zxweather

Installation Reference

Document Number: DAZW-IG01

July 2012

This manual provides installation and configuration guidelines for zxweather.

Revision/Update Information:	This is a new manual
Operating System:	Linux; Microsoft Windows NT 5.0+
Software Version:	zxweather 0.1

This document was prepared using LaTeX 2$_\varepsilon$

Contents

1

Introduction

zxweather a collection of tools to store and display data collected by automatic weather stations compatible with the Fine Offset WH1080. It is licensed under the GNU GPL making it free software.

Its main features are:

- A modern HTML5 web interface
- Basic HTML fallback for older browsers
- Full weather history

This manual provides installation and maintenance instructions for the entire core system. You should read this manual in its entirety to avoid problems. You may skip chapter 5 if you do not plan to use the feature it describes.

1.1 Related Documentation

Other available documentation for the zxweather system includes:

DAZW-WG01	WH1080 Utilities Users Guide, version 0.1
DAZW-DB01	zxweather Database Structure, version 0.1

1.2 System Structure

zxweather consists of three major components:

- Database (run on PostgreSQL)
- Data Logger (wh1080d) and other WH1080 utilities
- Web Interface

The database sits at the center of the zxweather system. The Data Logger feeds data into the database where the web interface and other clients can access it.

Because of this architecture all systems must be on the same network as the database server (using either SSH tunnels or VPNs). This can make running the Web Interface on a remote system (such as a VPS) difficult.

