1 Madhya Pradesh PCB API Version No 1.0 dated 1st April 2019

1.1 Overview

This Open API document will be used for integrating multi-software clients to Madhya Pradesh Pollution Control Board 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.

1.2 Supported Operations in Version 1.0

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

- Real Time Data Upload
- Delayed Data Upload
- Remote Analyser Calibration Check
- Analyser Diagnostic Fetch

1.3 Key Concepts

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

- 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

1.4 Client Side Software Requirement

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

1.5 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.
- The captured data should be transmitted to the Server immediately after encrypting the data with the digital private key. This digital private key should be kept safe and shouldn’t be tampered with or disclosed to others. The Client Software should provide a mechanism for generating industry specific digital signature to establish the data connectivity with Central Server Software. Industry should procure the Digital Signature as soon as possible, provide

the details of the Digital Signature and use that for data encryption to ensure authentic tamper proof data transmission.

- The transmitted data should be encrypted zipped data in ISO-7168. All API request data transfer should be using a REST Service over HTTP/HTTPS protocol.
- The client site software should wait for successful upload and also read subsequent instructions (Remote calibration, Configuration update, Diagnostics information etc.) from the Central Server Software
- On receiving instructions on Remote calibrations, the site client software should invoke the Remote Calibration update services to download the corresponding configurations.
- In-case of any communication failure or any delayed data transmitted beyond a period 2 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.

Basic Organization of API

http://<ipaddress:port>/MPPCBOnlineServer

Resource	Description	Route	Request type
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 2 min) will be accepted through /realtimeupload URL and any delayed data should be uploaded using /delayedUpload URL	/realtimeUpload /delayedUpload	POST
Calibration download service	When the RemoteCalibrationUpdateFlag is set to “True”, the client software should using this URL for downloading the configuration required for calibration. The remote calibration data and sequence should be initiated by local client	/getcalibrationconfig	POST
Update Calibration Start Status	Whenever site has started calibration this api has to be called.	/ updateCalibrationStart	POST
Update Calibration Stop Status	Whenever site has completed calibration this API has to be called.	/ updateCalibrationCompleted	POST
Diagnostic Upload service	When the DiagnosticUpdateFlag is set to “True”, the client software should using this URL for uploading	/uploadDiagnosticInfo	POST

	the diagnostic information including any internal state of the analyser as per the analyser make and model.		

2 Encryption and Authentication Mechanism for Madhya Pradesh State Pollution Control Board

This section of the document explains the Encryption and Authentication mechanism to be followed while transmitting the data. This document should be used by the software programmer used for developing the MPPCB API for Client software to Madhya Pradesh State Pollution Control Board.

Key principles used for encryption and authentication is as follows

1. Data in ISO-7168 format is encrypted using AES Algorithm to preserve data integrity during transmission
2. Data is transmitted over HTTP or HTTPS and the header for the HTTP/HTTPS request will have the authentication and signature specific to the Industry using the RSA Algorithm.

2.1 Key Details Required

- The data transmitted for real time upload and delayed upload will follow the ISO-7168 format. ISO-7168 format is adopted to ensure that the API is Open format with respect to data exchange. ISO-7168 is usually used for Air Quality Data but can be customized to include the Effluent Data as well. The specific customization will be provided during Industry Registration.
- Similar to the data format, the authentication and encryption mechanism is standard open format. Some of the parameters like padding, block size, Initialization Vectors, Key Size etc, has been customized to provide robust security from a long-term perspective. Periodically, these parameters will be changed as we perceive security threats. So the client software should be developed with flexibility to ensure that encryption scheme can be altered with minimal configuration changes.
- The specific details required for AES Encryption and RSA Security Keys will be provided to the Industry once the registration of the Industry is completed on the MPPCB website.

2.2 Data Upload

- The data upload follows an ISO-7168 format zip file.
- 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 Metadata file will specify the file formats (ISO-7168) etc.
- The data file should comply with the same.

1. ISO-7168 Data should have encrypted using the AES Algorithm using AES Data Encryption Key
2. File should zipped
3. Metadata file should provide the file specification and format

2.3 Header request should follow the RSA encryption principles and will have the following content

A. Timestamp
B. site_key → Site ID
C. Authorization → The authorization data consist of RSA Encrypted data of the following fields in order separated by the delimiter. The authorization data will be encrypted using the **MPPCB Server RSA Public Key** provided
   - site_key → Site ID provided by the MPPCB for each site
   - software_version_id → API version set by the MPPCB. Current value is "ver1.0"
   - time_stamp_data → Timestamp when the data was encrypted

D Signature (RSA)

The signature will be encrypting the authorization data (site_key, software_version_id, time_stamp with proper delimiter) using the Digital Signature (Private RSA Key) of the Industry to ensure that the data is sent by the industry. In-case digital signature is not available industry can use the Site-specific RSA Private Key generated and provided by MPPCB. In-case of using the Digital Signature, it is requested for Industry to provide the Public Key for the Industry to MPPCB.

