Predix Ubom Application Documentation/Guide

Created By Robert Freedy

**Overview**

This application handles the ingestion and up-keep of Ubom data. It uses the UAA and Asset Microservices to handle the Ubom data.

**Necessary Software**

* Predix DevBox (optional, but recommended)
  + Most of the below software is included in this
  + Can be downloaded on predix.io under the Resources link
* Cloud Foundry
* Eclipse
* Git
* Java SE Development Kit (JDK), Version 8
* Maven
* Node.js
* Cygwin
* Bower
* Grunt
* Python
* Postman REST Client

**Recommendations for starting in Predix**

* Predix Bootcamp
* Predix Tutorials on predix.io under Guides
* Talking to Predix experts on what Predix is/does

**Downloading the Application**

Make a directory for your project. Clone or download and extract the seed in that directory.

git clone https://github.com/rob-freedy/Asset-ingestion-app

**Services and Application Details**

* Application Name: Asset-ingestion-app (Can be changed in the manifest.yml)
* Services
  + UAA Service: Asset-uaa-service
    - Clients
      * Admin
        + Client Secret: admin\_secret
        + Permissions: Basic read and write for clients, Basic UAA
      * client1
        + Client Secret: client\_secret
        + Permissions: Everything, including Asset and UAA
    - Users
      * user1
        + Email: [user1@gmail.com](mailto:user1@gmail.com)
        + Password: user1
        + Permissions: Views Service, Asset Service
      * rocket
        + Email: [rocket@example.com](mailto:rocket@example.com)
        + Password: Gu@rdian5
        + Permissions: None
  + Asset Service: Asset-asset-service
  + Views Service: itpe\_views\_service\_instance
  + Redis Service: itpe\_redis\_service\_instance

**Files in the Repository**

manifest.yml – Changing the name of your application

app.js – main Javascript file that runs the application

predix-asset-service.js, connect.js, nginx.conf – Files to connect UI with microservices

**Data Model/Conversion from CSV to JSON**

-The CSV file must have the following fields:

Customer Name, Road Number, Part Number, Parent Part Number, Serial Number, and Position

The Position and Serial Number fields can be left blank, but they do need to exist

-uri : the defining tag for each of the individual parts.

locomotive/[customer name]~[road number]~[Part Number]~[Parent Part Number]~s~[Serial Number]~p~[Position]

To run script, enter the command line, change directories to the folder that the python script and Ubom CSV file are in and run the following command: python CSV-JSON.py . This will prompt you for an input file and the json will output into a file named jsonOutput.json

**Helpful Links/People**

**Sites-**

https://predix.io

Main Predix website. Contains all the tutorials and guides that you need.

<https://github.build.ge.com/HealthcareITPEPredixTraining/itpe-training-view-complete>

Healthcare example UI, contact Brian Henzelmann for any help with UI

<https://github.build.ge.com/GET-Data-And-Analytics>

Data and Analytics Predix Application example

<https://www.predix.io/support/article/KB0010674>

Deleting Services and Applications

**People-**

Ashok Madhuranath – GE Transportation, Digital Thread Hub

Brian Henzelmann – GE Healthcare, UI and General Application help

Kamran Saeed – GE Transportation, Chicago

Andy Wickersham – GE Transportation, Chicago

Mark Rutherford – GE Digital Predix Asset Team

Joe Schmaltz – GE Aviation Predix Asset/APM/ALM Team

**Creating your Own Application**

**BEFORE STARTING:**

Make sure you have downloaded all of the necessary software from predix.io. This can be found under the Guides section under the Development environment tab

Also make sure to set your proxy variables if you are not working in the Predix DevBox

For GE, the following proxy environment variables can be set through the command line using the following commands on the command line:

set HTTP\_PROXY=http://GET-Alpha-Pitc-SSO.corporate.ge.com:80

set HTTPS\_PROXY=http://GET-Alpha-Pitc-SSO.corporate.ge.com:80

set no\_proxy=http://GET-Alpha-Pitc-SSO.corporate.ge.com

**Basic Starter Application**

This is the basic Nodejs starter app that can be found on the Guides section of predix.io. The Basic Application Tutorial is very helpful for anyone just starting on Predix.

Start by cloning this repository: <https://github.com/PredixDev/predix-nodejs-starter.git>

git clone <https://github.com/PredixDev/predix-nodejs-starter.git>

cd predix-nodejs-starter

After cloning this basic application and changing directories, run the following command:

npm install

cx

Inside of your cloned repository directory, open the manifest.yml file. This file is where you can define the name of the app as well as where you can define some microservices. For now, change the name of your app to something you will remember. After editing, save this file.

