is moved under a group, location is renamed or there is a change to location's metadata.

Device Location Update

This event is sent when a device location is updated.

Step 8: At the bottom of the Events tab, you will see the Event Settings options. Here you can choose which type of locations you would like to see the data from, whether you would like to receive MAC addresses in the events, whether you would like to receive social identifiers (like emails or social media IDs) in the events, and finally whether you would like to receive simulated events in the Firehose stream.

For the location types, you can choose from a combination of Root Level, Groups, Network, Floor or Zones. The default option of all locations is a good place to start and depending on your end goals, you should tweak this to receive the optimal number of events and avoid having to process unnecessary events.

Also, the option of receiving simulated events is very useful to test your application and backend processing without having to activate the app on your production Cisco Spaces account. This is a good option for developers to use as they are starting to build their systems to make sure they can receive different types of simulated event messages and they are able to receive, store and process as needed. Once the system is ready to be deployed, you can either create another app or edit this option.



Event Settings

Choose Locations

Send events that are triggered at the following locations

All Location Types Location Types

MAC Address sharing

Receive Client Mac addresses with events

No Yes

Social Identifier Sharing

Receive Social Identifier with Events

No Yes

Receive Simulation Events

No Yes

Retail Workspace

Step 9: Go to the Integration Details tab. In this tab, you would make the choice of how you would like to receive the Firehose events. There are two types of options to choose from – push channels and pull channels (default). Push channels enable customers who already have some cloud-based systems deployed to be able to leverage cloud to cloud integration from Cisco Spaces to directly push the events to AWS Kinesis, AWS Kinesis Firehose, Microsoft Azure Event Hubs or Google Pub/Sub systems. If you need to use one of these, please choose that option and provide information like Access Keys, secrets, stream name, Topic ID etc. You can even choose multiple options or a mix of options. Each of these would be an independent stream. As another option you can choose to test out your system via a Pull channel (HTTP or gRPC). If you choose these options, the system will create a set of API keys for you to use for this specific application. You would need to use that X-API-KEY and make a pull call to Cisco Spaces to start receiving the events. For the purpose of this guide, and for developers to get started, we recommend just choosing the Pull channels to begin the integration and if needed, migrate to Push based channels to leverage the benefits of cloud systems.



 Cisco DNA Spaces | Partners

My Test App

CLOUD

App Center Use Cases App Tile Behaviors Events

s Integration Details

Integrations Types

Push Channels

- AWS Kinesis**
Use this channel to push events to an AWS Kinesis Data Stream

 - AWS Kinesis Firehose**
Use this channel to push events to an AWS Kinesis Data Firehose

 - Azure Event Hubs**
Use this channel to push events to an Azure Event Hubs

 - Google Pub/Sub**
Use this channel to push events to Google Pub/Sub

Pull Channels

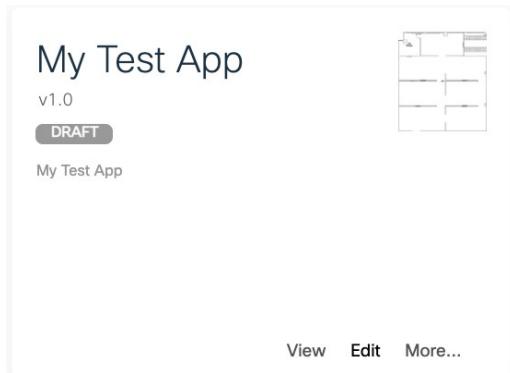
- HTTP**
Pull events over a streaming HTTP Connection
 - gRPC**
Pull events over a streaming gRPC call

API Credentials for Pull Channels

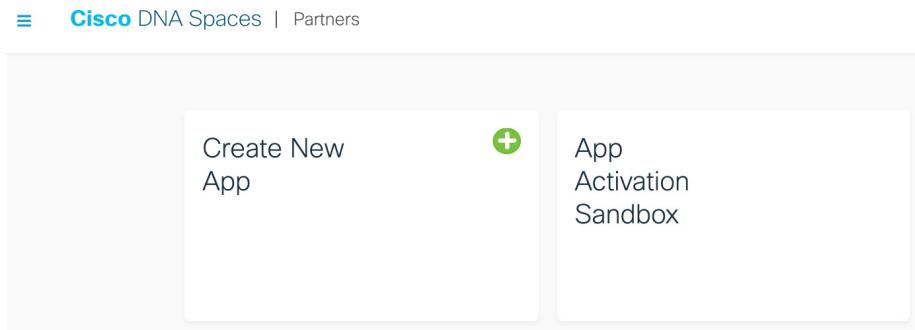
Note: On creation of the Application, the below keys will be generated. The same can be viewed through Edit App once created.

