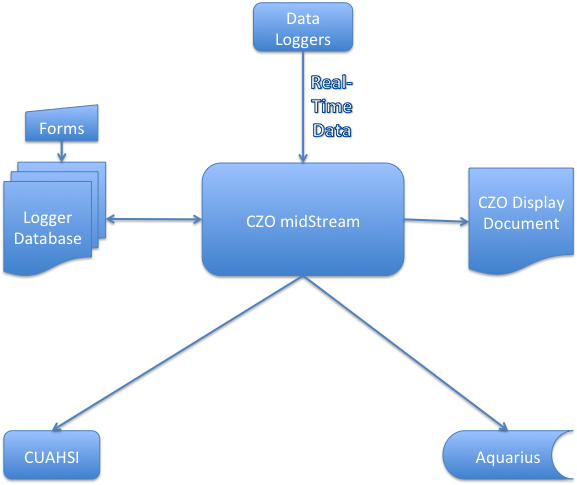
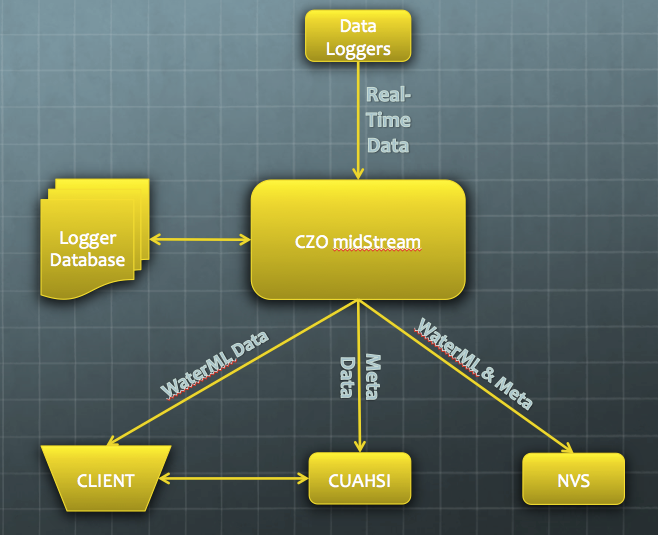
CZO midStream Manual

# Introduction:

Figure : Interactions of CZO midStream



    The Christina River Basin Critical Zone Observatory (CRB-CZO) has been leading the development of a flexible wireless environmental sensor network based on the open-source Arduino electronics platform and XBee ZigBee radio modules (<http://www.stroudcenter.org/research/projects/czo/arduino.shtm>).  This approach has the potential to transform environmental science and education due to the low technical and financial threshold to implementation.  However, a flexible but simple-to-use open-source streaming data management system does not yet exist to get the data from a heterogeneous sensor network to existing data visualization and archiving infrastructure (i.e. <http://his.cuahsi.org/>).

# Brief Scope:

A streaming data system will be developed to automatically translate high frequency, heterogeneous sensor streams with sparse meta-data into meta-data rich time series for local and remote storage to a variety of repositories using web-services.

The CZO midStream system is capable of movement and translation of real-time and data from various data logger modules into a WaterML web service.

# Full Scope:

The goal is to develop a lightweight system that is completely open source and platform independent, to accompany the open-source wireless sensor network package that is being actively developed with the Arduino platform and XBee ZigBee self-meshing radios.

<http://criticalzone.org/christina/infrastructure/sensors-field-instruments-christina/>

Each node transmits a single line once every time period (ex 15 minutes) that is intentionally minimal to preserve bandwidth:

* DateTime, UTCoffset, LoggerID, Value1, Value2, ..., ValueN

These lines are appended to a single CSV file on a server.

A relational database, using just Logger ID and DateTime will provide all the necessary metadata, such as SiteID, SensorIDs, Variables and Variable order, Units, etc. CZOmidStream will access the database to retrieve the metadata.

Wireless streaming sensor data is especially challenging because of the conflicting need for high-frequency output (i.e. output from every sensor in the network every 3-6 minutes) and the bandwidth limitations of a battery-powered wireless network in a forest.  Because every bit counts, the optimal data line transmitted from every logger only includes a time stamp, logger deployment ID and comma-separated data values from each sensed variable.  Logger time zone, sensor locations, variable ID, variable units, and other critical meta-data must all be stored on the network hub and "reattached" to the data as it is being stored, optimally in the form of data time series for each location-variable combination.  For the data to be available to users in near-real time, these time series must then be transferred to existing web-based infrastructure (i.e. <http://his.cuahsi.org/> and others) in xml feeds (i.e. WaterML 2.0 and others) using web services (i.e. <http://his.cuahsi.org/wofws.html> and others).

