Simulating a mission

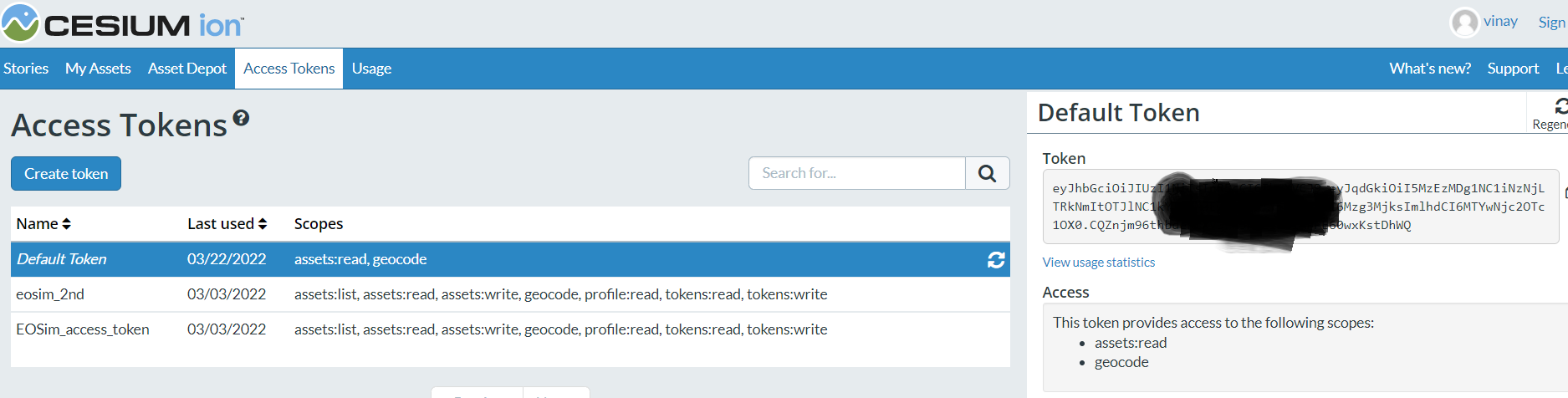
This document describes an example of configuring a basic mission, executing it and visualizing the results.

# Installation

## Pre-requisites

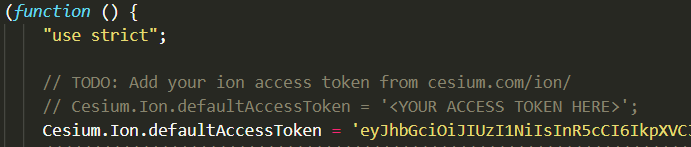
* Installation requires a Unix-like system (Ubuntu/ CentOS/ Mac). If using Windows, consider using WSL2 or a Virtual machine with Ubuntu.
* Install standard dev tools such as gcc, g++, gfortran, make and cmake. Most likely all these can be installed with a single standard essential build tools installation command.
* Install miniconda which will allow operation in a separate conda environment.
  + Conda command cheatsheet: <https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf>
  + Create a separate conda environment with python 3.8/
* Clone the InstruPy, OrbitPy and EOSim-GUI repositories to your local drive.
* Sign up and create an account in Cesium Ion <https://cesium.com/ion/signup/>. This allows access to the Earth imagery used in the animations.

## Steps

1. Install InstruPy: <https://github.com/EarthObservationSimulator/instrupy>
2. Install OrbitPy: <https://github.com/EarthObservationSimulator/orbitpy>
3. Install Eosim-GUI: <https://github.com/EarthObservationSimulator/eosim-gui>
4. Login to the Cesium Ion account and obtain your assigned access token. Copy the token.
5. Open the following file in the eosim-gui repo folder:

\eosim-gui\cesium\_app\Source\eosimApp.js

Insert the token as indicated in the file.



You are now ready to run simulations from the EOSim GUI!

# Mission Configuration

We shall configure a mission with 3 satellites in the same orbital plane at a 500km altitude, high-inclination orbit.