Login into CloudFoundry/Predix by entering the following command in the command line:

cf login

If asked for an API Endpoint, enter the following:

<https://api.system.aws-usw02-pr.ice.predix.io>

Enter your username and password for your Predix Account.

After successfully logging into your CloudFoundry/Predix account, you can now push to your local development space by running the following command:

cf push

After the application as started (You will see that the application has started after pushing successfully to CloudFoundry), enter the following url into your browser:

https://<name-of-predix-app>.run.aws-usw02-pr.ice.predix.io

where <name-of-predix-app> is the name you entered for the application in the manifest.yml file. You should see a basic Predix starter application. The links that you click on will not lead you to anything until you connect a UAA microservice to the starter application.

**Creating and Connecting a UAA Microservice**

Login into CloudFoundry by entering the following command in the command line:

cf login

If asked for an API Endpoint, enter the following:

https://api.system.aws-usw02-pr.ice.predix.io

Enter your username and password for your Predix Account.

After logging into CloudFoundry, you can run the following command to create your UAA service instance:

cf create-service predix-uaa Free <service-instance-name> -c ‘{“adminClientSecret”:”<my-secret>”}’

where <service-instance-name> is the name you want to call you UAA microservice and <my-secret> is your password for your admin client. Both are these are important and should be remembered/written down.

After creating the UAA microservice, we have to bind it to the starter application. This can be done by entering the following command:

cf bind-service <application-name> <UAA-service-instance>

where <application-name> is the name of your application that was created above and <UAA-service-instance> is the name of your UAA microservice.

We must now create a client that contains the permissions for the UAA microservice. In order to do this, you can use the Predix Starter Kit Website or the command line. We will use the Predix Starter Kit here because it is easier.

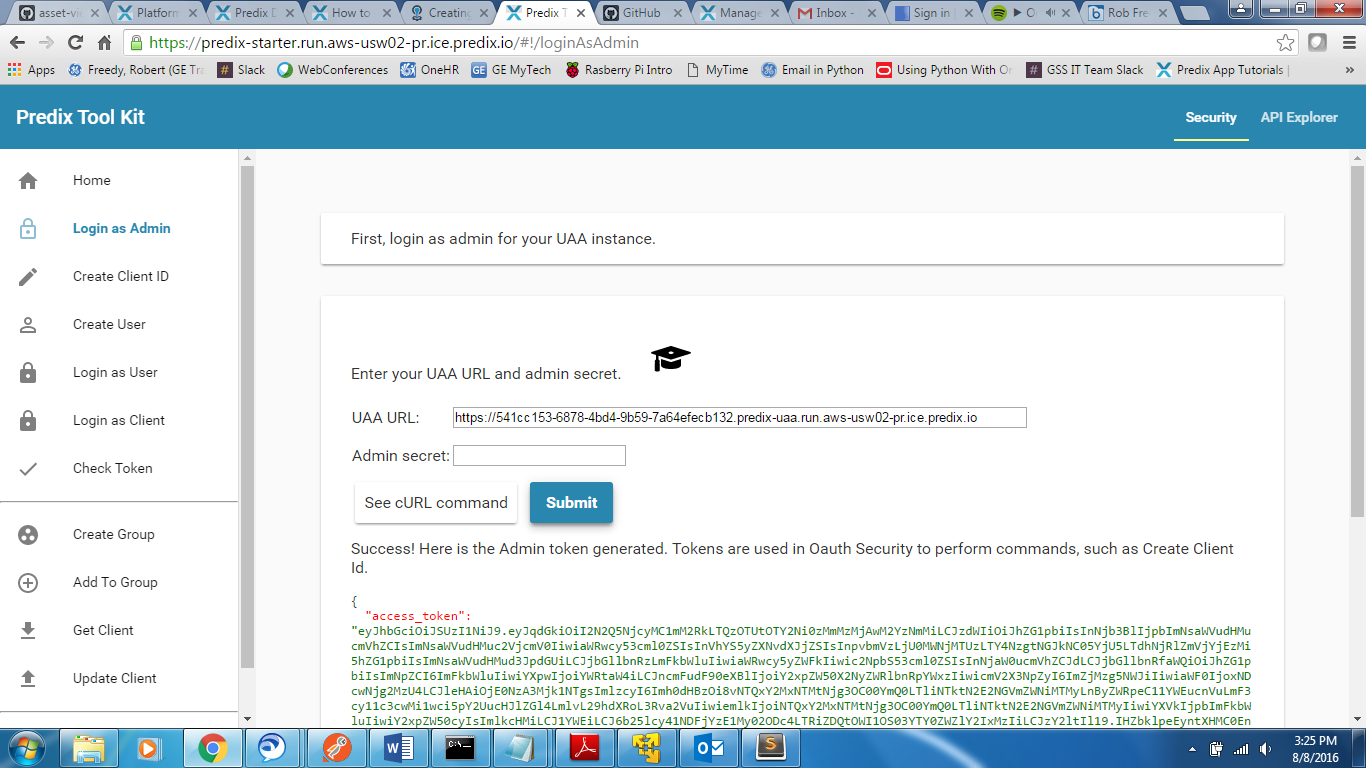
Link for the Predix Starter Kit: <https://predix-starter.run.aws-usw02-pr.ice.predix.io/#!/updateClient>

