User Manual
Version 1.0
April 28, 2020
EnginAir

Sponsor:
Harlan Mitchell, Honeywell
(harlan.mitchell@honeywell.com)

Mentor:
Scooter Nowak
(gn229@nau.edu)

Team Members:
Chloe Bates (ccb323@nau.edu),
Megan Mikami (mmm924@nau.edu),
Gennaro Napolitano (gennaro@nau.edu),
Ian Otto (dank@nau.edu),
Dylan Schreiner (djs554@nau.edu)

Contents

Chloe Bates ccb323@nau.edu
Megan Mikami mmm924@nau.edu
Gennaro Napolitano gennaro@nau.edu
Ian Otto dank@nau.edu
Dylan Schreiner djs554@nau.edu

1. Introduction

We are pleased that you have chosen the Connected Engine Upload Status System (CEUSS) for your business needs. CEUSS is a sophisticated and powerful graphical interface that provides convenient diagnostic tests and mapping visualizations. CEUSS has been custom designed to help automate and improve the engine maintenance process and includes key features like WiFi configuration heat mapping, engine upload failure diagnostic testing, upload location simulations, and flight tracking interfaces. The purpose of this manual is to help you install, operate, maintain, and troubleshoot CEUSS to utilize alongside with your current processes and working environments.

2. Installation

Since we use Docker as a virtualization layer on top of your host operating system, you can theoretically use any operating system that Docker supports. However, for the purposes of keeping this document clear and succinct, we will be using Ubuntu 18.04 as our example on setting up and installing Docker. Instructions for other OSes are available here: https://docs.docker.com/get-docker/.

2.1. Prerequisites

The important prerequisites for this system are as follows:

- Docker
- *docker-compose*
- Python (installed automatically through apt/yum, needs manual installation on Windows)

These systems help create and maintain a standard environment that is not susceptible to configuration changes on the host system, nor host system updates.

We also recommend running the CEUSS system on a single machine, with **at least** 8 cores and 16GB of RAM. The server-side-app processes approximately 50GB of raw data every time it correlates which can result in high memory and CPU usage.

To install these on an Ubuntu 18.04 system, type the following as the root user:

```
apt install docker.io docker-compose
```

Please note that a reboot may be required in order to ensure that Docker's system service is loaded properly.

Chloe Bates ccb323@nau.edu
Megan Mikami mmm924@nau.edu
Gennaro Napolitano gennaro@nau.edu
Ian Otto dank@nau.edu
Dylan Schreiner djs554@nau.edu

2.2. Initializing the Environment

In order to prepare the database and web application environments, we will use Docker Compose to set up a private networking bridge to securely allow communication between MongoDB, the server-side-app, and the web-app. It will also configure the web application and MongoDB when the system starts.

To start the "service" on Ubuntu 18.04, type the following as root or a non-privileged user in the docker group:

```
git clone https://github.com/enginair/server-side-app

cd server-side-app

docker-compose up -d
```

Docker will now automatically restart the MongoDB database and the web application. It also creates a virtual network adapter to allow the apps to securely communicate with each other without interrupting the host system's network operations. The web application becomes available at <host system IP>:3000. When this system is first started, the database will be empty and uninitialized. We will use the server-side-app to initialize this service.

To do this, perform the following steps as the root user or in a non-privileged account in the docker group:

```
cd <folder containing tails.json>

docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/data duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -A

docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/mega duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -t tails.json
```

The web-app and MongoDB instance are now prepared and ready for correlation. In the next section, we will explain how to import and correlate the daily upload data.

3. Configuration and Daily Operation

The backend command line application driver for all data manipulation and collection, this tool should be used on the daily to be sure the website is hosting the most accurate and up-to-date information for front end users.

The following command line options are currently supported by the application:

Short Flag	Long Flag	Parameter	Description
-t	--importTails	tails.json	Takes in the tails.json file containing all tail numbers to be tracked by the user.
-a	--importADSB	YYY-MM-DD	Imports data from ADSB API by date of YYY-MM-DD.
-c	--importCEDAS	cdas.json	Imports all CDAS Upload information in the form of JSON.
-d	--dataBaseName	dataBaseName	Name of Mongo Database used for the importation of data.
-A	--importAirports	N/A	Imports all airport data based off the most recent information provided by openflights*.

