Oh, The Places The International Space Station May Go!

A Web API for ISS Enthusiasts and Beyond

Kelechi Emeruwa

March 10th, 2023

1.1 Introduction

The National Aeronautics and Space Administration (NASA) and four other space agencies launched the International Space Station (ISS) on the 20th of November, 1998. Today, the ISS is one of the finest research laboratories orbiting in space. Moreover, tracking the ISS has never been easier thanks to the work of the Trajectory Operations and Planning Officer (TOPO) flight controllers in the Mission Control Center at NASA's Johnson Space Center. However, the tracking data, called an orbit ephemeris message (OEM), can be tedious to parse through. Here's a snippet of what some of the entries in the OEM (TXT formatted version) look like:

Figure 1: Orbital Ephemeris Message in TXT Format

2023-02-15T12:00:00.000 -4788.368507507620 1403.549622371260
-4613.109479300690 -4.47317640532645 -5.44388258946684 2.99705738521092

2023-02-15T12:04:00.000 -5675.021705065900 61.910987386751
-3734.576449237840 -2.87004030254429 -5.66832649751615 4.27967238757376

2023-02-15T12:08:00.000 -6148.993248504040 -1284.195507156520
-2583.725124493340 -1.05503300582525 -5.48063337615216 5.25228914094105

2023-02-15T12:12:00.000 -6175.021895469930 -2536.723661139980
-1244.142324035630 0.83992906678481 -4.89307346066615 5.84252973499924

2023-02-15T12:16:00.000 -5750.560812798620 -3604.169888126130
186.445271666768 2.67584228156696 -3.94766617813937 6.00579498886775

As shown above, the data from the OEM is difficult to understand and tedious to parse through. Thus, it would be convenient if one

could receive neat and succinct information about the ISS for use in other computer programs. Hence, the ISS API aims to enable users to query information about the ISS that would otherwise be tedious and enduring to process independently. Thanks to the ISS API, developers can effectively and efficiently interact with the ISS trajectory data in real time. The following section gives a brief overview of the ISS trajectory data, and how users can interpret/query data from the ISS API.

