In Your Orbit

User Manual

Version 1.0

Cloud-Based Planetary Ephemerides

United States Geological Survey

Mentor: Scott Larocca

Austin Carlile
Noah Schwartz
Minuka Trikawalagoda
Nicholas Gonzalez

The purpose of this design document is to explicitly detail the installation and maintenance of our software solution

4/30/2025

Table of Contents

1. Introduction

We are pleased that you have chosen Cloud-Based Planetary Ephemerides for your business needs. Cloud-Based Planetary Ephemerides is a powerful system for automatic ISD generation, caching and retrieval, that has been custom-designed to meet your needs. Some of the key highlights include:

- Web Service for clients to send label files to and receive ISDs back
- Caching server for memoization of generated ISDs
- Dockerfile, as well as DynamoDB and ECS CloudFormation files for AWS set up

The purpose of this user manual is to help you, the client, successfully install, administer, and maintain the Cloud-Based Planetary Ephemerides product in your actual business context going forward. We aim to make sure that you can profit from our product for many years to come!

2. Installation

To run the Web Service locally, you must first install conda-forge miniforge, ISIS3, DynamoDB and clone the Web Service Git Repository. Before beginning the Web Service, set up an Amazon AWS account and an IAM User, then generate Access Keys and give the IAM User DynamoDB permissions. After completing these steps, create environment variables in your linux environment for your AWS Access Keys. Now the Web Service is ready to run.

Here are the steps to set up the necessary environment for operation of the Web Service:

1. Begin with a linux based operating system

2. Install Miniforge, instructions can be found at:

   https://github.com/conda-forge/miniforge

3. Install ISIS and set environment variables, instructions can be found out:

   https://astrogeology.usgs.gov/docs/how-to-guides/environment-setup-and-maintenance/installing-isis-via-anaconda/

   NOTE: Make sure to install base ISIS data:

   downloadIsisData base $ISISDATA

   NOTE: Make sure to also install ISIS data for any missions you are using label files from.

   For example, if you are going to use viking1 label files, run:

   downloadIsisData viking1 $ISISDATA

   Or for all missions:

   downloadIsisData all $ISISDATA

4. Navigate to a directory for the Web Service and clone github repo:

   sudo apt install git && git clone https://github.com/AustinCarlile/Cloud-Based-Planetary-Ephemerides-Deliverable.git

5. Install necessary dependencies:

   conda install fastapi && conda install boto3 && conda install brotli && conda install pvl && conda install uvicorn && pip install mkl && conda install requests

To set up your Amazon AWS account for deploying the DynamoDB instance:

1. Set up a AWS Account:

   https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html

2. Set up an IAM User account:

   https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html

3. Create IAM User Access Key:

   https://docs.aws.amazon.com/keyspaces/latest/devguide/create.keypair.html

4. Give IAM User AmazonDynamoDBFullAccess permissions:

   https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-attach-detach.html

   https://docs.aws.amazon.com/aws-managed-policy/latest/reference/AmazonDynamoDBFullAccess.html

5. Deploy DynamoDB CloudFormation stack:

   https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html

   The DynamoDB CloudFormation file (dynamo.yml) is found in the cloudformation directory of the GitHub repository.

To set up the Web Service:

1. Set your IAM User access keys and region to environment variables in your .bashrc:

   AWS_ACCESS_KEY_ID="*IAM USER ACCESS KEY*"; export AWS_ACCESS_KEY_ID

   AWS_SECRET_ACCESS_KEY="*IAM USER SECRET ACCESS KEY*"; export AWS_SECRET_ACCESS_KEY

   AWS_REGION="*AWS REGION RESOURCE IS HOSTED IN*"; export AWS_REGION

   ## WARNING: DO NOT UPLOAD A DOCKER IMAGE TO A PUBLIC REGISTRY WITH YOUR ACCESS KEYS AS ENVIRONMENT VARIABLES

   For possible strategies to deal with this, refer to the STS information in the Configuration and Daily Operation section.

