Accessing the PI Web API with Python

Introduction

- **PI Web API** is a powerful tool for interacting with PI System programmatically.
- **Python** is a versatile language widely used for automation and data analysis.

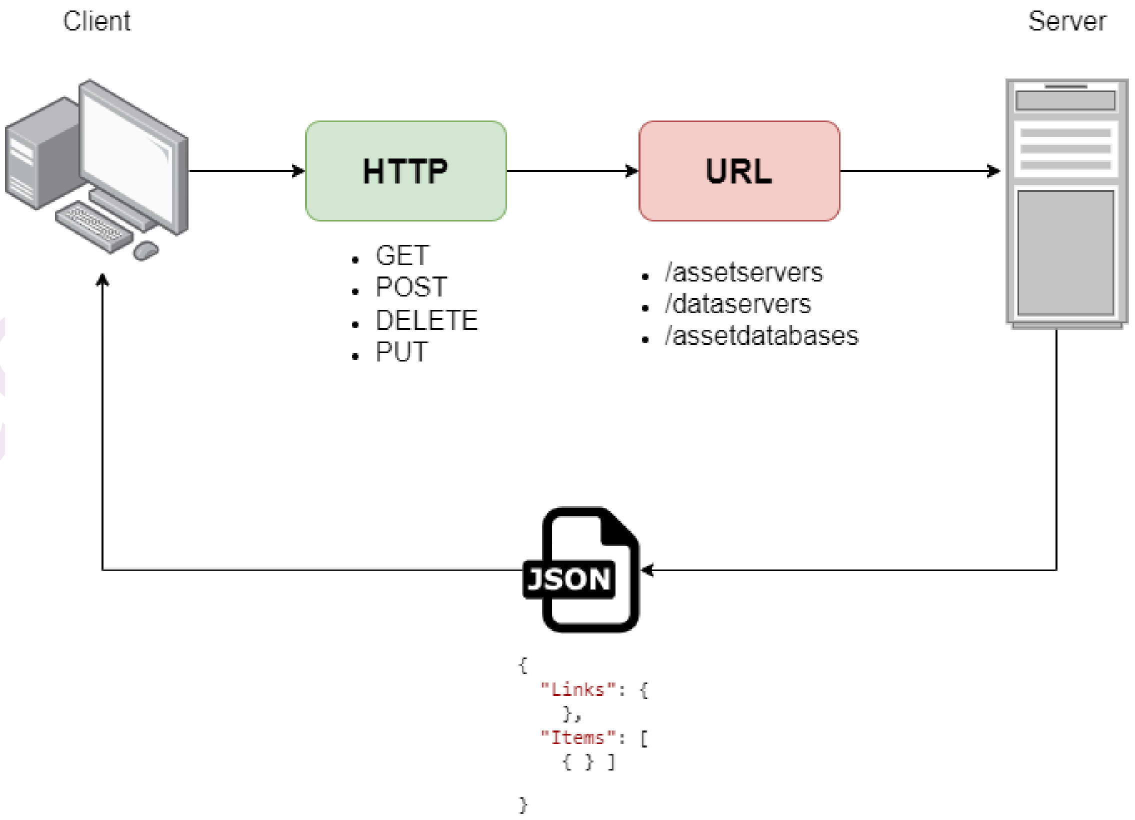
This presentation will guide you on how to access and manipulate data using the PI Web API with Python. This process can be useful for integrating PI data with other systems and automating tasks.

PI Web API

- The PI Web API is a RESTful interface for accessing PI System data (AF & DA) over HTTPS.

- It enables read and write access for client applications, fostering custom integrations.

- The API offers a modern, flexible approach to leverage your data for analytics, visualizations, and more.