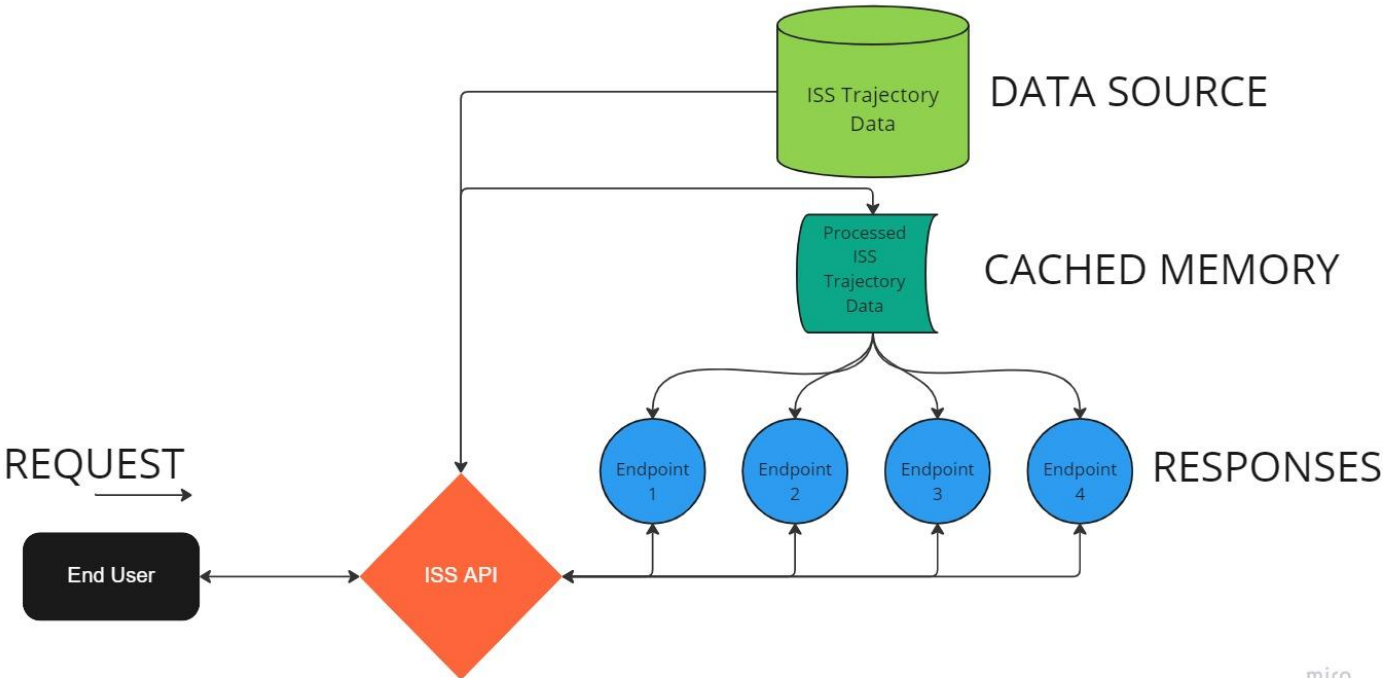
1.2 ISS Trajectory Data

The OEM (TXT format) begins with header information, metadata, and comments with additional information about the ISS (such as the ISS's mass (kg) and drag coefficient (m^2), etc). Following the comments is an extensive list of state vectors (as shown in Figure 1). The positional and velocity vectors are measured in the Mean of J2000 (J2k) reference frame. The ISS API parses the OEM txt data into a list of dictionaries, where each dictionary contains the state vectors for a given timestamp (including the timestamp). Moreover, the state vectors are listed at four-minute intervals for 15 days with two-second intervals during reboosts (translation maneuvers). The OEM is updated approximately three times every week, and can be accessed on the ISS Trajectory website.

1.3 System Design

Below demonstrates how the ISS API connects end-users with information about the ISS:

Figure 2: ISS API Flowchart:



In summary, the ISS API serves an intermediary service that enables users to interact with the ISS trajectory data in a more meaningful way. The ISS API sends a GET request to the ISS Trajectory Data website and utilizes string manipulation techniques to parse the .txt file into a list of dictionaries. The ISS API relies on Flask, to set the endpoints that users can query. Flask is a Python library that serves as a microweb framework that helps developers easily make web APIs. The diagram above only lists four endpoints, however, ISS API actually has several more. The following section lists each endpoint users can query along with additional details about each endpoint.

1.4 Routes

Currently, users can query 12 endpoints with the ISS API. Namely:

Table 1: ISS API Endpoints Table

Route	Method	Returned Data
/	GET	The entire data set (list of dictionaries) *- Includes optional parameters "limit" (positive int) to truncate results and "offset" (positive int) to change the starting position at which the data is returned*
/comment	GET	Returns comments from the ISS trajectory data source file
/header	GET	Returns header information from the ISS trajectory data source file
/metadata	GET	Returns metadata from the ISS trajectory data source file
/now	GET	Returns latitude, longitude, altitude, and geoposition of the ISS for an epoch that is nearest in time
/epochs	GET	All Epochs in the data set (list of strings) *- Includes optional parameters "limit" (positive int) to truncate results and "offset" (positive int) to*

		change the starting position at which the data is returned
/epochs/<epoch>	GET	State vectors for a specific Epoch from the data set (list of one dictionary) <epoch> Takes string inputs only.
/epochs/<epoch>/location	GET	Returns latitude, longitude, altitude, and geoposition of the ISS for a given epoch
/help	GET	Help text (string; not html friendly) that briefly describes each route
/convert?units	PUT	Converts data from 'SI' to 'USCS' and vice versa. The data is originally in SI units.
/delete-data	DELTE	Deletes all stored data in the application
/post-data	POST	Reloads flask application with the original ISS trajectory data

1.5 Installation (With Docker)

Users can install and run the ISS API locally in three ways. Namely, pulling the ISS API Docker image from the Docker Hub, building the ISS API Docker image manually, and lastly, cloning the ISS API repository and installing all its necessary dependencies.