The timestamp is ensured to be not more than 2 minutes from the NTP timestamp. The software version is verified against the registered software version 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 software version. The registered software version will be updated from Central Server Software time to time and hence is not depend on client software version.

3 API Request and Response Details

3.1 General Guidelines for uploading data to Central Server

All requests will be authenticated and hence should have the authentication header as described in section "**Encryption and Authentication Mechanism for Madhya Pradesh State Pollution Control Board"**

All request should have proper timestamp and the timestamp shouldn't have a deviation of more than 2 minutes than NTP Timestamp.

3.2 Realtime and Delayed Data upload to Central Server

Data will be uploaded in real time to the following URL

http://ipaddress:port/MPPCBOnlineServer/realtimeUpload

Any delayed data (due to internet failure or other reasons) will be uploaded in to the following URL

http://ipaddress:port/ MPPCBOnlineServer/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. Timestamp on data should be within 2 minutes of NTP. Anything delayed should be uploaded to delayedUpload URL

3.3 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 remote calibration or diagnostics upload is required or not.

Flag can have values "True" or "False"

Eg:

```
{
 "status": "Success",
 "serverConfigLastUpdatedTime": "2015-02-24T13:21:19Z",
 "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."
}
```

3.4 Remote Calibration Service

Each of the SOX, NOX Analysers are required to perform Remote Calibration based on requirement from MPPCB Central Server. To perform remote calibration, during the data upload process, MPPCB will send the flag for remote calibration. Based on the flag, each software client has to request to download the Calibration Configuration and perform calibration and upload the data to MPPCB.

This method download the configuration required for calibration.

http://ipaddress:port/MPPCBOnlineServer/getCalibrationConfig

Path: **getCalibrationConfig**

Method: **POST**

Parameter: The site id, monitoring id, CalibrationType will be passed as the parameter. CalibrationType will be “scheduled” when a schedule is submitted to client or “immediate” if an immediate request for calibration is required. A separate request should be sent for all monitoring stations so that each monitoring station specific calibration details is provided by server.

Returns: The response json contains the configuration required for calibration. This will provide all calibration details required to be scheduled or immediate for all analyzers within the monitoring station.

Request body:

```
{
  "siteId": <site-id>,

  "monitoringid": <monitor-id>,

  "CalibrationType": "Scheduled" or "Immediate"

}
```

Response provided by the Server will have the following fields. If any analyser maker needs any additional fields for performing, remote calibration, this can be discussed with MPPCB Online Monitoring team and can use "customparameters" tag in the json

The configuration details has the sequence for calibrations, the required parameters for calibrations and the schedule for the calibrations.

RESPONSE

```
{
  "status": "Success",
  "calibration": {
    "calibratorName": <calibrator-name>,
    "sequence": [
      {
        "function": <function name>,
        "duration_secs": <duration in seconds>,
        "gas": <gas>,
        "value": "0",
        "delay": <delay in minutes>,
        "sequenceName": <sequence name>,
        "duration": <duration in minute>,
        "type": <type of calibration>,
        "unit": <unit of gas>
      } ...................

    ],
    "siteName": <site name>,
    "monitoringType": <monitoring type>,
    "frequency": <frequency>,
    "analyzerId": <analyser id>,
    "parameterId": <parameter id>,
    "remoteCalibrationId": <remote calibration id>,
    "parameterName": "SO2",
    "cycleUnit": "1",
    "total_duration": <total duration>,
    "frequencyDay": <day>,
    "siteId": <site id>,
    "startTime": {
      "date": <date>,
      "time": <time>
    },
    "executeImmediate": "True",
    "day": <day>,
    "cycle": <cycle>,
    "frequencyTime": <frequency time>,
    "calibratorId": <calibration id>,
    "monitoringUnit": <monitoring unit>,
    "value": "",
    "channelNumber": <channel number>,
    "analyzerType": <analyser type>,
    "endTime": {
      "date": <date>,
      "time": <time>
    },
    "remoteCalibrationName": <remote calibration name>,
    "analyzerName": <analyser name>
  },
  "serverCalibrationLastUpdatedTime": <serverCalibrationLastUpdatedTime>,
  "siteCalibrationLastUpdatedTime": <siteCalibrationLastUpdatedTime>,
  "lastCalibratedOn": <lastCalibratedOn>,
  "siteId": <siteid>
}
```

