Accessing Asset Tracker Device Data from Google PubSub

Overview

NimbeLink uses Google Pub/Sub to decouple the data stream produced by the asset tracker devices and NLink from your organization's data ingestion application. The Pub/Sub service provides a buffer between your application and NLink so that data sent by devices is not lost if your application is unavailable for short periods of time. Unacknowledged device data is available in your Pub/Sub queue for up to 7 days, after which time it will be discarded.

NimbeLink will create an organization specific Pub/Sub subscription, limited access service-account and provide you with the project_id, subscription, and topic name as well as the credentials file needed to connect and retrieve device data from the Google Pub/Sub service.

Setup

Using the Google SDK is the easiest method to get started with Pub/Sub and confirm that you are able to access your device data stream.

1. Download and install the Google Cloud SDK at https://cloud.google.com/sdk/. Skip the "gcloud init" step since you are going to be using a pre-configured service account to access the Pub/Sub data.

2. Set your specific information (provided by NimbeLink) as environment variables:

```
$ PROJECT_ID="nl-at-XXXXXX-prod"
$ SUBSCRIPTION="org-XXXXXX"
```

3. Create a local configuration

```
$ gcloud config configurations create myconfig
Created [myconfig].
Activated [myconfig].
```

4. After obtaining your credentials file from NimbeLink, add it to your local config for testing

```
$ gcloud auth activate-service-account --key-file ./nl-at-XXXXXX-prod.json
Activated service account credentials for: [org-abc123@alpha-beta.iam.gserviceaccount.com]
```

5. Set your default project

```
$ gcloud config set project ${PROJECT_ID}
Updated property [core/project].
```

Retrieve Device Data

Pull a single message from the Pub/Sub subscription and auto-acknowledge it, which removes it from the queue:

```
$ gcloud pubsub subscriptions pull ${SUBSCRIPTION} --project ${PROJECT_ID} --auto-ack --limit 1 --format json
[
  {
    "ackId": "ABC...123",
    "message": {
      "attributes": {
        "eventID": "ABC...789",
        "id": "at-xxxxxxxxxxxx"
      },
      "data": "W10=",
      "messageId": "000000000000000",
      "publishTime": "2017-09-01T14:22:11.274Z"
    }
  }
]
```

`[].message.data` contains the base64 encoded payload from NLink which is described in the Asset_Tracker_Outbound_JSON_Schema document.

The device ID that the record originated from is available in the **`[].message.attributes.id`** field as well as in the base64 encoded data.

Further Examples

Google provided client libraries (C#, GO, JAVA, NODE.JS, PHP, PYTHON, RUBY) are available for application integration here: https://cloud.google.com/pubsub/docs/reference/libraries

For example, to pull messages from the subscription using Python: https://googlecloudplatform.github.io/google-cloud-python/latest/pubsub/subscriber/index.html

The following Python snippet prints all the messages in the queue to the console but doesn't acknowledge them:

```
from google.cloud import pubsub_v1

def callback(message):
    # handle message
    print(message)

project_id = 'project-id'
sub_name = 'org-xxxxxx'

client = pubsub_v1.SubscriberClient()
sub_path = client.subscription_path(project_id, sub_name)

subscription = client.subscribe(sub_path,callback=callback)


subscription.open(callback)

while (1):
    pass
```