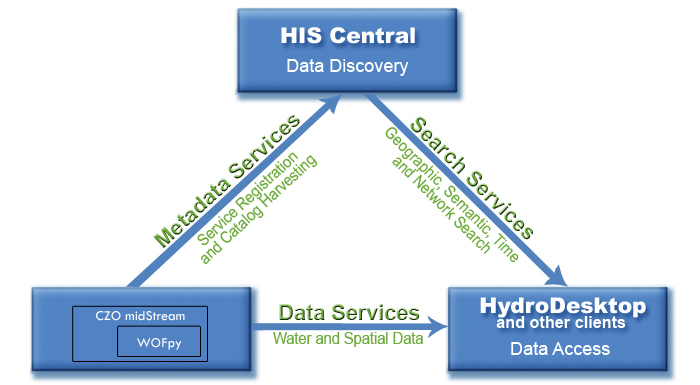
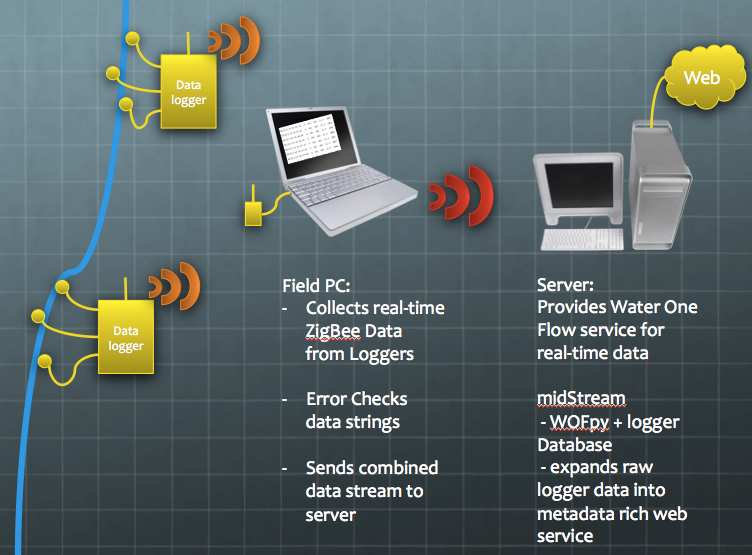


Figure 2: CUAHSI HSI integrating CZO midStream

# Implementation:



CZO midStream consists of three main components:

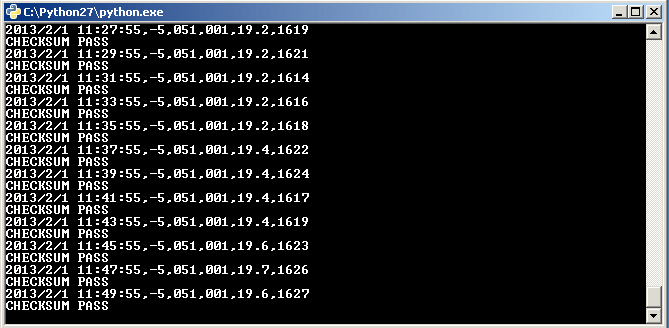
* midStream\_receiver.py
  + This script is constantly running on the field PC
  + Collects strings from transmitting data loggers
  + Error checks and stores strings both locally and remotely to the midStream server
  + On C:\MacMiniData\midStream  
    - monthly data files (ex 2013-1.csv)  
    - invalid\_strings.csv  
      
    On HIS Server (M:\midStream\data)  
    - monthly data files (ex 2013-1.csv)  
    - invalid\_strings.csv  
      
    - If the network disconnects, the local files keep writing, and the  
    strings waiting for the remote server are placed in memory queues.  
    When the server is available again, the queues are emptied into the  
    appropriate files on the server.

- invalid\_strings.csv stores anything that didn't pass the tests: a bad or missing checksum, a bad date/time code, or wrong number of columns, etc

* midStream logger database
  + A database implemented in PostgreSQL containing all the metadata for the logger data streams including:
    - Site properties
    - Sensor properties
    - Researcher information
    - Variable, units, medium
* midStream
  + Provides the Water One Flow web service using data from the logger database as well as raw data received by midStream\_receiver
  + When a request for data is received from the web service, midStream finds the appropriate information in the logger database and logger data files. A WaterML response is served.