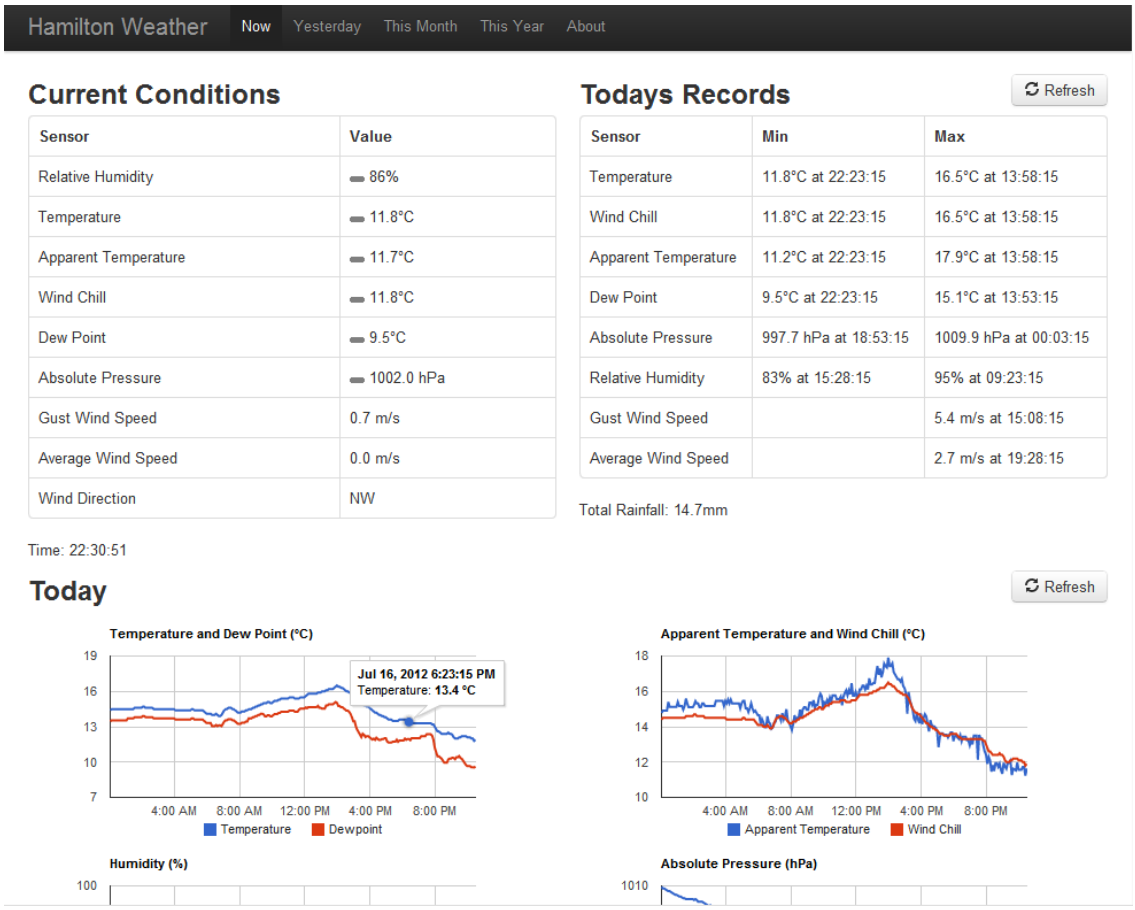

Figure 1.1: Station Overview Page

To support this sort of configuration the Web Interface has a very basic database replication facility built into it. This allows it to have its own database with weather data being pushed out to it at regular intervals.

This sort of configuration is described in more detail in Chapter 5.

1.3 Supported Hardware

This version of zxweather only supports weather stations 100% compatible with the FineOffset WH1080. These weather stations are resold by many companies under various names.

The weather stations sample interval should be set to five minutes. If you have previously used software such as wview you may find your stations sample interval is set to one minute - this is not supported and will not work. Resetting your device should fix this.

Intervals larger than five minutes should work but there may be minor issues in the web interface.

1.4 System Requirements

Linux is the recommended operating system for running the core components of zxweather. Where this is not practical it is possible to run under Microsoft Windows with some limitations.

The only supported database engine is PostgreSQL. Any version from 9.0 and up should be suitable. Older versions may work but this is not tested. Other RDBMS (such as MySQL) are not supported in any way.

The WH1080 tools are only supported on little-endian CPU architectures at this time. This includes Intel IA32 (x86) and most ARM processors. Bad things will happen if your processor is big-endian.

1.4.1 Software Environment

This section covers the software environment required to run the Data Logger and Web Interface. For PostgreSQL system requirements consult its documentation.

Data Logger

If you are running these tools under Microsoft Windows a binary distribution is available that contains all of the required libraries, etc. If you are using this then you do not need to read this section.

The Data Logger and other WH1080 utilities require the following libraries to be installed on your system:

- libusb-1.0 (on linux only)
- libecpg (A part of ECPG)
- libpq (PostgreSQL client library)

Additionally, to compile these you will need:

- GNU C Compiler and GNU Make
- ECPG tool (a part of PostgreSQL)
- Development packages for libusb-1.0 and libecpg

This software and libraries should be available from your operating systems package manager. On Debian 6.0 the packages would be `build-essential libusb-1.0-0-dev libecpg-dev`.

Web Interface

The web interface requires the following to be installed on your system:

- Python 2.6 or 2.7
- The following python libraries:
  - Psycopg2
  - web.py

  - Jinja2
  - python-gnupg
  - requests (if database replication is used)
- Gnuplot
- GnuPG (if database replication is used)
- Apache2 with mod_wsgi (or another web server with WSGI support)

1.5 Distribution Contents

The standard zxweather distribution should be extracted to somewhere on your disk such as `/opt/zxweather`:

```
$ pwd
/opt/zxweather
$ ls
database  db_push  desktop  doc  plot  wh1080  zxw_web
```

The subdirectories are:

- *database* - Scripts to create the database (chapter 2)
- *db_push* - Database replication tools (chapter 5)
- *desktop* - Desktop interface
- *doc* - Documentation
- *plot* - Tools for plotting static charts (chapter 4)
- *wh1080* - The Data Logger and other WH1080 utilities (chapter 3)
- *zxw_web* - The web interface (chapter 4)

The wh1080 and desktop directories contain C/C++ source code. If you are running Microsoft Windows pre-compiled versions of these programs are available. See `http://ftp2.zx.net.nz/pub/DGS/zxweather/readme.html` for download links.

The doc directory contains LaTeX source code for producing the zxweather documentation as well as the compiled versions in Adobe PDF format. This documentation is available in other formats from the URL above.

2

Database

This chapter describes the database setup required by zxweather. The database is required by the core system and is not optional. It must be setup on a system that all other components have network access to.

Alternatively, if it is impractical for all systems to have direct access to the database it is possible to setup a second replica database on the system running the Web Interface. Chapter 5 covers this setup in more detail.

