Glens API Specifications

Overview

Glens provides an Open API for integrating multi-software clients to GLens server. All communication between GLens Central Server and client are all managed through REST HTTP-based API. Any software client complying with the specified API can upload the data to the Central Glens Server. **The API doc is minimal and simplified with details on the Upload and Delayed Upload only for RSPCB**.

Contacts

Glens provides clients for various analysers which meets these requirements. Glens software team can assist any industry or analyser supplier to enable them to comply with this common format. For any queries please send the request to glens@knowledgelens.com or call at +91 9916025302

Basic Organization of API

http://<ipaddress:port>/GLensServer (The specific URL will be provided to you based on the pre-deployment checklist)

Resource	Description	Route	Request type
Data upload	This is for uploading data to the central server from the client. Any approved client software with proper data format can upload data to the server using this api. The server supports different client softwares which communicates through this api	/upload /delayedUpload	POST

This feature make the system capable of supporting multiple client software approved by Pollution Control Boards which can communicate with the central server.

The central server is exposing a Representational State Transfer (REST) based Application Programming Interface (API) for the client software.

The bandwidth required for this communication is very minimal. The communication will easily happen through the existing Broadband/LAN/GPRS/WiFi/Dongle connectivity.

Data Upload

The standard response format is described below. Any approved client software adhering to the proper format can send data to the central server using this API.

Data Upload Format

The zip file upload to the server will be multipart/form-data format. The data should be sent in zip format. The uploaded zip file will have two files, namely 1. Data File, 2. Metadata File. The zip file should be uploaded to the server in the proper format.

Request Details

Upload data to central server

This method uploads data to the server.

http://ipaddress:port/GLensServer/upload OR http://ipaddress:port/GLensServer/delayedUpload

Path: **upload or delayedUpload**
Method: **POST**
Parameters: The file to be uploaded should be send as the parameter.

Returns: Response JSON which contains the status as either **success** or **failure**

The zip file has to be posted as JSON Encoded POST format. The encoding has to be in the format {'file': <fileObject>}. For the **purpose of debugging** the POST request, the file uploading request will have the following message

mime_multipart.header.content-disposition form-data;name="file";filename="<filename.zip>"\r\n

<filename.zip> should be the zip file which is uploaded.

Note: upload URL will take only data that is captured from the analyser during the last poll frequency defined by regulator. Anything delayed should be uploaded to delayedUpload URL

If the upload is success, the following response will be obtained.

```
{
  "status": "Success",
  "serverConfigLastUpdatedTime": "<time>",
  "siteConfigUpdateFlag": "<Flag>",
  "siteCalibrationUpdateFlag": "<Flag>",

  "serverCalibrationLastUpdatedTime": "<Flag>",

  "statusMessage": "file uploaded successfully."
}
```

Where the **<time>** is the last updated time of server configurations and **<Flag>** is a Boolean value depending upon whether the site configuration is updated or not.

Flag can have values “True” or “False”. The responses from the services are listed below.

Eg:

If the upload is a success the following response will be obtained.

```
{
  "status": "Success",
  "serverConfigLastUpdatedTime": "",
  "siteConfigUpdateFlag": "False",
  "statusMessage": "1 files uploaded successfully.",
  "serverCalibrationLastUpdatedTime": ""
}
```

For Delayed Upload File:

```
{
  "status": "Success",
  "serverConfigLastUpdatedTime": "",
  "siteConfigUpdateFlag": "False",
  "statusMessage": "1 delayed files uploaded successfully.",
  "serverCalibrationLastUpdatedTime": ""
}
```

If the upload is a failure the following response will be obtained.

```
{
  "status": "Failed",
  "serverConfigLastUpdatedTime": "",
  "siteConfigUpdateFlag": "False",
  "statusMessage": "No attachment found. Please upload zip file containing
the data.Upload will not be completed without attachment",

  "serverCalibrationLastUpdatedTime": ""

}
```

For Delayed Upload File:

```
{

  "status": "Failed",

  "serverConfigLastUpdatedTime": "",

  "siteConfigUpdateFlag": "False",

  "statusMessage": "No attachment found. Please upload delayed zip file
containing the data.Delayed Upload will not be completed without
attachment",

  "serverCalibrationLastUpdatedTime": ""

}
```

If GET method is used you will get the following response:

```
"GET method not supported for uploading, use POST method"

"GET method not supported for delayed file uploading, use POST method"
```