PI Vision API

Overview

The VisionApi class provides methods to interact with the PI Vision API, specifically for retrieving and updating PI Vision display data. This class uses Kerberos authentication to securely access the PI Vision API.

Prerequisites

- Python 3.x
- Required Python libraries: requests, urllib3, requests_kerberos, json
- Kerberos authentication configured on the client machine

Installation

To install the required Python libraries, you can use pip:

```
pip install requests urllib3 requests_kerberos
```


Usage

Initialization

Create an instance of the VisionApi class by providing the base URL of the PI Vision API endpoint.

```
from vision_api import VisionApi

url_endpoint = "https://<servername>/PIVision/Utility/api/v1"
api = VisionApi(url_endpoint)
```

Retrieving PI Vision Data

To retrieve data from the PI Vision API, use the get_PiVisionApi method with the specific API endpoint.

```
displayId = "0000"
custom_url = "displays/{displayId}/export"
response_get = api.get_PiVisionApi(custom_url)
```

Updating PI Vision Data

To update data on the PI Vision API, use the update_PiVisionApi method with the specific API endpoint and the data to be updated. This example change the owner of display.

```
response_get["Display"]["Owner"] = "user_owner"
response_get["DuplicateDisplayWriteBehavior"] = "Overwrite"
custom_url = "displays"
response_update = api.update_PiVisionApi(custom_url, response_get)
```

Methods

get_PiVisionApi(custom_url)

Retrieves data from the PI Vision API.

- Parameters:
    - custom_url (str): The specific API endpoint to retrieve data from.
      Returns:
    - dict: The JSON response from the API or an empty dictionary if an error occurs.
      update_PiVisionApi(custom_url, body)
      Updates data on the PI Vision API.
- Parameters:
    - custom_url (str): The specific API endpoint to update data on.
      body (dict): The data to be updated in JSON format.
- Returns:
    - dict: The JSON response from the API or an empty dictionary if an error occurs.

update_PiVisionApi(custom_url, body)

Updates data on the PI Vision API.

- Parameters:
    - custom_url (str): The specific API endpoint to update data on.
    - body (dict): The data to be updated in JSON format.
- Returns:
    - dict: The JSON response from the API or an empty dictionary if an error occurs.

Error Handling

The VisionApi class handles various HTTP status codes and logs appropriate error messages:

- 200: Success
- 400: Invalid request
- 401: Unauthorized
- 403: Forbidden.
- 404: Resource not found

- 500: Internal server error.
- 503: Service unavailable

Example 200 response body:

```
{
  "Items": [
    {
      "Id": 78103,
      "Name": "Display1",
      "Owner": "Domain\\user"
    },
    {
      "Id": 78127,
      "Name": "Display2",
      "Owner": "Domain\\user"
    }
  ],
  "HasMore": false
}
```

Example 400 response body:

```
{
  "Message": "The request is invalid.",
  "ModelState": {
    "FolderId": [
      "folderId parameter should be a positive integer."
    ]
  }
}
```

Notes

The code disables SSL warnings using urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning). This is generally not recommended for production environments.

Ensure that the Kerberos authentication is correctly set up on the client machine to avoid authentication issues.

Alternatively, you can use Basic Authentication if Kerberos is not available. To use Basic Auth, modify the _request method in the VisionApi class to include auth=HTTPBasicAuth('username', 'password') in the requests.request call. Note that Basic Auth sends the username and password in the request header, which is less secure compared to Kerberos.

Example of using Basic Authentication

Replace the _request method in the VisionApi class with the following code:

```
response = requests.request(
            method, api_url, auth=HTTPBasicAuth('username', 'password'),
headers=headers,
            json=body, verify=False
        )
```