* openflights information can be found at https://openflights.org/data.html#airport

Table 1: Command Line Options Supported by Application

In practice, the command line application should run once every 24 hours to provide the best results regarding flight information. With the abundant amount of options, they should be run in the following order:

1. Import tails
2. Import Airports
3. Import ADSB
4. Import CEDAS

Running in this order will ensure that all data is collected, handled, and correlated correctly. Examples of running this in the given order are as follows:

3.1. Importing Airplane Tail Numbers

```
docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/data duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -t /data/tails.json
```

In this example, we are passing in the `-d` flag with the parameter value of the Mongo Database name, in this example case it is `honeywellDatabase`.

Then, we pass the `-t` flag with the location and file name to the `tails.json` file in which is in `/data/tails.json`.

These two combined flags import the tail number information into the provided database.

3.2. Importing Airport Information

```
docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/data duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -A
```

In this example, we are passing in the `-d` flag with the parameter value of the Mongo Database name, in this example case it is `honeywellDatabase`.

Then, we pass the `-A` flag which does not require any additional parameters.

These two combined flags will import the Airport data into the provided database.

3.3. Importing ADSB Information

```
docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/data duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -a “2016-08-01”
```

In this example, we are passing in the `-d` flag with the parameter value of the Mongo Database name, in this example case it is `honeywellDatabase`.

Then, we pass the `-a` flag with a date string formatted in “YYYY-MM-DD” in our example case it is `“2016-08-01”`.

These two combined flags will import the ADSB information into the provided database for a specific date.

3.4. Importing CEDAS Upload Information

```
docker run --network="root_default" -e START_RAM=1G -e MAX_RAM=10G -e
MONGO_HOST=mongo -v $(pwd):/data duper51/enginair-server-side-app
./bootstrap.sh -d honeywellDatabase -c /data/cedas.json
```

In this example, we are passing in the `-d` flag with the parameter value of the Mongo Database name, in this example case it is `honeywellDatabase`.

Then, we pass the `-c` flag with file location and name; in our example case we use `/data/cedas.json`.

These two combined flags will import the CEDAS upload information in json format to the provided database.

Chloe Bates ccb323@nau.edu | Megan Mikami mmm924@nau.edu | Gennaro Napolitano gennaro@nau.edu | Ian Otto dank@nau.edu | Dylan Schreiner djs554@nau.edu

3.5. Important Notes

Step 1 does not need to be run daily. It is only needed when additional tail numbers need to be tracked by CEUSS.

In a production environment you would ideally want the server-side tool to be run on a specific schedule. While not required, it is recommended to build a cron job in linux to do this on a timely basis of your choice.

More information on cron jobs can be found here:
https://code.tutsplus.com/tutorials/scheduling-tasks-with-cron-jobs--net-8800

Examples of the CEDAS.json and Tails.json formatting can be found here:
CEDAS.json - https://github.com/EnginAir/server-side-app/blob/master/CEDAS.jSon
Tails.json - https://github.com/EnginAir/server-side-app/blob/master/tails.json

4. Maintenance

Due to the nature of Docker and the way our software is architected, little to no maintenance is needed. However, a few situations may arise where maintenance must be performed.

4.1. Correlation/Web App is slow

If the correlation step or the web application's responses become slow, it may be due to many entries present in the ADSBData table on MongoDB. This can be resolved by clearing out the old entries either by using a software like "MongoDB Compass Community" or directly through the MongoDB CLI.

MongoDB Documentation about removing old records can be found here:
https://docs.mongodb.com/manual/reference/method/db.collection.remove/

5. Troubleshooting

The application may require troubleshooting from time to time. Here are some simple guidelines to follow whilst troubleshooting for the various applications.

5.1. Server-side App

The server-side app will print error messages to the console when it encounters an issue. The primary method of troubleshooting the server-side app is to check the logs for error messages to determine which stage it was at when the error was encountered.

5.2. MongoDB/Web App

MongoDB and the web application both publish their logs to the docker central logging service. To get these logs, simply type the following command into a privileged user account:

```
# docker-compose logs [mongo or enginair-web-app]
```

These should give you a pretty clear indication of what is happening. A restart may also help in some situations. You can perform this by running the following as root:

```
# docker-compose restart
```

6. Conclusion

Now that we have prepared you to operate CEUSS, we wish you many productive years with this product. We have enjoyed developing CEUSS for you and we hope it helps Honeywell ensure proper engine functionality and safe flights for years to come. From your EnginAirs, Chloe, Megan, Gennaro, Ian, and Dylan.

Chloe Bates ccb323@nau.edu
Megan Mikami mmm924@nau.edu
Gennaro Napolitano gennaro@nau.edu
Ian Otto dank@nau.edu
Dylan Schreiner djs554@nau.edu