DataServer

Contains the functions for data collection server backend

> Author: Aubrey Thenell, Daniel Schroede, Parker Bruni.
>
> Source: Source/dataServer.js, line 1

Methods

`(inner) addEntry(meter, body, serialAddress) → {Promise}`

Function to add a data entry to a database. Checks for duplicate first. Conditions: - Called whenever POST request comes in

Parameters:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>meter</code></td><td>Object</td><td>Contains elements of the source meter
<em>Properties</em>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>_id</code></td><td>String</td><td>Contains the ID of the meter as it is in the database</td></tr>
<tr><td><code>name</code></td><td>String</td><td>Contains the public facing name of the meter as it is in the database</td></tr>
<tr><td><code>building</code></td><td>String</td><td>Contains the building that the meter is associated with as it is in the database</td></tr>
</table>
</td></tr>
<tr><td><code>body</code></td><td>Object</td><td>Contains the POST request body with all extraneous pieces cut</td></tr>
<tr><td><code>serialAddress</code></td><td>Object</td><td>Contains the POST request body that can access the serial and address information</td></tr>
</table>

> Source: Source/dataServer.js, line 310

Returns:

Type `Promise`

`(inner) addMeter(meter) → {Promise.<Meter>}`

Function that is called when the server gets a POST request from a meter that is not in our database already. Only called when a matching meter cannot be located in the database

Parameters:

Name	Type	Description
`meter`	Object	Contains a POST request body that is slightly trimmed

> Source: Source/dataServer.js, line 283

Returns:

newmeter

Type `Promise.<Meter>`

(inner) checkMeterTimestamps() → {void}

Helper function to see when the last time a meter posted Called once per day and Sends email if meter hasn't checked in for over a day but under two

> Source: Source/dataServer.js, line 172

Fires:

- event:emailAlert

Returns:

Type `void`

(inner) checkUsage()

Function used to find and alert during high usage spikes Conditions - Only sends email if Average for day is greater than the average for week plus a standard deviation and the meter has more than 100 entries

> Source: Source/dataServer.js, line 214

Fires:

- event:emailAlert

(inner) emailAlert(email) → {void}

Sends out alert to users (admins). Specifically when A meter hasn't checked in over a day. Also when a meter reports unusually high usage.

Parameters:

Name	Type	Description

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>email</code></td><td>Object</td><td>Contains email contents to be sent.
<em>Properties</em>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><code>subject</code></td><td>String</td><td>Contains the email's subject.</td></tr>
<tr><td><code>body</code></td><td>String</td><td>Contains the email's body.</td></tr>
</table></td></tr>
</table>

Source: Source/dataServer.js, line 119

Returns:

Type `void`

`(inner) post->receiveXML(path, xml) → {Promise}`

Handles POST requests received in XML format Receives POST requests, converts from XML to JSON

Parameters:

Name	Type	Description
`path`	string	Express path
`xml`	Object	Contains the XML data coming in.

Source: Source/dataServer.js, line 48

Fires:

- event:checkUsage
- event:checkMeterTimestamps
- event:addMeter
- event:addEntry

Returns:

Type `Promise`

Documentation generated by JSDoc 3.5.5 on Tue Jun 12 2018 15:29:29 GMT-0700 (Pacific Daylight Time) using the Minami theme.