Environment API Key

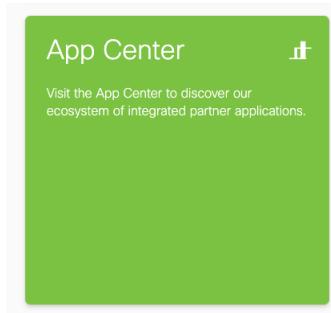
Step 10: Make sure all the needed configurations and hit Create. At this point, you would be able to see the app you created in the Your Apps section in the Partners Dashboard. You would see it as a ‘Draft’ App to begin with. You can click the Edit button in the App Tile to make changes. Once the configuration looks good, the next step is to activate the application and start consuming the Firehose API data.



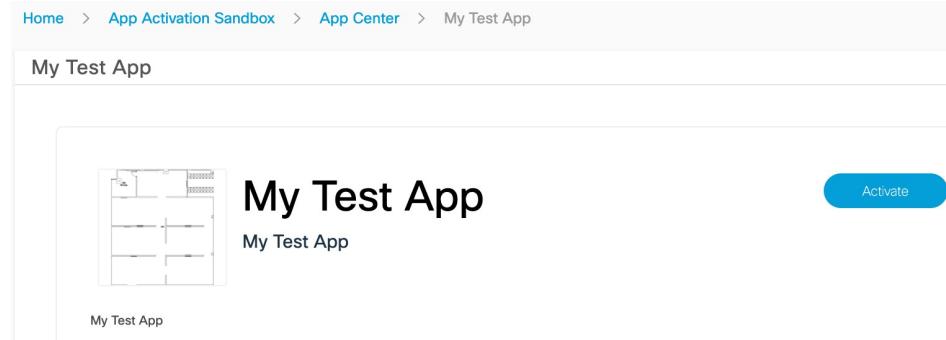
Step 11: In the Partner Dashboard, click on App Activation Sandbox. You would see a Sandbox Cisco Spaces Dashboard page.



Step 12: Scroll down and select the App Center tile to view the applications available to be activated. You would see your newly created application in the list of apps.



Step 13: Select your app and click Activate.



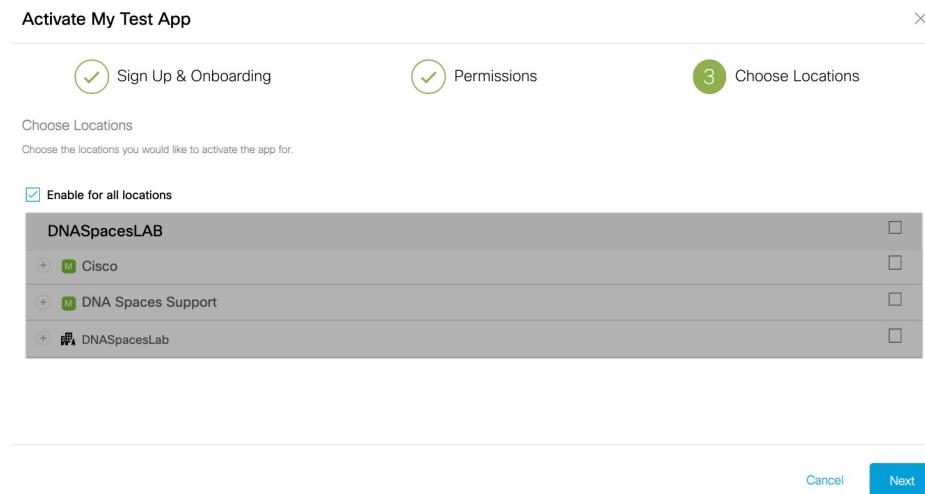
Home > App Activation Sandbox > App Center > My Test App