# Instructions

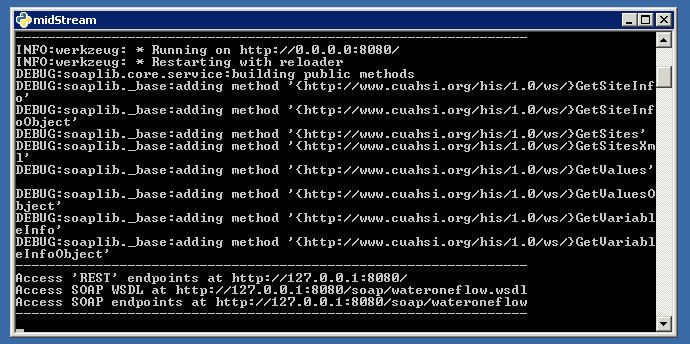
## midStream\_receiver

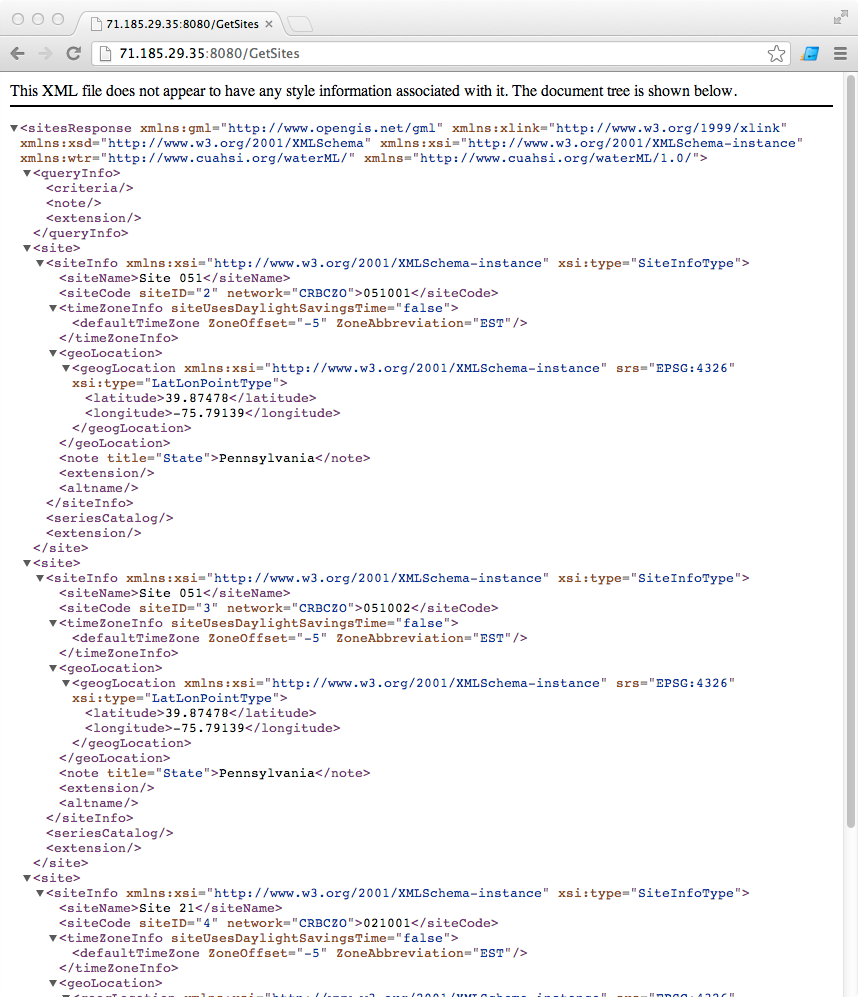


This script currently runs on a macMini running Windows XP. It is set to automatically start when the computer starts. It will begin collecting data from the ZigBee receiver immediately and it will store the data locally and to the HIS server if possible. There is no interaction needed with this script. It runs autonomously and is able to handle many types of errors including mal-formed logger strings as well as network connection problems to the server.

The script will store data:

On C:\MacMiniData\midStream  
- monthly data files (ex 2013-1.csv)  
- invalid\_strings.csv  
  
On HIS Server (M:\midStream\data)  
- monthly data files (ex 2013-1.csv)  
- invalid\_strings.csv  
midStream (on HIS server)





midStream running on the HIS server is the most complex of the three main components. It is based on WOFpy, a python library for providing Water One Flow web services. midStream also runs autonomously and does not require any interaction. It runs as a service on the HIS server and will start automatically when the server boots.