2.1 Installation

The zxweather database has only been tested with PostgreSQL version 9.0 and above. This documentation assumes you already have a suitable version of the server and client tools installed on your system.

This section documents creating the zxweather using the command-line PostgreSQL tools. If you are more comfortable with the pgAdmin III GUI tool you may use that instead.

2.1.1 Create Database

To create the weather database, execute a command such as the following:

```
$ createdb -h dbserver -U username weather "Weather database"
```

Where:

- *dbserver* is the hostname (or IP Address) of the machine running PostgreSQL.
- *username* is the username to login to the server with. This will often be something like "postgres". The account being used must have the *CREATEDB* permission. You will be prompted for the password if required.
- *weather* is the name of the new database.
- *"Weather Database"* is a description of the database. This is optional.

More information on the createdb program can be found at `http://www.postgresql.org/docs/9.1/static/app-createdb.html`.

2.1.2 Create Database Structure

The database structure needed by zxweather to store data is created using the `database.sql` SQL Script located in the *database* subdirectory of the zxweather distribution. Running this SQL script will create the full database structure in one step.

You can run this script with a command such as the following inside the zxweather distribution directory:

```
$ psql -h dbserver -U username -d weather -f database/database.sql
```

The supplied parameters are:

- *dbserver* - The hostname (or IP Address) of the machine running PostgreSQL
- *username* - The user account to login to the server with. Using a superuser account (often called "postgres") will be easiest. You will be prompted for a password if required.
- *weather* - The name of the database you created in the previous section.

2.2 Permissions

Four programs will be accessing the database. They require user accounts with the following permissions on the weather database:

Program	Permissions
wh1080d	CONNECT, SELECT, INSERT, UPDATE
wh1080	CONNECT, SELECT, INSERT
weatherplot	CONNECT, SELECT
web interface	CONNECT, SELECT

You can create individual accounts for each program or have them all sharing the same account.

If the database replication feature in the Web Interface is being used then its user account will need the INSERT and UPDATE permissions in addition to those described in the table above. This is covered in more detail in chapter 5.

3

Data Logger

The Data Logger is a daemon which continuously downloads weather data (both live and historical samples) from the weather station and loads it into the database. It is called *wh1080d* and is one of the WH1080 Utilities. Its full documentation is contained in the *WH1080 Utilities Users Guide, version 0.1* (`DAZW-WG01`).

This chapter covers how to compile and install this tool. Additionally, section 3.2 includes some important maintenance information that must always be taken into consideration before you start wh1080d.

3.1 Compiling

This section only covers compiling the tools under Linux. If you are using Microsoft Windows it is recommended that you use the pre-compiled executables and skip to the next section.

3.1.1 Requirements

To compile the tools under Linux the following software must be installed:

- GNU Make
- GNU C Compiler
- ECPG (a PostgreSQL utility)

The following development libraries are also required:

- libpq (PostgreSQL client library)
- libecpg (library for ECPG)
- libusb-1.0

These libraries and software packages should be available from your operating systems package manager. On Debian 6.0 the packages would be `build-essential libusb-1.0-0-dev libecpg-dev`.

3.1.2 Compiling the WH1080 Tools

To compile the WH1080 tools, cd into the wh1080 subdirectory of the zxweather distribution and run `make`. This should kick off the build process and leave you with a handful of programs inside the wh1080 directory.

3.2 Loading Data

Before you can start the data logger on a non-empty database it is important to note that you may have to run a *full update* using the *wh1080 tool* first. This prevents existing data stored in your weather stations memory from being lost.

Performing a full update when it is not necessary will result in duplicate data being loaded into your database. It is important that a full update is performed only when necessary.

3.2.1 When to perform a Full Update

There are only three occasions when it is acceptable to perform a full update:

- You have just erased the weather stations memory
- You have reset the weather station
- The database is more than 4080 samples out of date

The first two deal with the case where the sample the database says needs to be downloaded next no longer exists on the weather station. The data logger will detect this condition and will refuse to start printing out the message below:

```
Checking for station reset condition...
Fatal Error: wh1080d cannot be restarted after a device reset. Consult
installation reference manual for maintenance procedure to clear error
condition.
```

Performing a full update will fix this and allow the data logger to start.