My Test App

My Test App

Activate

Step 14: In the activation workflow, you can leave the sign-up option as default (I have an existing account). In the permissions tab, you can review the data that will be sent and accept the permissions. This is based on the selection of the events you have made in the app creation. In the Choose Locations tab, you can choose which specific locations from the location hierarchy you would like to see the events from. Many customers run tests with a specific location to test their systems before enabling the events from all the locations. If you have specific IOT Telemetry events and groups, you can also choose to receive data from specific groups only. The details such as device ID, group name etc. would be part of the Firehose event. Click Next after selecting the locations.



Activate My Test App

Sign Up & Onboarding

Permissions

Choose Locations

Choose Locations

Choose the locations you would like to activate the app for.

Enable for all locations

DNASpacesLAB	<input type="checkbox"/>
+ Cisco	<input type="checkbox"/>
+ DNA Spaces Support	<input type="checkbox"/>
+ DNASpacesLab	<input type="checkbox"/>

Cancel

Next

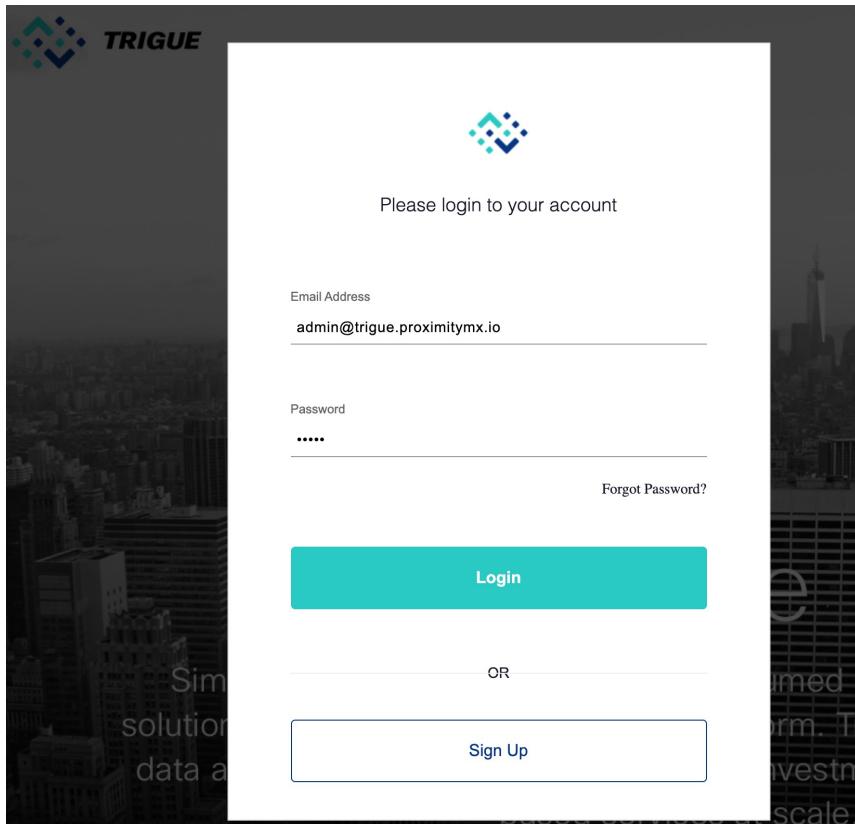
Step 15: Next, you would see the OAuth splash page. Since we went with the default pre-filled OAuth configuration during the app creation, you would see a default page here. If you have chosen your specific OAuth settings, then this is where you would see your login page.



For the default process, you should see the email address and password filled in. If not, please enter these credentials on the page and click Login.

