Collector – Server Transport Documentation

Introduction:

This document describes how the collector node sends data to the server and database. See the system map if lost.

Sending data:

Setup: Make sure there is a file named '.pass' in the same directory as the script. This should contain the passcode to the server (be the same as .../keys/.htaccepted_passcode) and be a numerical value.

To send data, the collect should run the send_data function provided by submit_data.py. This take the parameters: Latitude, Longitude, Sensor Type, Sensor Value, Timestamp. An example is as follows:

```
send_data(-45.7745, 170.6, ”gate”, True, 1462600277)
```

Note: all time is stored as UNIX time, and formatted human readable on final display.

The system works as follows:

This is a 2 stage system. Open <domain>/resource/datahandling/submit.php?stage=1 on the server, this is the request stage, and the server script will return a public key. If this is successful you can proceed to the next stage. For this stage, encrypt the passcode with the public key using the following formula:

```
Cypher = (passcode^e) % n; where n is the returned value before the comma, and e is the returned value after the comma.
```

Then again open the submit.php script on the server, with these options:

stage=2 ; indicates the second step of the process.

pass=cypher lat=<latitude> lng=<longidute> type=<sensor type>

value=<sensor value> time=<timestamp>

Note: All data is send in plain text, the encryption is only to ensure the collector has authentication.

Note: The second stage must be submitted with the timeout period of the server, this is 50 by default.

Receiving Data:

Setup: submit.php should be located in the www directory /resources/datahandling. There should also be a keys directory within datahandling, with the files '.htprivatePt1.key' '.htprivatePt2.key' '.httimeout.dat'. Apache needs write permission for these files. Additionally, it needs read permission for keys/.htaccepted_passcode and this file needs to contain the same numerical string as collectors.

Once everything is setup, the script will drop .httechnical.log, showing the time of submission, data submitted, ip of submitter, and passcode used. This should be using for debugging purposes only and cannot be shown online.

It will also create status.log, which is a human readable log file, with processed data, and warnings. This is used by status.php to give an overall status of the system. It is also used by the notifying system, to send SMS and/or tweets.

To add a custom entry to the status log file, edit submit.php. Find the following line:

```
function add_status_entry($lat, $lng, $type, $value, $time){
```

Below this function, create your own function that creates a string of what you want your log entry to be. This can use any of the data passed to the add_status_entry() function, as well as any older data via a database query (see db docs).

Then add your function to the add_status_entry() function as following:

$entry .= <function name>($lat, $lng, $type, $value, $time);

Different entry functions are appended atop each other, for this to work, please end in a free space and keep to one line. Comment any entry function that aren't needed/wanted.