The third occasion when it is acceptable to perform a full update is when your database is very out of date such that no sample on the weather station exists in your database. This condition is not detected automatically.

In this case it is not strictly necessary to perform a full update as failure to do so will not cause any damage. Not performing a full update in this case will just result in some new data on your weather station being missed.

In order for a full update to be necessary your weather station must be more than 4080 samples out of date. If it is configured to take one sample every five minutes then the database must be more than 340 hours out of date (a little over 14 days).

If in any doubt perform a full update on a test database and compare what it download with what is already in your weather database to see if there is any overlap at all.

3.2.2 Performing a Full Update

To perform a full update you must use the the wh1080 tool as the data logger (wh1080d) can not perform this operation. wh1080 is one of the WH1080 Utilities and is a basic tool for inspecting the contents of the weather station and downloading samples into a database.

The `-l` command-line option causes the wh1080 tool to perform a Full Update instead of a regular one:

```
wh1080 -l -d databasename@hostname -u username -p password
```

This is covered in more detail in Section 2.2.4 of *WH1080 Utilities Users Guide* (DAZW-WG01).

3.3 wh1080d Configuration

3.3.1 Linux

wh1080d (the Data Logger) takes the following arguments:

Argument	Parameter	Description
-d	database	Database connection string
-u	username	Database username
-p	password	Database password
-f	filename	Log file to write messages to

Under linux the Update Service runs as a daemon. To start it just run something like the following from your system startup scripts:

```
wh1080d -d database -u username -p password -f logfile
```

The log file is truncated when the daemon starts.

3.3.2 Windows

Currently wh1080d is not capable of running as a service on windows. Instead you must run the wh1080dtest program which provides the same functionality but stays open in a console window. It takes the following arguments:

Argument	Parameter	Description
-d	database	Database connection string
-u	username	Database username
-p	password	Database password

This will change in a future release.

4

Web Interface

The Web Interface is the primary way for viewing data in the zxweather database. It consists of two components - the WSGI web application and the chart plotting program.

Both components are required for correct operation. This chapter describes how to install and configure them.

4.1 WSGI Application

The web interface is written as a Python WSGI application. This section only covers installing the application under Apache httpd. If you are not using Apache then consult your web servers documentation for instructions on installing wsgi applications.

4.1.1 Installation

At this time zxweather cannot be run in a subdirectory - it must live in the root directory of the website. This generally means giving it its own virtual host.

The system you are installing the web interface on must have the following installed on it:

- Apache httpd
- mod_wsgi
- Python 2.6 or 2.7 with the following packages:
  - psycopg2
  - web.py
  - python-gnupg

Installation and configuration of these packages is outside the scope of this document. If you are setting up on a Linux system the packages are likely all available from your distributions package repositories.

Where examples are provided in this section they are for Debian-based Linux distributions. Installation and configuration on Windows systems is similar but requires more effort.

WSGI Setup

As the zxweather web interface is a Python WSGI application you must have the mod_wsgi installed and enabled. On Debain-based systems the package is called `libapache2-mod-wsgi` and can be enabled using the `a2enmod` tool:

```
$ a2enmod wsgi
```

Virtual-host Configuration

All that is required to make zxweather work from the Apache end is adding the following to your vhost configuration file:

```
WSGIScriptAlias / /opt/zxweather/zxw_web/zxweather.py
```

An example virtual host configuration might look like this:

```
<VirtualHost *:80>
        ServerAdmin admin@example.com
        ServerName weather.example.com

        DocumentRoot /var/www
        ErrorLog ${APACHE_LOG_DIR}/weather-error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/weather-access.log combined

        WSGIScriptAlias / /opt/zxweather/zxw_web/zxweather.py
</VirtualHost>
```

4.1.2 Configuration

The Web Interface attempts to load configuration files in the following order. Only the first one it finds is loaded.