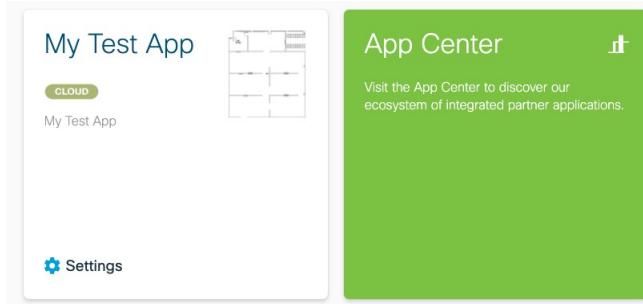
Email Address: admin@trigue.proximitymx.io

Password: admin

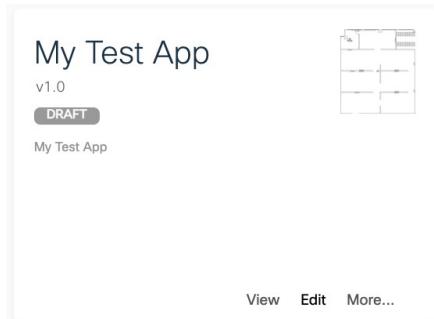


Step 16: Click Next and proceed leaving the values as default. You should see the App Activation Successful message and your app should be visible in the Sandbox Dashboard now confirming that the activation is complete.

App Activation Successfully



Step 17: Now that the app is activated, you can start to receive the events. If you had chosen push channels, then the events have already started to be sent from Cisco Spaces. For Pull channels, you need to get the API key to use. To get your API key, go back to the partner dashboard where you have created your app and click Edit.



Step 18: Go to the Integration Tab and you will see the API keys available for Sandbox, Pre-Production and Production environments. You can use any but for purposes of testing, we recommend just choosing the sandbox key. Copy your key and keep it safe.



☰ Cisco DNA Spaces | Partners

My Test App CLOUD | Rest of World | DRAFT | v1.0

App Center Use Cases App Tile Behaviors Events Integration Details

Pull Channels

HTTP
Pull events over a streaming HTTP Connection

gRPC
Pull events over a streaming gRPC call

API Credentials for Pull Channels

Use these channels to power your apps. As a security measure we suggest renewing your key every 90 days

Environment	API Key
Sandbox Created on 18th Sep 2021 Recommended renewal - 90 days	16B*****B38
Pre Production Created on 18th Sep 2021 Recommended renewal - 90 days	CE8*****E20
Production Created on 18th Sep 2021 Recommended renewal - 90 days	B11*****921

Step 19: With the API key, there are multiple options for you to start receiving the events. But at the fundamental level, you just need to send a HTTP request to <https://partners.dnaspaces.io/api/partners/v1/firehose/events> (Or .eu for EU customers) along with passing your API key as a X-API-KEY in Header. While a proper way would be to receive the events and store them in a database (One example of Cisco Spaces Firehose + MongoDB Handler is provided in reference section), for quick tests one of the options would be to just receive events via a terminal window and curl command.