## Open GUI

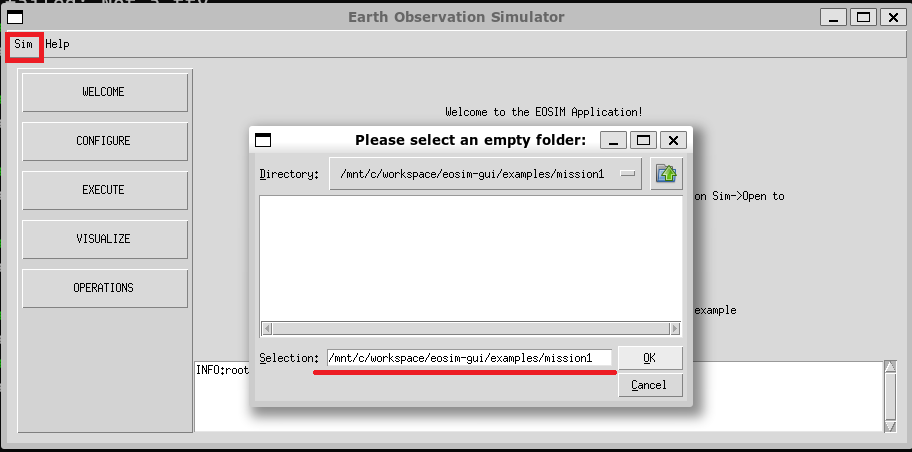
From the eosim-gui repo folder run: python bin/eosimapp.py

A GUI should pop up.

## Start a new simulation.

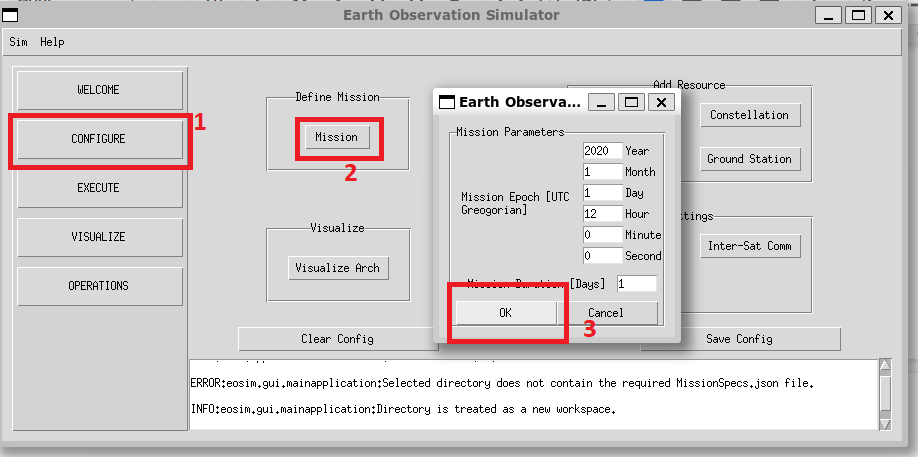
Select any empty folder to hold the mission data. In the main menu-bar click on Sim -> New

Below the eosim-gui/examples/mission1/ folder has been selected.



1. Configure mission interval.

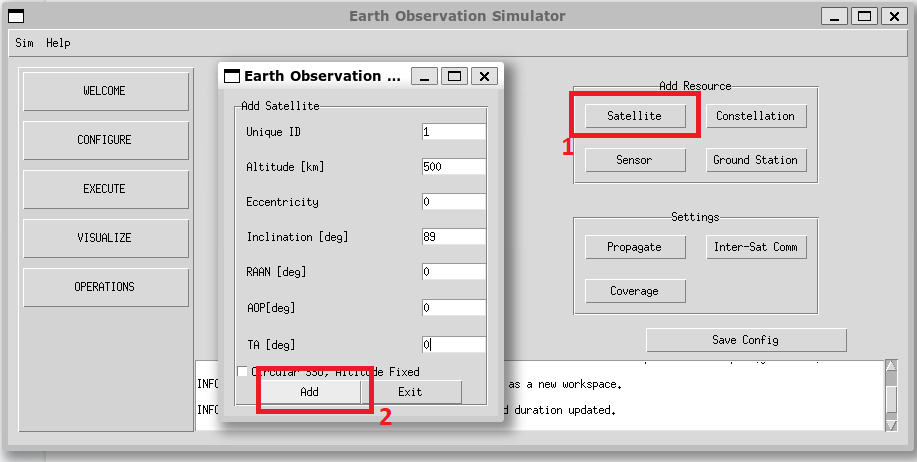
Click on CONFIGURE on the left sidebar. Click on Mission, enter the Mission Epoch and duration as UTC 1 Jan 2020, 12:00:00 hrs, and duration as 1 day.