- `config.cfg` (in the current directory)
- `zxw_web/config.cfg`
- `/etc/zxweather.cfg`

It is recommended that you install your configuration file in `/etc`. To do this, copy the `zxweather.cfg.sample` file included in the zxweather distributions `zxw_web` subdirectory:

```
$ cp /opt/zxweather/zxw_web/zxweather.cfg.sample /etc/zxweather.cfg
```

This configuration files format is similar to the INI format commonly used on Microsoft Windows. The '#' character marks a comment, group names are inside square brackets. Key-value pairs are written in the following way:

```
key: value
```

Database

The first group is for database configuration. It looks like the following:

```
[database]
host: localhost
port: 5432
database: weather
user: weatheruser
password: password
```

Where the settings are:

- host - the hostname or IP address of your database server
- port - The port your database listens on. This is commonly 5432 for PostgreSQL
- database - Name of your database
- user - The user to login to the server as
- password - The users password

Data

The Data group stores details about the data available in your database:

```
[data]
live_data_available: True
sample_interval: 300
plot_interval: 1800
```

The settings are:

- live_data_available - If Live Data is available from the database. If you are using wh1080d to populate your database then this should be left as `True`.
- sample_interval - How often new samples appear in the database (in seconds). Only a 5 minute sample interval is supported at the moment so this should be left as 300.
- plot_interval - How often the weatherplot program runs (see section 4.2) in seconds.

Site

The Site group stores basic web interface configuration:

```
[site]
default_ui: s
site_name: zxweather
# UNCOMMENT THESE AND SET THEM! The defaults will NOT work.
#site_root: http://weather.example.com/
#static_data_dir: /opt/zxweather/zxw_web/weather/static/
#station_name: abc
```

Where the settings are:

- default_ui - The default UI to use. 's' is the Standard UI, 'b' is the Basic (HTML-only) UI. Unless your only browser is from 2001 you will want to leave this on 's'.
- site_name - The name of your site. This appears on the left of the navigation bar and in the title of every page.
- site_root - The URL for the root of your website. For example, http://weather.example.com/.
- static_data_dir - Where on your hard disk the static data directory is. If zxweather was extrated to `/opt/zxweather` then this should be set to `/opt/zxweather/zxw_web/static/`.

The final setting in this group is station_name. This is a very short (a few characters) name for your weather station. It should be kept very short as it will appear in the URL of every page in your site. For example, if you set this value to "foo" your station overview page will be `http://weather.example.com/s/foo/`.

You must also create a directory of the same name inside the static data directory:

```
$ mkdir /opt/zxweather/zxw_web/static/foo
```

The purpose of this setting is to allow multiple weather datasets to be handled by a single Web Interface instance in a future version of zxweather. Uses for this functionality will be:

- Multiple weather stations
- Separating datasets if you move the weather station to a different location

Once you have chosen a value for this setting you should never change it as it will change every URL in your site.

database_replication

This group configures the database replication feature. If you do not plan on using this feature then the default values (which turn the feature off) are acceptable.

If you are going to use this feature then these settings are described in Chapter 5.

4.1.3 About Page

On the navigation bar of the standard Web Interface is an "About" link which will take you to a generic about page which you will want to customise.

To do this, navigate into the *static data* directory (`/zxw_web/static/`) and copy the `about.html` file into your weather stations subdirectory. If, for example, your station is called "foo" and zxweather is installed in `/opt/zxweather` you would copy `/opt/zxweather/zxw_web/static/about.html` to `/opt/zxweather/zxw_web/static/foo/about.html`. You can then customise this copy of the file. You should avoid modifying the original copy as it may be overwritten without warning by future versions.

When editing your copy of `about.html` you will find two comments near the bottom of the page; `<!-- BEGIN_USER_CONTENT -->` and `<!-- END_USER_CONTENT -->`. You can put anything you want between these comments.

It is best to avoid making any changes outside of these comments as future versions may make changes to the file outside these comments as part of the upgrade process.

4.2 Chart Plotting

The *weatherplot* program is responsible for generating static charts primarily used by the basic HTML web interface. It must be setup as a scheduled task to be run at regular intervals.

The machine it runs on must have *gnuplot* installed and must have access to the database server and directory the zxweather web interface runs from.