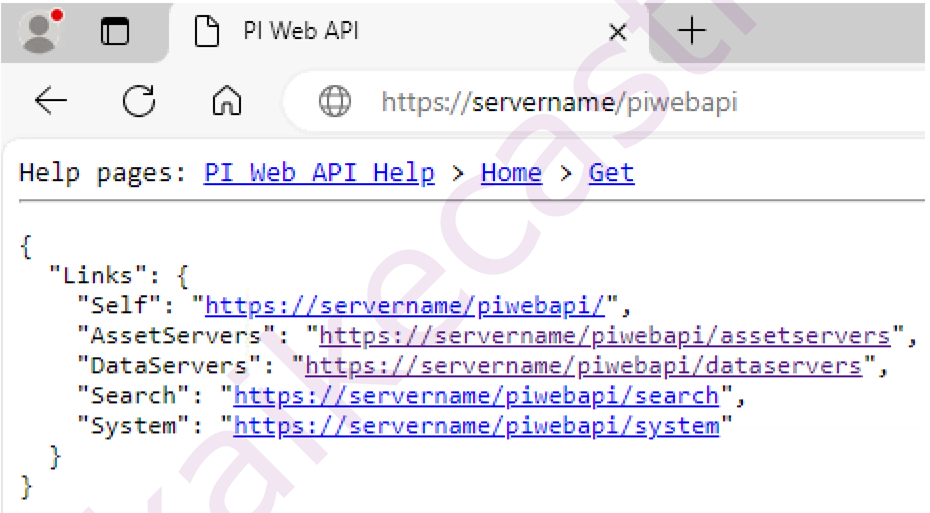
Why Use the PI Web API?

- **Secure and Efficient Data Access:** The PI Web API uses HTTPS for secure communication and supports OAuth 2.0 for authentication. Its stateless and resource-oriented design ensures efficient and scalable data retrieval and manipulation.
- **Easy Integration and Interoperability:** By using JSON for data interchange, the PI Web API facilitates seamless integration with various programming languages and systems, enhancing interoperability across applications..
- **Dynamic and Discoverable Resources:** The API leverages hypermedia links (HATEOAS) to navigate resources, allowing client applications to dynamically discover and interact with resources without hard-coding URLs..
- **Advanced Data Handling Capabilities:** It provides comprehensive read and write access to PI and AF data, supports batch operations, ad-hoc stream sets, and caches AF data for performance optimization, accommodating complex data retrieval and processing scenarios.

Key Principles of PI Web API

- **Stateless:** Each request is independent, simplifying client development.
- **Resource-oriented:** Interaction revolves around resources representing PI System objects (elements, attributes, points).
- **Navigable:** Links connect resources, enabling seamless navigation and discovery.
- **Uses HTTP Verbs:** GET, POST, PUT, PATCH, DELETE for standard CRUD operations.
- **JSON Format:** Data exchanged in easy-to-parse JSON, compatible with various programming languages.

Home Web API



Help pages: PI Web API Help > Home > Get

```
{
  "Links": {
    "Self": "https://servername/piwebapi/",
    "AssetServers": "https://servername/piwebapi/assetservers",
    "DataServers": "https://servername/piwebapi/dataservers",
    "Search": "https://servername/piwebapi/search",
    "System": "https://servername/piwebapi/system"
  }
}
```

Getting Started

Prerequisites:

- **PI System Access:** Ensure you have access to a PI System with the PI Web API enabled.
- **Python Installation:** Have Python installed on your system.
- **Required Libraries:** Install the `requests` library: `pip install requests`

Code Example

```python
import requests

# Define the PI Web API URL and the endpoint for fetching data
pi_web_api_url = "https://your-pi-server/piwebapi"
data_endpoint = "/elements/{webId}/elements"

# Specify your PI Web API credentials
username = "your_username"
password = "your_password"

# Make the request to fetch data
response = requests.get(
    f"{pi_web_api_url}{data_endpoint}",
    auth=(username, password)
)

# Check if the request was successful
if response.status_code == 200:
    data = response.json()
    print(data)
else:
    print(f"Failed to fetch data. Status code: {response.status_code}")
```