1. Configure the orbits of the 3 satellites.

Click on Satellite. Enter the parameters for the 1st satellite as shown in below table. Repeat for the next two satellites.

|  | **Unique ID** | **Eccentricity** | **Inclination** | **RAAN** | **AOP** | **TA** |
| --- | --- | --- | --- | --- | --- | --- |
| **Sat1** | 1 | 0 | 89 | 0 | 0 | 0 |
| **Sat2** | 2 | 0 | 89 | 0 | 0 | 120 |
| **Sat3** | 3 | 0 | 89 | 0 | 0 | 240 |

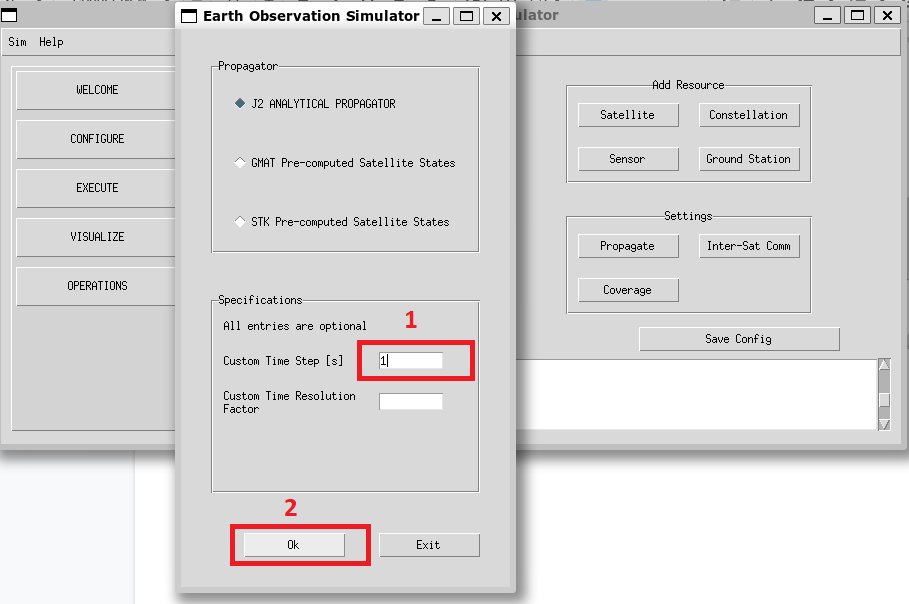


After each Satellite addition the message window at the bottom shall display the ‘Satellite added’ message.

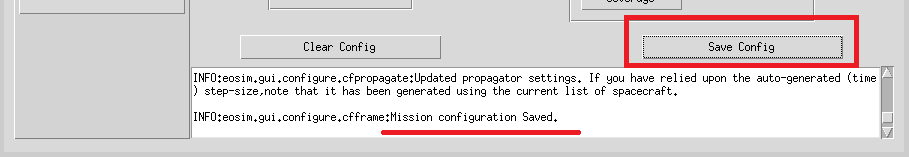


1. Configure the propagation step-size.

CLick on Propagate and set the Custom Time Step [s] field to 1. Click on OK.



1. Click on Save Config.

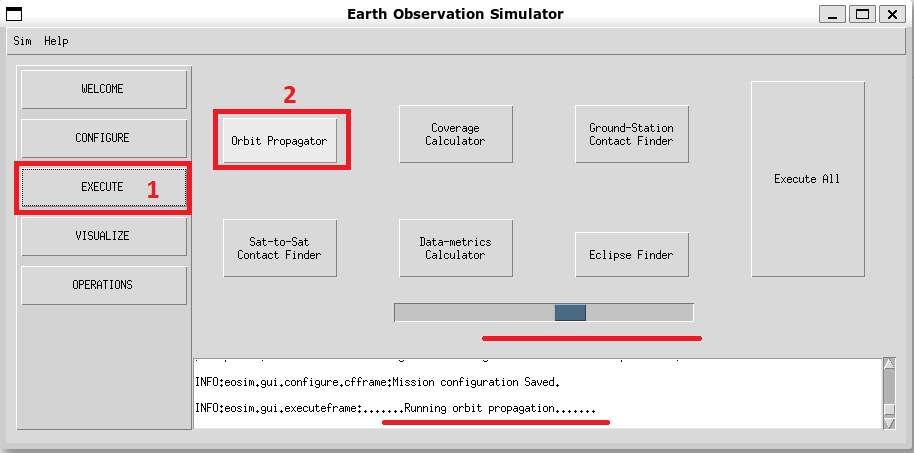


The mission configuration is now finished. A MissionSpecs.json file is saved in the user directory. At the appendix of this document the expected MissionSpecs json entry is available. Please check it matches.

# 

# Execution

We will now execute the satellite propagation. Click on Execute in the left sidebar and then click on Orbit Propagator. The (indeterminate) progress bar indicates the run. It should be completed in a few seconds.