4.2.1 Usage

The weatherplot program is written in the Python language and lives in the `plot` subdirectory of the zxweather distribution. It is executed on the command-line with python as:
`$ python plot/weatherplot.py [arguments].`

The charts it generates must be put in the web interfaces *static data* directory in a subdirectory with the same name as your station. For example, if your station is named "foo" and the zxweather distribution was extracted to `/opt/zxweather` then you would generate charts into `/opt/zxweather/zxw_web/static/foo/`.

By default an executable called `gnuplot` is expected to be in the path which it can use to generate the charts. If your install of gnuplot is not in the path or goes by another name use the `--gnuplot-binary` parameter to specify its name.

Command-line Arguments

The weatherplot program accepts the following command-line arguments:

Argument	Parameter	Description
`-t` `--database`	dbname	Name of the database to use. Required.
`-n` `--host`	hostname	Database server hostname. Required.
`-u` `--user`	username	Username for database server. Required.
`-p` `--password`	password	Password for database server. Required.
`-d` `--directory`	directory	Output directory. Required.
`-a` `--plot-new`	filename	Only plots charts for dates on or after that stored in the specified file.
`-r` `--replot-pause`	seconds	Number of seconds to wait before replotting.
`-g` `--gnuplot-binary`	filename	Name of the gnuplot executable to use if it is something other than "gnuplot".

The database, host, user, password and directory parameters are always required.

4.2.2 Plotting All Charts

To regenerate charts for your entire database run the weatherplot program with only the minimum command-line arguments:

```
$ python plot/weatherplot.py --database weather --host localhost \
--user postgres --password password \
--directory zxw_web/static/station_name/
```

This will create charts for all days and months in your database and store them in the specified directory. Depending on the size of your database this may take some time.

Some software upgrades may require you to do this when new chart types have been added or the style of the charts has been adjusted.

4.2.3 Running as a Scheduled Task

The recommended way to setup the weatherplot program is to run it as a scheduled task from cron or the windows task scheduler. When run in this way it is important that it be set to only plot charts containing new data.

The `--plot-new` command-line argument implements this. The argument takes a single parameter which is the name of a file to store the date of the last plotted day in.

Each time the weatherplot program is executed with this parameter it will replot all charts for all days and months on or after the date in that file and then update the file with todays date. That way only charts that need to be regenerated are regenerated.

Example

When run as below weatherplot will only replot charts that have changed since it was last run:

```
$ python plot/weatherplot.py --database weather --host localhost
--user postgres --password password --directory static/station_name/
--plot-new plot_status_file
```

To make this run every 30 minutes you would add a line such as the following to /etc/crontab:

```
0,30 *  * * *   root    cd /var/zxweather && \
  python plot/weatherplot.py -t weather -n localhost -u postgres \
  -p password -d zxw_web/static/station_name -a plot_status_file
```

If the static charts are not important to you, you may wish to only regenerate them every few hours or once a day.

4.2.4 Plotting Continuously

The weatherplot program is capable of running interactively in continuous mode. When run like this it will automatically replot charts at a specific interval until you terminate it with Ctrl+C. This is primarily intended for testing purposes.

It can be run in this mode by supplying the `--replot-pause` parameter with a suitable interval in seconds.