In order to use the website, you must obtain your uaa service uri. This can be done by entering the following command:

cf env <name-of-your-app>

where <name-of-your-app> is what you named the application. This command will give you all of the microservices that are bound to your application. At this point, the only microservice that should be bound to your app is your UAA microservice.

Copy the value under the UAA Service that is labeled “uri”. On the Starter Kit website, click on the Login as Admin section and paste the value you just copied. Enter your admin password and click “Submit”. If you bound your microservice correctly, it should return a success message as well as an access token.



After logging into your UAA service as an admin, you can now create clients and users with permissions to give you access to all of the Predix microservices.

To create a client, click on the Create Client Id panel on the left side of the screen. Enter a client name and a client secret. Make sure to write these down because you will need them later.

Next, click on the Login as Client to login in as the client you just created. If successful, you should get an Authorization token.

To create a user, click on the Create User panel on the left side of the starter screen. Create a user name and password for your uaa service. Make sure to write this username and password down because you will need it later.

Click on the Login as User to verify that your user is under the correct client and was properly created. If successful, all of the user attributes should be displayed. We will update these later in the guide.

**Creating and Connecting an Asset Microservice**

There are two ways to create an instance of the Asset Microservice: Create it using the catalog on predix.io or create it through the command line. For this guide, we will create it using the command line.

After logging into cloud foundry (instructions for logging in are shown above), copy the issuerId variable from your UAA microservice by running the following command:

cf env <your-application-name>

where <your-application-name> is the name of your application.

After copying the issuerId variable from your UAA service, create your instance of the Asset microservice by running the following command

cf create-service predix-asset Tiered <name-of-service> -c ‘{“trustedIssuerId”:[“<uaa\_issuerId>”]}’

where <name-of-service> is what you want to call your Asset microservice and <uaa\_issuerId> is the issuerId that you copied above.

If the asset service cannot be created through the command line, go to predix.io and click on Catalog. Under the catalog find the asset service and create your asset service through the web page.

To make sure that both your UAA and Asset services have been created, run the following command to see what services you have in your space:

cf services

Bind your asset service to your application using the following command:

cf bind-service <your-application-name> <your-Asset-service-name>

where <your-application-name> is the name of your application and <your-Asset-service-name> is the name of your Asset microservice.

Next, we have to update the client permissions to allow for a specific client to be able to ingest data into the microservice. To do this we must go to the Predix Starter website which is pasted below:

Predix Starter WebSite: https://predix-starter.run.aws-usw02-pr.ice.predix.io

Log in to your UAA microservice using your UAA microservice uri (can be obtained using the cf env command described above). After logging in to your microservice, log in as the client that you created above.

After logging into the client, click on the “Get Client” feature on the bottom left side of the web page. Type in your client name and click submit. A list of your clients attributes should appear. After getting your client, click on the bottom left of the screen to “Update Client”. This tool can be used to update your client with certain authorities and scopes. Update your client to have the following attributes. Enter each of the authorities below into the new authority bar and press submit after entering each one.

<Asset\_Instance\_Id> and <UAA\_instance\_Id> can be found by running the cf env command that was described above. The Asset instance id can be found under the “instanceID:” value of the Asset microservice description from the cf env <app-name> command. The Uaa instance id can be found under the “http-value” description from the cf env <app-name> command.

Authorities:

clients.read

clients.secret

idps.write

uaa.resource

scim.me

openid

zones.<UAA\_Instance\_Id>.admin

clients.admin

scim.read

predix-asset.zones.<Asset\_Instance\_Id>.user

clients.write

idps.read

scim.write

After updating all of these attributes, run the following command to push your application to CloudFoundry:

cf push

**Obtaining an Authorization Token and Ingesting Data**

Before starting, make sure you have the Postman REST Client installed. We will use this software to ingest our data.

First, we must obtain an authorization token from our client which we have given the asset ingestion permissions. We can do this either through the Predix Starter website (link above) or the Postman REST Client.