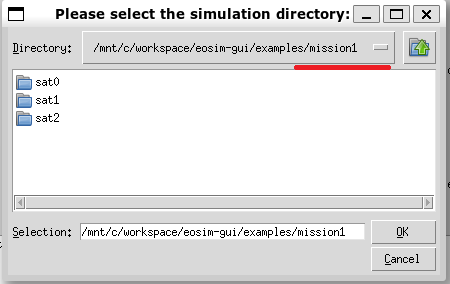
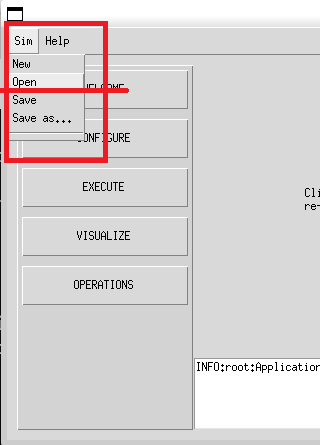


Click on Sim, and then Save.

At this point the mission has been configured and executed and saved. When revisiting the simulation, the mission can be loaded and the results visualized directly.

To try this, click on the close (the X button on top right). Restart the EOSim-GUI with the command python bin/eosimapp.py (run from the eosim-gui repo directory).

Click on Sim-> Open, and select the folder corresponding to the mission.

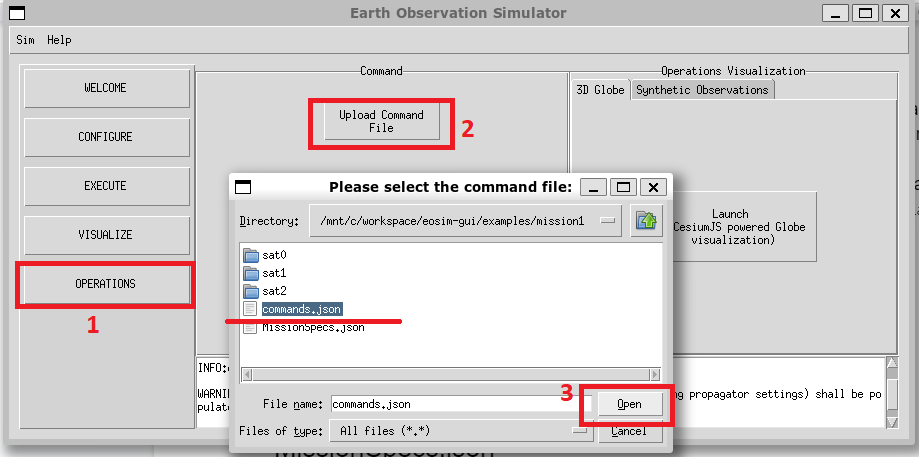


# Operations animation

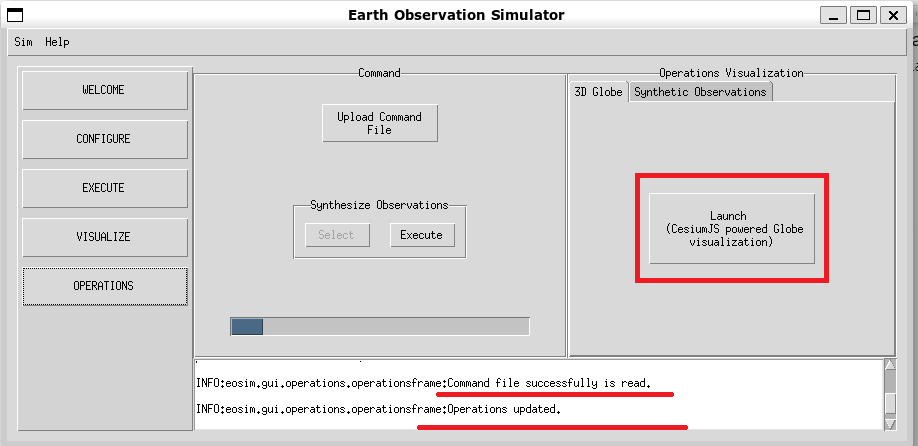
The executed mission can be visualized in various ways (XY plots, csv files, Map plots, Animation). Here we shall animate satellite operations corresponding to taking images at various ground locations.

First a commands.json file which defines the time instances and the ground-locations at which the satellite takes images needs to be defined. Please see the document ‘Commands File Schema.docx’.

Click on Operations, and then click on Upload Command File.



The Message window displays an Operations updated message. Now click on Launch CesiumJS powered Globe visualization.