Example

When run as below the behaviour is the same as setting it up to be run by cron every 30 minutes except it runs continuously attached to the terminal.

```
$ python plot/weatherplot.py --database weather --host localhost \
--user postgres --password password --directory zxw_web/static/rua \
--plot-new plot_status_file --replot-pause 1800

Weather data plotting application v1.0
        (C) Copyright David Goodwin, 2012


Connecting to database...
Server version: PostgreSQL 9.1.2, compiled by Visual C++ build 1500
Generating temperature plots in zxw_web/static/rua
Plotting from 2012-05-10
Plotting graphs for 2012...
Plotting graphs for 2012 may...
Plot zxw_web/static/rua/2012/may/temperature_tdp_large.png
Plot zxw_web/static/rua/2012/may/temperature_awc_large.png
Plot zxw_web/static/rua/2012/may/humidity_large.png
Plot zxw_web/static/rua/2012/may/indoor_humidity_large.png
Plot zxw_web/static/rua/2012/may/pressure_large.png
Plot zxw_web/static/rua/2012/may/indoor_temperature_large.png
Plot zxw_web/static/rua/2012/may/temperature_tdp.png
Plot zxw_web/static/rua/2012/may/temperature_awc.png
Plot zxw_web/static/rua/2012/may/humidity.png
Plot zxw_web/static/rua/2012/may/indoor_humidity.png
Plot zxw_web/static/rua/2012/may/pressure.png
Plot zxw_web/static/rua/2012/may/indoor_temperature.png
Plotting graphs for 2012 may 9...Skip
Plotting graphs for 2012 may 10...
Plot zxw_web/static/rua/2012/may/10/temperature_tdp_large.png
Plot zxw_web/static/rua/2012/may/10/temperature_awc_large.png
Plot zxw_web/static/rua/2012/may/10/humidity_large.png
Plot zxw_web/static/rua/2012/may/10/indoor_humidity_large.png
Plot zxw_web/static/rua/2012/may/10/pressure_large.png
Plot zxw_web/static/rua/2012/may/10/indoor_temperature_large.png
Plot zxw_web/static/rua/2012/may/10/temperature_tdp.png
Plot zxw_web/static/rua/2012/may/10/temperature_awc.png
Plot zxw_web/static/rua/2012/may/10/humidity.png
Plot zxw_web/static/rua/2012/may/10/indoor_humidity.png
Plot zxw_web/static/rua/2012/may/10/pressure.png
Plot zxw_web/static/rua/2012/may/10/indoor_temperature.png
Plot zxw_web/static/rua/2012/may/10/7-day_temperature_tdp_large.png
Plot zxw_web/static/rua/2012/may/10/7-day_temperature_awc_large.png
Plot zxw_web/static/rua/2012/may/10/7-day_humidity_large.png
Plot zxw_web/static/rua/2012/may/10/7-day_indoor_humidity_large.png
```

```
Plot zxw_web/static/rua/2012/may/10/7-day_pressure_large.png
Plot zxw_web/static/rua/2012/may/10/7-day_indoor_temperature_large.png
Plot zxw_web/static/rua/2012/may/10/7-day_temperature_tdp.png
Plot zxw_web/static/rua/2012/may/10/7-day_temperature_awc.png
Plot zxw_web/static/rua/2012/may/10/7-day_humidity.png
Plot zxw_web/static/rua/2012/may/10/7-day_indoor_humidity.png
Plot zxw_web/static/rua/2012/may/10/7-day_pressure.png
Plot zxw_web/static/rua/2012/may/10/7-day_indoor_temperature.png
Plot completed at 2012-05-10 22:31:46.953000
Waiting for 1800 seconds to plot again. Press Ctrl+C to terminate.
```

5

Database Replication

zxweather includes a very basic database replication feature allowing data to be pushed to a remote server running the web interface with its own copy of the database.

This is intended for situations where the web interface needs to run on a remote network that doesn't have access to your weather database (such as on a VPS).

Before using this feature you should investigate whether SSH tunnels, VPNs or any of the standard PostgreSQL database replication solutions are practical.

If you decide to use this feature you should have a fully working local setup installed first.

5.1 Overview

The built-in replication feature consists of the *db_push* program (in the `db_push` subdirectory of the zxweather distribution) running locally pushing digitally signed data updates out to the remote web interface which then uses the data to update its own private database.

The db_push program runs continuously and sends an HTTP POST to the web interface every time live data is updated or new samples are inserted into the main weather database.

When run in this configuration you will have two database servers with identical database structures:

- Your main weather database running on your local server (the local database)
- The replica database running on the remote server (the remote database)

You will also need to have the full web interface running on the remote server (in addition to it running on your local server if desired). It is recommended that you have the web interface setup somewhere locally too for testing purposes and to familiarise yourself with the setup procedure before trying to get it running on a remote system.

5.1.1 Security

The web interface will only accept data posted to it with a valid PGP signature. If a key fingerprint is specified in the web interface configuration file then it will only accept data signed with that particular key.

The data posted to the web interface is not encrypted in any way as it contains no sensitive information. The data posted to the web interface consists of nothing more than a subset of what is displayed by the web interface.

Usernames and passwords are unnecessary as the web interface will reject anything that doesn't have a valid signature created using your private key if setup correctly.

