Cloud Monitoring API

Contents

[I. Change Log 2](#_Toc440635832)

[II. Foreword 3](#_Toc440635833)

[III. API Overview 3](#_Toc440635834)

[IV. API Functions Reference 3](#_Toc440635835)

[1. fetchBarGraphData 3](#_Toc440635836)

[2. fetchOwnerUpdate 4](#_Toc440635837)

# Change Log

|  |  |
| --- | --- |
| **Revision** | **Comments** |
| 1.0 – 2015/01/15 | Initial Version |
|  |  |

# Foreword

This document is a quick reference to some of the Cloud monitoring interface that can be called from a browser or any other application.

Please note that the API is not meant for high rate of polling and any application found to poll the server at a rate faster than once a minute will be blocked from accessing the Cloud server. There is no data that requires faster access than once every 5 minutes so a polling rate of once every 5 minutes is what is recommended.

# API Overview

The API allows an application to directly query the Cloud Monitoring Database and returns results over a given period. The more data is requested, the slower the query so it is important not to overload the server by requesting data that won’t be directly useful. It is also recommended to cache or save the data on the client side so that the same data does not get requested all the time.

This API was mainly developed for the Cloud Monitoring website (cloud.chiliconpower.com) and as such may not fit the need of other applications.

# API Functions Reference

Note the URL in the call syntax must be preceded by the installation URL (e.g.: cloud.chiliconpower.com/installation/{installation\_hash}).

The response data is a JSON object.

## fetchBarGraphData

This interface should be called once a day at most. In order to get updates on the latest power production, use fetchOwnerUpdate since it is not as taxing on the server.

**Call syntax:**

url: "/ajax/fetchBarGraphData",

data: {

'today': '2015-01-15’

}

**Response:**

[todayData, weekData, monthData, yearData, lifetimeEnergy, currentProduction]

The one input parameter is ‘today’ which is the date for which the response data is requested.

The output is a list of 6 objects as follows:

todayData: array of 288 installation power points for the given day at a 5 min interval

weekData: array of energy harvest for the 7 days inclusive of and preceding the input date

monthData: array of energy harvest for the 31 days inclusive of and preceding the input date

yearData: array of energy harvest for the 12 months inclusive of and preceding the input date

lifetimeEnergy: lifetime energy harvest (in kWh) for this installation (independent of input date)

currentProduction: last power reading sent from the Gateway of that installation in kW (independent of input date)

## fetchOwnerUpdate

**Call syntax:**

url: "/ajax/fetchOwnerUpdate",

data: {

'today': '2015-01-15’

}

**Response:**

[todayData, lifetimeEnergy, currentProduction]

The one input parameter is ‘today’ which is the date for which the response data is requested.

The output is a list of 6 objects as follows:

todayData: array of 288 installation power points for the given day at a 5 min interval

lifetimeEnergy: lifetime energy harvest (in kWh) for this installation (independent of input date)

currentProduction: last power reading sent from the Gateway of that installation in kW (independent of input date)

## fetchGatewayLastCommunicationTime

This interface will return a Python deltatime object of the time delta between now and the last communication exchange with the Gateway.

**Call syntax:**

url: "/ajax/fetchGatewayLastCommunicationTime",

**Response:**

[lastDate]

There is no input parameter.

The output is a Python timedelta object between the current date and the last date the cloud server received a ping from the Gateway. Note that the Gateway will relay telemetry data at regular intervals of around 5 minutes maximum when the microinverters are producing. When they stop producing, the Gateway pings the server at regular intervals but not for sending data but only to fetch commands from the Cloud server if any. Under that regime, the Cloud server updates the last communication time of the Gateway only every hour.

lastDate: timedelta object

## fetchData

This interface retrieves all the data points of a single attribute (say power, temperature, …) of all the microinverters for a given day or week (7 day period). It is very time consuming to run this request on the Cloud Database since it generates a large number of I/Os. So this should be used with parsimony.

**Call syntax:**

url: "/ajax/fetchData",

data: {

'selection': ‘p\_out\_avg’,

'lastDay': '2015-01-15’,

'timeSpan': 7

}

**Response:**

[seriesData]

There are three input parameters for this call.

The first one called 'selection' determines the attribute that needs to be returned. The most common one would be ‘p\_out\_avg’ which is the output power generated by the microinverter as measured by itself (self-reported). A list of available values is in appendix A.

The second parameter 'lastDay' is the date of the last day for which the data is requested (which would be the same as the first day if there is only one day – which is the most common way to call this method).

The last input parameter 'timeSpan' is the number of days to fetch from the server. This can only be 1 or 7 days.

The output is an object that is a list of data points. Each data point being represented by a list of three values:

[dateValue, attributeValue, data.global\_inverter\_id\_id]

Where dateValue is the date in seconds since Jan 1st, 2000

attributeValue: is the actual value of the attribute selected.

globalInverterId: is the ID of the inverter that sent this data point to the Gateway which then relayed it to the Cloud Database

# Appendix A: list of microinverter attributes

<option value='p\_out\_avg'>Average Output Power (W)</option>

<option value='adc\_vpv\_last\_sum'>PV Voltage (V)</option>

<option value='i\_pv'>PV Current (A)</option>

<option value='flash\_page\_total\_energy'>Total Energy Produced (kWh)</option>

<option value='sensor\_temp\_1'>DC/DC Temperature (°C)</option>

<option value='sensor\_temp\_3'>Board Temperature (°C)</option>

<option value='v\_grid\_sum\_sq\_ema'>Grid Voltage (Vrms)</option>

<option value='v\_grid\_thd'>Grid Voltage THD (%)</option>

<option value='i\_grid\_thd\_slow\_ema'>Grid Current THD (%)</option>

<option value='cycle\_duration\_jitter\_average'>Cycle Duration Jitter (µs)</option>

<option value='max\_cycle\_duration\_error\_abs'>Cycle Duration Max Error (µs)</option>

<option value='flash\_page\_decoded\_packet\_cnt'>Decoded Packet Count</option>

<option value='flash\_page\_comm\_check\_sum\_error'>Checksum Error Count</option>

<option value='flash\_page\_operating\_time'>Operating Time (Hrs)</option>

<option value='flash\_page\_firmware\_version\_id'>Firmware Version</option>