For example, run the following command in your terminal window. Replace the key with your API key.

```
curl "https://partners.dnaspaces.io/api/partners/v1/firehose/events" -H "X-API-Key:B1*****878F"
```



```
niravturokhia@NTURAKHI-M-V3UV -> curl "https://partners.dnaspaces.io/api/partners/v1/firehose/events" -H "X-API-Key:78...J6F"
{"recordUid": "event-749e172c", "recordTimestamp": 1632012133905, "spacesTenantId": "", "spacesTenantName": "", "partnerTenantId": "", "eventType": "KEEP_ALIVE"}, {"recordUid": "event-b8d0bf4b", "recordTimestamp": 1632012133855, "spacesTenantId": "spaces-tenant-9bb0484b", "spacesTenantName": "WirelessTMEDMZ", "partnerTenantId": "wirelessstmedmz", "eventType": "IOT_TELEMETRY", "iotTelemetry": {"deviceInfo": {"deviceType": "IOT_BLE_DEVICE", "deviceId": "d4:57:57:eb:95:aa", "deviceMacAddress": "d4:57:57:eb:95:aa", "label": "IV Pump - 2343"}, "detectedPosition": {"xPos": 61.9, "yPos": 39.1, "latitude": 37.40747204903919, "longitude": -121.92864158937431, "confidenceFactor": 32.0, "mapId": "", "locationId": "location-bb2b03b9", "lastLocatedTime": 1632012131000}, "location": {"locationId": "location-bb2b03b9", "name": "Main Lab", "inferredLocationTypes": ["CMXZONE"], "parent": {"locationId": "location-26b5c0e7", "name": "TME Lab Floor", "inferredLocationTypes": ["FLLOOR"], "parent": {"locationId": "location-8428fc03", "name": "SJC14 TME Lab", "inferredLocationTypes": ["NETWORK", "BUILDING"], "parent": {"locationId": "location-b5154fle", "name": "WirelessTMEDMZ", "inferredLocationTypes": ["ROOT"]}, "sourceLocationId": "", "sourceLocationId": "73b105c9-7833-493d-9b03-1422fce445cc"}, "sourceLocationId": "00107e72-f840-4c05-bb1f-04cdebea75ae", "sourceLocationId": "425fb3a1-e01f-4763-ab2b-bbeb002172fd"}, "deviceRtcTime": -1, "rawHeader": 0, "rawPayload": "AgEGDxZa/gIIagfI/GdZNGIxM0==", "sequenceNum": 0, "maxDetectedRssi": -68}}, {"recordUid": "event-bf215cf4", "recordTimestamp": 1632012133948, "spacesTenantId": "spaces-tenant-9bb0484b", "spacesTenantName": "WirelessTMEDMZ", "partnerTenantId": "wirelessstmedmz", "eventType": "IOT_TELEMETRY", "iotTelemetry": {"deviceInfo": {"deviceType": "IOT_BLE_DEVICE", "deviceId": "d3:28:81:6b:a7:0e", "deviceMacAddress": "d3:28:81:6b:a7:0e", "label": "", "group": ["Entry-Beacons"], "detectedPosition": {"xPos": 35.4, "yPos": 34.9, "latitude": 37.407395799615685, "longitude": -121.9286551289762, "confidenceFactor": 4.0, "mapId": "", "locationId": "location-bb2b03b9", "lastLocatedTime": 1632012130000}, "location": {"locationId": "location-bb2b03b9", "name": "Main Lab", "inferredLocationTypes": ["CMXZONE"], "parent": {"locationId": "location-26b5c0e7", "name": "TME Lab Floor", "inferredLocationTypes": ["FLLOOR"], "parent": {"locationId": "location-8428fc03", "name": "SJC14 TME Lab", "inferredLocationTypes": ["NETWORK", "BUILDING"], "parent": {"locationId": "location-16642a87", "name": "Spaces", "inferredLocationTypes": ["CAMPUS"], "parent": {"locationId": "location-b5154fle", "name": "WirelessTMEDMZ", "inferredLocationTypes": ["ROOT"]}, "sourceLocationId": "", "sourceLocationId": "73b105c9-7833-493d-9b03-1422fce445cc"}, "sourceLocationId": "00107e72-f840-4c05-bb1f-04cdebea75ae", "sourceLocationId": "425fb3a1-e01f-4763-ab2b-bbeb002172fd"}, "deviceRtcTime": -1, "rawHeader": 0, "rawPayload": "AgEGERZq/gKAAAFk9DYwcDkwMDJD", "sequenceNum": 0, "maxDetectedRssi": -68}}}
```

Step 20: At this point, customers and developers would be expected to do their local storage and processing of the events and create the solution they are looking to create. The Firehose API is responsible for providing all the events in a streaming fashion and delivering it at scale. You can make changes to the application in the Partner Dashboard and even delete the activation from the App Activation Sandbox Dashboard. Please note that if you have elected to receive simulation events, then those simulated events would continue to be shown. Please turn those off once you are ready to activate the events in your production Cisco Spaces account.

Best Practices for using Firehose API

For advanced use cases and applications, customers / developers may need to harden their solutions further. Listed below are the common best practices for HTTP/gRPC.