We will use the Starter Kit Website for this part, but a demo of how to use the Postman REST client for obtaining an Authorization token can be found on page of 87 of the Predix Bootcamp Lab Guide (can be found in the box I made).

Link for Starter Website: <https://predix-starter.run.aws-usw02-pr.ice.predix.io>

Click on “Login as Admin” on the left side of the screen. Log into your UAA Service by entering the UAA url (can be found using the cf env command) and your admin secret. Next, log into your client by clicking on the “Login as Client”. Enter your client name that you created and your client secret. Copy and paste the access\_token that appears upon a successful Login. Click on the “Check Token” tab and paste your token in. If correct, this should return the attributes of your client.

Open the Postman REST Client and create a new tab. Change the method to POST. In the request URL, enter the following:

https://<asset\_uri>/locomotive

where <asset\_uri> is your asset service uri, which can be found using the cf env command described above. /locomotive is an endpoint for the data and will be used in the assets that are ingested. Other endpoints can be added to the asset microservice by changing the /locomotive to a different name.

Next, click on the Headers Tab. We will create 3 new headers for this Post as follows:

1. Key: Authorization Value: bearer <access\_token>
   1. <access\_token> is the value of the access token obtained above from the Starter Website for your client
2. Key: Content-Type Value: application/json
3. Key: Predix-Zone-Id Value: <Asset-Id>
   1. <Asset-Id> can be obtained by running a cf env command on your application. It will be labeled as the instanceId

Next, click on the Body tab. Change the body type to Raw and copy and paste the sample JSON Data below into the body.

[

{"id": "1111", "position\_name": "Right", "serial\_number": "1a2s", "uri": "/locomotive/CSX~123~1111~5555~s~1a2s~p~Right", "name": "CSX", "road\_number": "123", "parentId": "5555"}

,{"id": "22222", "position\_name": "Top", "serial\_number": "3d4f", "uri": "/locomotive/CSX~123~22222~6666~s~3d4f~p~Top", "name": "CSX", "road\_number": "123", "parentId": "6666"}

,{"id": "33333", "position\_name": "", "serial\_number": "5g6h", "uri": "/locomotive/CSX~678~33333~777~s~5g6h~p~", "name": "CSX", "road\_number": "678", "parentId": "777"}

,{"id": "4444", "position\_name": "", "serial\_number": "", "uri": "/locomotive/CSX~678~4444~8888~s~~p~", "name": "CSX", "road\_number": "678", "parentId": "8888"}

]

After pasting this into the body, hit submit. If successful, a 204 status should be returned.

To make sure that the data ingestion works, open a new tab in Postman and change the type to GET. Enter all the same headers as you did for the POST above and click submit. If successful, a 200 OK status will be returned and you data will display in json format in Postman.

**Deleting/Updating Assets using Postman**

To delete an asset, change the method to DELETE. For the URL, enter the following:

https://<asset\_uri>/locomotive/<part\_uri>

where <asset\_uri> is your asset service uri and <part\_uri> is the uri of the asset you would like to delete.

Next, click on the Headers Tab. We will create 3 new headers for this Post as follows:

1. Key: Authorization Value: bearer <access\_token>
   1. <access\_token> is the value of the access token obtained above from the Starter Website for your client
2. Key: Content-Type Value: application/json
3. Key: Predix-Zone-Id Value: <Asset-Id>
   1. <Asset-Id> can be obtained by running a cf env command on your application. It will be labeled as the instanceId

Click submit. If the delete is successful, a 200 OK status will be returned.

To check if your asset has been deleted, run a GET with the following url and the same headers as above:

https://<asset\_uri>/locomotive

If the delete was successful, the asset will no longer be in the output.

To update an asset, run a GET on the following url with the same headers as described above:

https://<asset\_uri>/locomotive/<part\_uri>

where <asset\_uri> is your asset service uri and <part\_uri> is the uri of the asset you would like to delete.

Copy the output of this and paste it in the body of a new tab. This new tab will have the same URL and Headers as above but will be a PUT Method. After submitting this, run a GET method to make sure that your Asset has been updated.

**Beginning to build the UI**

Clone the repository below. This is a link to the Healthcare Example UI.

https://github.build.ge.com/HealthcareITPEPredixTraining/itpe-training-view-complete

Follow the instructions in the Github Readme file in order to get the Example UI bound to your application.

After going through all of those instructions, edit the following files with all information about your microservices:

Predix-asset-service.js

Connect.js

Nginx.conf

Make sure that your Views service is bound to all users and clients. This can be done by using the Starter Kit Website that is described in the steps above.

To update user permissions, you must make groups for each of the authorities and add your user (created in the above steps) to those specific user groups. This can be done through the starter website.