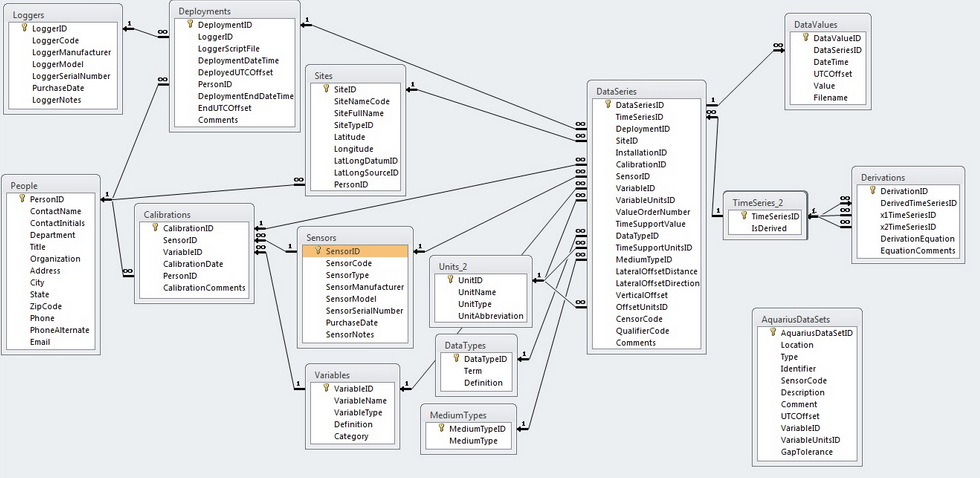
Outside interaction with midStream occurs through the web service located at:

REST Interface: http://71.185.29.35:8080/

SOAP WSDL: http://71.185.29.35:8080/soap/wateroneflow.wsdl

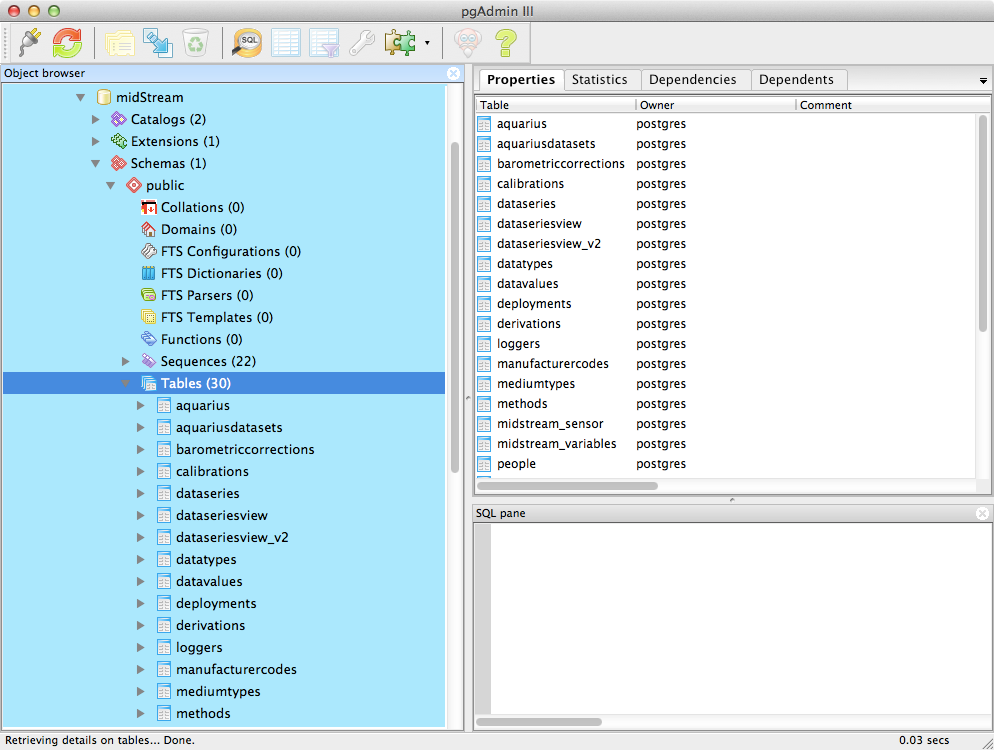
SOAP: http://71.185.29.35:8080/soap/wateroneflow