- Reconnection during connection failures -
In case of any errors on the HTTP/GRPC channels, customers should handle the errors by following standard retry policies. We recommend you implement an exponential retry with a 5-minute time reset to avoid error overheads. You should use the fromTimestamp parameter to retrieve old data during this error period.
- De-duplication of events -
Duplication of events may occur due to stream failures/retries/any other cases. You should be able to handle de-duplication of events on your end using the recordUid field.
- Privacy Settings -
We recommend you look at Privacy Settings part of the application creation and decide the PII fields you want to enable in your app based on real need or use case, instead of opting to receive all event details. For example, if your use case needs to uniquely

identify a device, you may not need to receive the MAC address details, instead each event carries the deviceld (generated by Cisco Spaces) and that may be adequate.

- Events Settings -
We recommend you look at Events Settings part of the application creation and select only the events that you are interested in, instead of selecting all event types. For example: if your use case deals with locating the current physical location on a floor, you might only need the Device Location Updates, App Activation, and the Location Change events type details.
- Location Settings -
We recommend you look at Location Settings part of the application creation and select only the Location Type settings that you are interested in, instead of selecting all event types. For example: if your use case deals with floor location, you might only need the Floor Location Type event details.

Troubleshooting and Frequently Asked Questions

Q: I am not able to access the Partner Dashboard

A: Access to the Partner Dashboard is currently not open to all Cisco Spaces customers by default. Please reach out to your Cisco representative or Cisco Spaces Support team to request access to the Partner Dashboard. Cisco Spaces team would consider the request, check your licenses, and accept or deny the access request as appropriate.

Q: I am not able to see any events, or I only see KEEP_ALIVE messages. How should I troubleshoot?

A: Please check the steps mentioned in the guide to make sure you have followed the steps.

Most common reasons for not being able to see data events are:

- a. You have selected to not receive any simulated events and in your production account, there is no activity happening which would be generating any events. The Firehose API is an events-based API which would not send any event when there is no change.
- b. You have not selected the right location(s) during the activation.
- c. You have chosen to receive data from some IOT groups, but devices are in a different group or vice versa.
- d. You have not selected the right events in the app creation. Consider selecting some other type of events.
- e. You have used the incorrect API key.

Q: Does Cisco not provide a REST API for all the apps like with CMX before? The Cisco Spaces REST APIs are not as complete as CMX before.



A: No. We have made a conscious decision to move away from traditional APIs due to the nature of always having to query the system and limited scalability. Firehose API is designed from the ground up for cloud scale and we have achieved incredible levels of data that is passing over the API daily (multiple TBs per day). The Firehose API is the future direction we have taken, and all new developments would be on the Firehose API. We provide limited REST based APIs for some older applications to make it easier for customers to be able to migrate from legacy to next-generation systems. But customers need to start looking at developing their applications by leveraging the Firehose API.

Q: Can I receive IOT data (like BLE) using any REST APIs?

A: No. IOT streaming telemetry is only available via the Firehose API.

Q: I have existing applications created using REST APIs on CMX. Can I just migrate them by pointing to Cisco Spaces URL instead?

A: No. The CMX and Cisco Spaces REST APIs are not 1:1. Migrating applications from CMX to Cisco Spaces may be very easy depending on your use case, but customers need to understand the APIs available in Cisco Spaces and decide about what is the best way for them to migrate over.

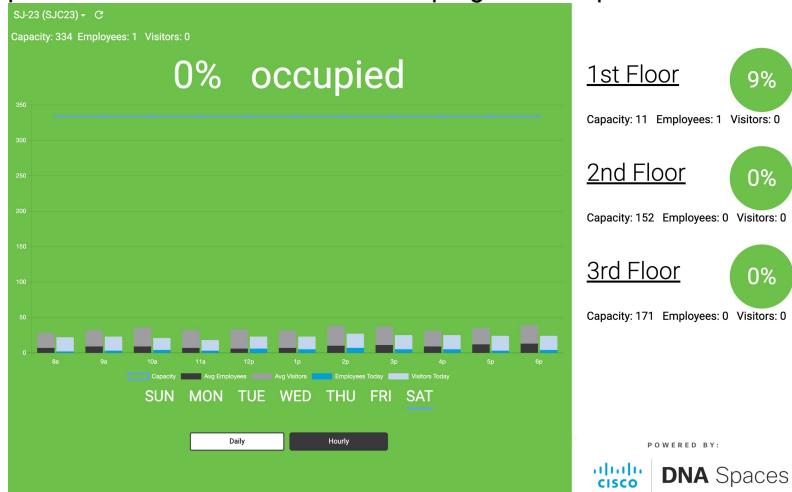
Q: Can you share some examples of real-world Cisco Spaces customers who have created solutions using Firehose API?