If you're unfamiliar with Docker, you may be surprised to know that using Docker is the most convenient installation method over the others. This is because Docker allows developers to share and run identical programs (regardless of variations in hardware specifications) that may otherwise not run on the hardware of one's local computer. This is accomplished via the building of Docker images and the running of Docker containers.

If those terms are unfamiliar, then, briefly, a Docker image can be understood as the "blueprint" of a given isolated virtual environment. For example, it specifies libraries and packages to include in container instances. A Docker container is an instance of a Docker image (similar to how objects are instances of classes). Moreover, Docker containers allow developers to enter and interact (via the command terminal) in this isolated virtual environment pre-defined by the Docker image.

2.1 Usage

The ISS API is best suited for queries via a terminal or software program. This is because the data returns valid JSON responses that can be easily parsed by other programs. Below are example queries and responses produced by the ISS API:

Table 2: ISS API Example Queries and Responses:

Route	**Returned Data**

http://localhost:5000?limit=1&offset=12	{
	"X": "-5675.021705065900",
	"X_Dot": "-2.87004030254429",
	"Y": "61.910987386751",
	"Y_Dot": "-5.66832649751615",
	"Z": "-3734.576449237840",
	"Z_Dot": "4.27967238757376",
	"epoch": "2023-02-15T12:04:00.000"
	}

http://localhost:5000/header	{
	"CCSDS_OEM_VERS": "2.0",
	"CREATION_DATE": "2023-03-04T04:34:04.606",
	"ORIGINATOR": "NASA/JSC/FOD/TOPO"
	}

http://localhost:5000/comment	["Source: This file was produced by the TOPO office within FOD at JSC.", "Units are in kg and m^2", "MASS=473413.00", "DRAG_ARE...",]

In the first row, the 12th state vector is queried. In which, users can obtain positional, and velocity data about the ISS at the given timestamp (epoch). It's important to note that the units of the positional and velocity data are, by default, kilometers, and kilometers per second respectively. In addition, epochs listed for each dictionary are in Coordinated Universal Time (UTC). As mentioned in Section 1.2, the third row includes a truncated response of the comments returned after performing a GET request to the "/comment" endpoint.

2.1.1 Performance and Calculations

The ISS API uses Binary Search to locate state vectors of a given epoch which greatly improves the speed of the service. Further, The speed of each dictionary of state vectors is calculated using the Euclidean Norm of the"X_Dot", "Y_Dot" and "Z_Dot" vectors.

When users query the "/now" route, the longitude and latitude values are sent to GeoPy, an external python library, to determine the associated geolocation location of the ISS. It's important to note that sometimes the ISS is above an ocean in which case GeoPy cannot determine the geolocation. The following section is intended to address the ethical responsibilities users should aim to adopt while using this and other web API's.

2.2 Ethics and Final Remarks

While the ISS API has the potential to be of great value to professional developers and enthusiasts, we must also recognize the ethical concerns using web applications such as these can raise.

Firstly, users of APIs like the ISS API must be made aware of sources from which the APIs retrieve their information. In doing so, users can validate the reliability of the data returned by said web API.

Secondly, developers of web APIs should consider gathering data from sources open to sharing their data with the public. For example, scraping personal information from websites and making that information easily accessible to users via a web API should not be encouraged. Not only does that expose people to spam and malicious attacks, but it is also discouraged due to the potentially invasive nature of web scraping. In other words, developers should be cautious when relying on web-scraping programs to collect data in their APIs, as it can be invasive and potentially harmful. Data presented in TXT, XML, and other recognized data storing formats are generally more safe.

Lastly, as alluded to previously, it's important to recognize the impact one's web API may have on the surrounding community. In this case, the ISS API produces a generally positive outcome for the community because tracking the ISS more effectively can make

way for more sophisticated web applications, and even inspire others to learn more about space travel. In conclusion, developers and users must continue to reflect on the power of web APIs (and technology as a whole) to use them for good and avoid the dangers of using technology irresponsibly or, worse, for evil.

References

Keeter, Bill. "Spot the Station." *NASA*, NASA, 27 July 2021, https://spotthestation.nasa.gov/trajectory_data.cfm.