2. After setting environment variables in .bashrc, run:

   exec bash

   then:

   mamba activate isis

3. Run uvicorn command to start Web Service:

   uvicorn isdAPI:app --reload

The Web Service is now running and listening for HTTP requests on 127.0.0.1:8000.

3. Configuration and Daily Operation

Make sure to download all isis data needed for mission label files. If you use “web=true” in the spiceinit section of the web service, it will cause a segmentation fault and core dump. The web service will still be operational but will save junk data to memory.

If you already have the AWS CLI set up, you can use the ~/.aws/credentials file instead of setting the access keys and region to environment variables:

[default]

aws_access_key_id=*IAM USER ACCESS KEY*

aws_secret_access_key=*IAM USER SECRET ACCESS KEY*

aws_region=*AWS REGION RESOURCE IS HOSTED IN*

If you want to to dynamically get AWS credentials, you can use AWS STS:

https://docs.aws.amazon.com/code-library/latest/ug/python_3_sts_code_examples.html

https://botocore.amazonaws.com/v1/documentation/api/latest/reference/services/sts.html

The Web Service expects a Brotli compressed byte stream of a label file and will return a Brotli compressed byte stream of an ISD. Any client that interacts with the Web Service will need to be able to send and receive Brotli compressed byte streams. A Tester Client can be found in the webservice/tests directory of the GitHub Repository, along with a README on how to use it. This Tester Client can be used to test and debug the Web Service, as well as a reference for creating or adapting any other clients.

Configuration for end-user:

- Interact programmatically with the REST API to access endpoints

Workflow for end-user:

- Submit a request by sending the web service a .lbl file for the target image
- Receive a freshly generated ISD or an ISD from the cache

4. Maintenance

The Web Service automatically generates a print.prt log file that chronicles all ISD generation actions that have taken place. Periodically, the user may want to erase or delete this file to prevent its size from growing too large.

The Web Service will also automatically create a “temp.json” file when generating an ISD, this is overwritten each time and will not grow in size. The user may want to automatically remove this file if they do not want to have it in memory at all times.

The isd_generate function inside the Web Service uses the “-v” verbose flag, which will print all ISD generation operations to the Web Services’ console. If you would like to keep the Web Service as lightweight as possible, consider removing this flag.

5. Troubleshooting

The following are a list of possible issues you may encounter when deploying and running the Web Service and DynamoDB instance, as well as their potential solutions.

Improper File Format:

- Ensure the label file being submitted is formatted correctly

Improper installation of ISIS:

- A lack of drivers or sufficient ISIS data will result in failures during ISD generation

Incorrect Environment:

- Make sure directories and paths are correct when attempting to generate an ISD

Error deploying CloudFormation file:

- Check the error given during the CloudFormation deployment in the AWS console. Make sure that the user has permissions required.

Web Service Segmentation fault:

- Make sure that you have downloaded the base ISIS data:

```
downloadIsisData base $ISISDATA
```

- Make sure that you have downloaded the mission data for any mission label file you send the Web Service.

- For example, if you are going to use viking1 label files, run:

    downloadIsisData viking1 $ISISDATA

- Or for all missions:

    downloadIsisData all $ISISDATA

    NOTE: A segmentation fault will happen if you add "web=true" to the spiceinit command in isdAPI.py

Brotli decompress failure:

- Check whether this error is coming from the Web Service or the client.
- If from the Web Service, the client is likely sending an incorrect byte stream.
- If from the Client, the Web Service is likely returning a non-ISD HTTP response, which will not be Brotli compressed. This is either due to an ISD generation failure or an incorrect cache retrieval or sending of an ISD.

6. Conclusion

This concludes the user manual for Cloud-Based Planetary Ephemerides. We wish you many happy years of productive use of our solution. Thank you for allowing us to develop this product for you.

With best wishes from your Cloud-Based Planetary Ephemerides developers:

Austin Carlile, Nicholas Gonzalez, Noah Schwartz and Minuka Trikawalagoda