A: We have many Partners that are currently available in Cisco Spaces Partner App Center which leverage the Firehose API for powering actual real world use cases at scale for paying customers. These are companies who have built a business around the Cisco Spaces platform. In addition, we have multiple Cisco end customers who have also leveraged the Firehose API to create their own in-house solutions for their business needs.

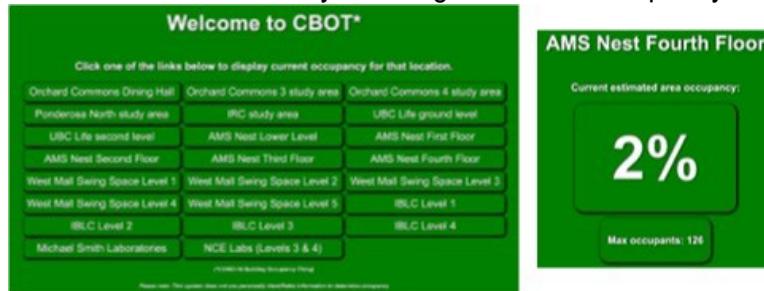
Some examples are:

- a. Cisco IT – They are leveraging the Cisco Spaces platform to provide an occupancy use case to visualize the number of people in Cisco campus buildings and making sure Covid

protocols are adhered to and keeping the campus safe for Cisco employees.



- b. University of British Columbia – They are also leveraging the Cisco Spaces platform to derive occupancy levels and create a safe University Campus. Students can choose to view the occupancy levels of multiple locations on the campus before leaving their homes to make sure they do not go over the occupancy limits allowed per regulation.



Q: Is there any language version compatibility I need to be aware of when using Firehose API?

A: No. The Firehose API is meant to be a way to provide all the events in real time to customers. Once the customers receive the events either via push or pull channels, the solution creation is completely up to them.

Q: Is there a scale limitation with Firehose API?

A: No. Cisco Spaces Firehose API is specifically created to remove any scale limitations that exist with legacy platforms. The Firehose API is used to power mission critical use cases in customers' networks worldwide.



Q: I still need help. What should I do if I require assistance?

A: Please see the reference at the end of the guide for additional documentation. If you still need help with creating your own solution using Cisco Spaces APIs, please reach out to the Cisco Spaces support team.

Example Script to use Firehose API

To help customers better understand how to use the Firehose API, a sample script is provided below. The goal of this set of scripts is to just consume Firehose API input in a streaming fashion from your Cisco Spaces account.

Example code available at: <https://github.com/niravturakhia/FirehoseTest>

References

- Cisco Spaces Partner Dashboard (US Region) –
<https://partners.dnaspaces.io> Cisco Spaces Partner Dashboard (EU Region)
– <https://partners.dnaspaces.eu>
- Cisco Spaces Partner API Documentation -
<https://partners.dnaspaces.io/documentation> API Events Details -
<https://www.cisco.com/c/en/us/td/docs/wireless/cisco-dna-spaces/partner-app/partner-firehose-api/std.html>
- Firehose Sample App (Detect and Locate) –
<https://github.com/CiscoDevNet/DNAspaces-FirehoseAPI-DetectAndLocate>
- Firehose Sample App (Right Now) - <https://github.com/CiscoDevNet/DNAspaces-FirehoseAPI-RightNowVisitors>
- Firehose Sample App (Occupancy) - <https://github.com/pawel-kontakt/demo-space-occupancy> Cisco Spaces – MongoDB Handler Sample script -
https://developer.cisco.com/codeexchange/github/repo/SimonLight001/DNASFirehose_Mongo_DB_Handler
- iOS and Android SDK (Software Development Kit) for implementing Cisco Spaces Open Roaming
 - <https://github.com/CiscoDevNet/DNAspacesSDK-IOS>