## Logger database (on HIS server)

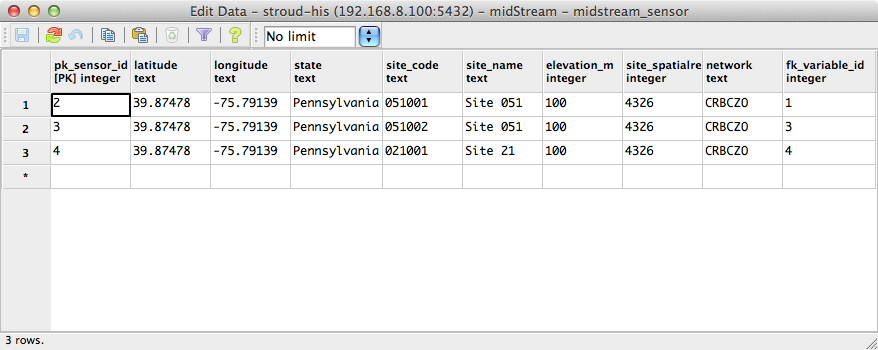


The logger database holds all the metadata for the logger data streams. This is where the back-end configuration of midStream takes-place. Each data stream must have metadata associated with it or the stream will not be available via the Water One Flow web service.

To connect and edit the database, use PGAdmin III which is available on the HIS server. If you want to connect from your local machine the ip address is 192.168.8.100 and the username is postgres. No password is needed when connecting from an internal IP address. Direct connections from external IPs are denied.



Currently, the main table in the database is called midstream\_sensor

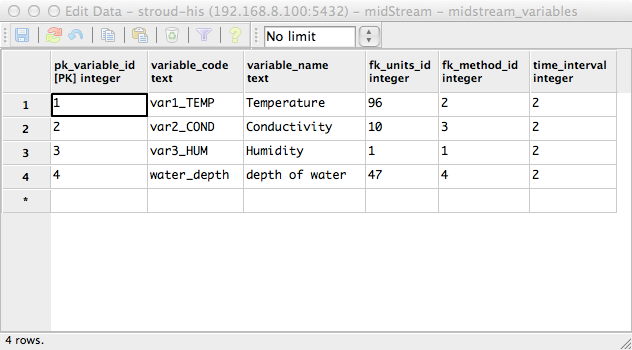


To setup a new real-time logger:

Edit midstream\_sensor

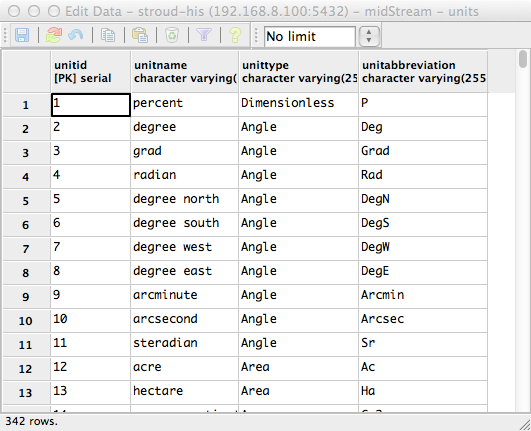
* add a sensor\_id, latitude, longitude, state, site\_name, site\_spatialref
* the site\_code must match the site code being transmitted by the datalogger
* the network should remain CRBCZO
* fk\_variable\_id must correspond to an entry in the midstream\_variables table

The next important table is called mistream\_variables

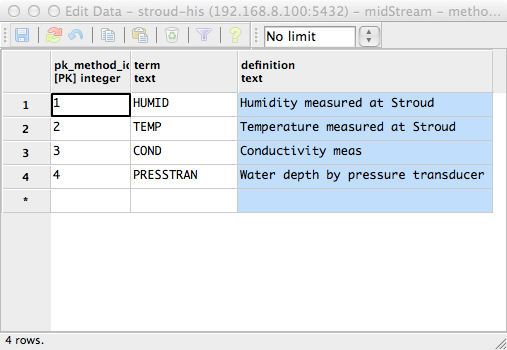


The variable\_code and variable\_name are, for now, arbitrary and can be anything. However, the fk\_units\_id must correspond to a valid units\_id in the units table. The same is true for fk\_method\_id. Time\_interval is the number of minutes between transmissions. This number will not affect the data stream, rather it is simply metadata that is reported along with the data stream.

Units table



Methods table



Once the metadata is entered for a data stream, that data stream will be instantly available by the midStream web service.

<http://71.185.29.35:8080/GetSites>

And you should see your new site among the list of sites.

<http://71.185.29.35:8080/GetSiteInfo?site=CRBCZO:051001>

Replace ‘051001’ with your new site code, and you should see all the metadata for your new site