Open API for Online Continuous Monitoring System

April 6, 2016

Version No 1.0 Dated 6th April 2016

Overview

This API document is the reference document for Industries to transmit the Continuous Emission and Effluent Monitoring system. Any authenticated software client residing in the Industry site will be accepted for transmitting data to Maharashtra Central Server Software. All communication between Central Server and acquisition clients (at Industry site) are all managed through HTTP-based REST API. All the API are authenticated. Any approved software client complying with the specified open API can upload the data to the Central Server.

Supported Operations in current API Version 1.0

The following are the operations supported in Version 1.0 of the Open API. All clients should support full integration with all these operations.

- Real Time Data Upload
- Delayed Data Upload → For data that is transmitted due to internet connectivity issues
- Analyser Diagnostic Fetch

Key Concepts

The following are the key concepts to be followed while working with the Open API

- Site Key: Unique Key provided to each site for data transmission. This key should not be shared with any other site. This will be used for data encryption.
- Site ID: Unique Site ID identifying the specific industry
- Monitoring ID: Each Site has multiple monitoring stations. Each monitoring station will be assigned a unique monitoring ID relative to the Site
- Analyser ID: Each analyser make and model will be assigned a unique Analyser ID
- Parameter ID: Each monitored parameter will have a common unified ID across all industries

Client Side Software Requirement

Each client software implementing the API should also comply with “Client Side Software Requirement published by MPCB“

Key API Requirements

- Each site client software has to collect the data from the analyser based on the poll frequency defined. Ideal frequency for data sampling from analyser is 10 second. Data

transmission to the Central Server should be at 1 minute frequency. The raw data and linearized data should be transmitted to the server along with the data quality code and captured timestamp. All calculations at the Central Server for Exceedance measurement will be performed at 15 Minute Average.

- The transmitted data should be encrypted with site key provided. All API request data transfer should be using a REST Service over HTTP protocol.
- In-case of any communication failure or any delayed data transmitted beyond a period 15 minutes, site software should store the data the locally and upload to the Delayed Data Upload URL and not to the Real Time upload URL. This is to ensure that the delayed data is captured separately at the Central Server and can be tracked for any integrity issues.
- All client should transmit data captured directly from the analyser from the site location. Any data transmission from different location will be rejected by the server.
- All the requests from the client to the server should be authenticated requests only. Any unauthenticated requests will be discarded or not processed.
- All transmission to be made directly from the industry location. Any transmission from non-industry location will be rejected by the server during processing of the data.

Basic Organization of API

http://220.225.78.13/mpcb

Resource	Description	Route	Request type
Real Time Data Upload / Delayed Data upload	This is for uploading data to the central server from the client. Any authenticated client with proper credentials can upload data to the server using this api. Only real time data (delay of max 15 min) will be accepted through /realtimeupload URL and any delayed data should be uploaded using /delayedUpload URL	/realtimeUpload /delayedUpload	POST
Diagnostic Upload service	The client software should using this URL for uploading the diagnostic information including any analyser status and alarms of the analyser as per the analyser make and model.	/uploadDiagnosticInfo	POST

Open API for Online Continuous Monitoring System

April 6, 2016

Authentication Mechanism for the API

Each API request header should have the following information

A. Timestamp
B. Authorization → Encrypted data from the site which has the authentication digest. Each request should send authentication digest with the following encrypted data
   - site_key → Unique Site Key provided by the Maharashtra Pollution Board for each site for authentication
   - api_version_id → api version used for communication (The current value will be "ver_1.0")
   - time_stamp_data → Timestamp when the data was encrypted

The authentication digest is decrypted using the Site Private Key. The timestamp is ensured to be not more than 15 minutes (configurable) from the current timestamp. The api version is verified against the registered software api with the Central Server Software. This ensures the data is encrypted just before transmission and the client program have access to Site Private Key and the current registered api version. The registered software version will be updated from Central Server Software time to time and hence is not depend on client software version.

Data Upload

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

Data Upload Format

The GLens API supports 2 different types of data format for data upload. The data upload follows an ISO-7168 format zip file or a simplified delimited or fixed width file 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 Data file should be encrypted using the Site Private Key. The zip file should be uploaded to the server with proper authentication using the key. Else the response with HTTP 401 with "Authentication Failure" will be returned.

The metadata file will specify the file formats (ISO-7168, CSV, FixedWidth) etc. and the data file should comply with the same. This gives flexibility to support different file formats based on the analyser or client software capability.

However, all files has to follow the basic guidelines

1. Data should encrypted
2. File should zipped
3. Metadata file should provide the file specification and format
4. Header should have the encryption digest for decryption of the data

Request Details

Upload data to Central Server

This method uploads data to the server. The requests will be authenticated and hence should have the authentication header as described in section "Authentication Mechanism for the API"

http://220.225.78.13/mpcb/realtimeUpload OR

Open API for Online Continuous Monitoring System

April 6, 2016

http://220.225.78.13/mpcb/delayedUpload

Path: **realtimeUpload 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**

Note: realtimeUpload 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>",
 "ConfigurationDownloadFlag": "<Flag>",
 "ConfigurationUpdateFlag": "<Flag>",
 "RemoteCalibrationUpdateFlag ": "<Flag>",
 "DiagnosticUpdateFlag": "<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”

Eg:


```
{
 "status": "Success",
 "serverConfigLastUpdatedTime": "2015-02-24T13:21:19Z",
  "ConfigurationDownloadFlag": "True",
 "ConfigurationUpdateFlag": "False",
 "RemoteCalibrationUpdateFlag ": "True",
 "DiagnosticUpdateFlag": "False",
 "statusMessage": "file uploaded successfully. "
}
```

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

```
{
  "status": "Failed",
  "statusMessage": "No files were uploaded."
}
```

Upload Diagnostic Information

The diagnostic information should be uploaded once every 15 minute minimum to ascertain the health of the system. All alarms, alerts generated by the analyser should be transmitted using this interface.

http://220.225.78.13/mpcb/uploadDiagnisticsInfo

Path: **uploadDiagnosticInfo**

Method: **POST**

Parameter: The diagnostic information of the Site in the json format. For details on the specific analyser please contact mpcb at the support contact numbers.

Returns: The response json contains, success in case of success or failure message in case of failure. The diagnostics json will be an array of key value pair with the corresponding category associated to the key.

Request body:

```
{
  "Command": "DiagnosticFetch",
  "SiteDetails": {
    "siteName": <SiteName>,
    "siteLabel": <SiteLabel>,
    "siteConfigLastUpdatedTime": <SiteConfigUpdatedLastTime>,
    "siteId": <site id>,
    "monitoringId": <monitoring id>,
      "customparameters" :{}
  },
  "CollectorDetails":[ {
    "CollectorType": <>,
    "CollectorName": <>,
    "ConfiguredChannels": <>,
    "PollingStep": <polling step>,
    "ChecksumStatusBit": <checksum bit>,
     "Address": <address>,
    "HeartBeat": <heartbeat>,
    "DataFormatBits": "00",
    "Port": <port>,
    "CommunicationTimeOut": <communication bit>
    "customparameters" :{}
 }],
```

```
  "diagnosticJson": [{"analyserId":<analyser-id>,"parameterName":"",
diagnostics":[{"key":<key>, "value":<value>,"category":<category>}]}]

}
```

Response for the Request will be

Success status

```
{

  "status": "Success",

  "diagnosticUpdateStatus": "Received Site diagnostics successfully"

}
```

Failure status

```
{

  "status": "Failed",

  "diagnosticUpdateStatus": "Failed to receive Site diagnostics. Please
retry"

}
```