# Appendix

## MissionSpecs.json

{

*"@type"*: "Mission",

*"epoch"*: {

*"@type"*: "JULIAN\_DATE\_UT1",

*"jd"*: 2458850.0

},

*"duration"*: 1.0,

*"spacecraft"*: [

{

*"name"*: null,

*"orbitState"*: {

*"date"*: {

*"@type"*: "JULIAN\_DATE\_UT1",

*"jd"*: 2458850.0

},

*"state"*: {

*"@type"*: "CARTESIAN\_EARTH\_CENTERED\_INERTIAL",

*"x"*: 6878.137,

*"y"*: 0.0,

*"z"*: 0.0,

*"vx"*: -0.0,

*"vy"*: 0.1328583318368932,

*"vz"*: 7.611448733262277

},

*"@id"*: null

},

*"spacecraftBus"*: {

*"name"*: null,

*"mass"*: null,

*"volume"*: null,

*"solarPanelConfig"*: null,

*"orientation"*: {

*"referenceFrame"*: "NADIR\_POINTING",

*"convention"*: "EULER",

*"eulerAngle1"*: 0.0,

*"eulerAngle2"*: 0.0,

*"eulerAngle3"*: 0.0,

*"eulerSeq1"*: 1,

*"eulerSeq2"*: 2,

*"eulerSeq3"*: 3,

*"@id"*: null

},

*"@id"*: null

},

*"instrument"*: null,

*"@id"*: "1"

},

{

*"name"*: null,

*"orbitState"*: {

*"date"*: {

*"@type"*: "JULIAN\_DATE\_UT1",

*"jd"*: 2458850.0

},

*"state"*: {

*"@type"*: "CARTESIAN\_EARTH\_CENTERED\_INERTIAL",

*"x"*: -3439.0684999999985,

*"y"*: 103.95772623766837,

*"z"*: 5955.734147376984,

*"vx"*: -6.592712064587972,

*"vy"*: -0.06642916591844657,

*"vz"*: -3.8057243666311367

},

*"@id"*: null

},

*"spacecraftBus"*: {

*"name"*: null,

*"mass"*: null,

*"volume"*: null,

*"solarPanelConfig"*: null,

*"orientation"*: {

*"referenceFrame"*: "NADIR\_POINTING",

*"convention"*: "EULER",

*"eulerAngle1"*: 0.0,

*"eulerAngle2"*: 0.0,

*"eulerAngle3"*: 0.0,

*"eulerSeq1"*: 1,

*"eulerSeq2"*: 2,

*"eulerSeq3"*: 3,

*"@id"*: null

},

*"@id"*: null

},

*"instrument"*: null,

*"@id"*: "2"

},

{

*"name"*: null,

*"orbitState"*: {

*"date"*: {

*"@type"*: "JULIAN\_DATE\_UT1",

*"jd"*: 2458850.0

},

*"state"*: {

*"@type"*: "CARTESIAN\_EARTH\_CENTERED\_INERTIAL",

*"x"*: -3439.068500000003,

*"y"*: -103.95772623766833,

*"z"*: -5955.734147376981,

*"vx"*: 6.592712064587969,

*"vy"*: -0.06642916591844665,

*"vz"*: -3.8057243666311416

},

*"@id"*: null

},

*"spacecraftBus"*: {

*"name"*: null,

*"mass"*: null,

*"volume"*: null,

*"solarPanelConfig"*: null,

*"orientation"*: {

*"referenceFrame"*: "NADIR\_POINTING",

*"convention"*: "EULER",

*"eulerAngle1"*: 0.0,

*"eulerAngle2"*: 0.0,

*"eulerAngle3"*: 0.0,

*"eulerSeq1"*: 1,

*"eulerSeq2"*: 2,

*"eulerSeq3"*: 3,

*"@id"*: null

},

*"@id"*: null

},

*"instrument"*: null,

*"@id"*: "3"

}

],

*"propagator"*: {

*"@type"*: "J2 ANALYTICAL PROPAGATOR",

*"stepSize"*: 1.0,

*"@id"*: null

},

*"grid"*: null,

*"groundStation"*: null,

*"outputInfo"*: null,

*"settings"*: {

*"@type"*: "Settings",

*"outDir"*: "/mnt/c/workspace/eosim-gui/examples/mission1/",

*"coverageType"*: null,

*"propTimeResFactor"*: 0.25,

*"gridResFactor"*: 0.9,

*"opaqueAtmosHeight"*: 0.0,

*"midAccessOnly"*: false,

*"@id"*: null

},

*"@id"*: null

}