<u>Failure</u>

```
{

   "status": "Failed. Calibration configuration not available"

}
```

3.5 Calibration Start Acknowledgement

Whenever the site client software has received the calibration sequence and has started the calibration on the analyser, the calibration start acknowledgement has to be provided to MPPCB Central Server to ensure that server doesn't request for calibration configuration again.

This method gives the status of calibration.
http://ipaddress:port/MPPCBOnlineServer/updateCalibrationStart

Path: **updateCalibrationStart**

Method: **POST**

Parameter: The site id and monitoring id will be passed as the parameter.

Returns: The response json contains, success in case of success or failure message in case of failure.

Request body:

```
{

  "siteId": <site-id>,

  "monitoringid": <monitor-id>,

  "analyzerid": <analyzer-id>,

  "CalibrationType": "Scheduled" or "Immediate",

  "CalibrationStatus": "Completed" or "Failed" or "Aborted"


}
```

<u>Response to Calibration Update Received by Client Software</u>

Success Response

```
{
  "status": "Success",
  "siteId": <site-id>,

  "monitoringid": <monitor-id>,

  "analyzerid": <analyzer-id>,

  "calibrationUpdateStatus": "Acknowledged Calibration Start"
}
```

Failure response

```
{
  "status": "Failed",
  "siteId": <site-id>,

  "monitoringid": <monitor-id>,

  "analyzerid": <analyzer-id>,

  "calibrationUpdateStatus": "Failed to acknowledge calibration start"
}
```

3.6 Calibration Completed Acknowledgement

Whenever the site client software has received the calibration sequence and has started the calibration on the analyser, the calibration start acknowledgement has to be provided to MPPCB Central Server to ensure that server doesn't request for calibration configuration again.

This method gives the status of calibration.
http://ipaddress:port/MPPCBOnlineServer/updateCalibrationCompleted

Path: **updateCalibrationCompleted**

Method: **POST**

Parameter: The site id and monitoring id will be passed as the parameter.

Returns: The response json contains, success in case of success or failure message in case of failure.

Request body:

```
{

  "siteId": <site-id>,

  "monitoringid": <monitor-id>,

  "analyzerid": <analyzer-id>,

  "CalibrationType": "Scheduled" or "Immediate",
  "CalibrationStatus": "Completed" or "Failed" or "Aborted"

}
```

Response to Calibration Update Received by Client Software

Success Response

```
{
  "status": "Success",
  "calibrationUpdateStatus": "Acknowledged Calibration Start"
}
```

Failure response

```
{
  "status": "Failed",
  "calibrationUpdateStatus": "Failed to acknowledge calibration completed"
}
```

3.7 Fetch Diagnostic Information From Client

This method will be invoked by the client to upload the current diagnostic information in the analyser to the Central Server Software when the DiagnosticUpdateFlag is set to “True”. Every time MPPCB wants Diagnostic Information, this has to be transmitted by the client.

http://ipaddress:port/MPPCBOnlineServer/uploadDiagnisticsInfo

Path: **uploadDiagnosticInfo**

Method: **POST**

Parameter: The diagnostic information of the Site in the json format

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>,
    "siteConfigLastUpdatedTime": <SiteConfigUpdatedLastTime>,
    "siteId": <site id>,
    "monitoringId": <monitoring id>,
    "analyzerId": <analyser id>,
    "customparameters" :{}
  },
  "diagnosticJson": [{"analyserId":<analyser-id>,"parameterName":"",
diagnostics":[{"key":<key>, "value":<value>,"category":<category>}]}]
}
```

<u>Response for the Request will be</u>

<u>Success status</u>

```
{
  "status": "Success",
  "siteId": <site-id>,
  "monitoringid": <monitor-id>,
  "analyzerid": <analyzer-id>,
  "diagnosticUpdateStatus": "Received Site diagnostics successfully"
}
```

<u>Failure status</u>

```
{
  "status": "Failed",
  "siteId": <site-id>,
  "monitoringid": <monitor-id>,
  "analyzerid": <analyzer-id>,
  "diagnosticUpdateStatus": "Failed to receive Site diagnostics. Please
retry"
}
```