5.2 Remote Server Installation

The remote server requires the following components installed on it:

- Database
- Web Interface

Setting up the remote database is exactly the same as setting up your local database. Follow the instructions in chapter 2 except on your remote server rather than the local one.

The procedure for setting up the web interface is much the same as for local except for a few details:

- The database configuration must be for the remote database (which, for added confusion, is local to the web interface).
- Database Replication must be enabled in the configuration file.
- GnuPG must also be installed on the remote server the web interface is running on in addition to all the standard requirements.

To install the web interface follow the instructions in chapter 4.

5.3 db_push Setup

The db_push program runs on your local server. It is responsible for sending data updates to the remote web interface.

This section covers generating signing keys and the configuration necessary for your local server to run this program.

You must also complete the setup covered by section 5.4.1 before you will be able to execute this program and begin pushing data updates to the web interface.

This program is not currently capable of being run as a daemon - it must be run interactively or detached from the terminal. It can be run under both windows and linux and requires a direct connection to the database server.

5.3.1 Software Requirements

The db_push program requires the following software to be installed and configured on your local server:

- Python 2.6 or 2.7
- The following python packages:
  - requests
  - python-gnupg

5.3.2 Creating Signing Keys

It is recommended that GnuPG uses a home directory specific to zxweather so that only the keys used for sending data to the remote database are present. All you need to do is create a blank directory and GnuPG will do the rest:

```
$ mkdir /opt/zxweather/gnupg_home
```

You can then create new signing keys using either GnuPG or the *generate_key* tool included in the `db_push` subdirectory of the zxweather distribution:

```
$ python db_push/generate_key.py -d /opt/zxweather/gnupg_home \
-f public_key.asc
```

The `-d` argument specifies the GnuPG home directory to store the new key in. This should be your zxweather-specific GnuPG home directory. The `-f` argument specifies the name of the file to export the newly generated public key to. This public key file must be later uploaded to your web server.

When run the program will produce output like the following:

```
This tool generates signing keys for zxweather database replication.
GnuPG Home: /opt/zxweather/gnupg_home
Public Key Filename: public_key.asc
Creating a new signing key. This may take a while.
Key Fingerprint: C7EC0D7AA1BD69BC0671CCA401787FE106493116
Public key (for web server) written to public_key.asc
```

The full list of arguments accepted by the generate_key program is:

Argument	Parameter	Description
`-d` `--gpg-home-directory`	directory	GnuPG Home Directory
`-b` `--gpg-binary`	gpg binary	The GnuPG binary to use
`-f` `--pubkey-filename`	filename	File to export the new public key to.

5.3.3 Running db_push

The db_push program can either be run continuously (pushing database changes as they appear) or it can be run for a single update. It can not be run as a daemon at this time.

Continuous mode is enabled using the `-c` argument. This is the mode you should always run in except perhaps when testing or bulk loading data into the remote database.

Only one instance of db_push should be run at a time to prevent possible corruption of the remote database.

In order for the program to successfully load data into the remote database it must be enabled in the remote web interfaces configuration file. This is covered in section 5.4.1.

Example

```
$ python db_push/db_push.py -c -t weather -n localhost \
-u weather_user -p password -s http://weather.example.com/ \
-d /opt/zxweather/gnupg_home \
-k C7EC0D7AA1BD69BC0671CCA401787FE106493116
```

The arguments used here are:

- `-c` to enter continuous mode
- `-t weather` to specify the name of the weather database
- `-n localhost` to set the database server hostname
- `-u weather_user` to set the database username
- `-p password` to set the database password
- `-s http://weather.example.com/` is the URL for the web interface we are pushing data to
- `-d /opt/zxweather/gnupg_home` is the GnuPG home directory for zxweather
- `-k ...` specifies the signing key to use. If your GnuPG home directory only contains one key (as it should if its specific to zxweather) then you can leave this off.

When run you will see something like the following:

```
zxweather database replicator v1.0
(C) Copyright David Goodwin, 2012


Connecting to database...
Server version: PostgreSQL 9.1.2, 64-bit
Performing update...
Updating from: 2012-07-15 16:33:15+12
Performing chunked sample load...
Chunk 0/2
Sending data...
Response: OK
Live Data Updated: False
Samples Inserted: 50
Chunk 1/2
Sending data...
Response: OK
Live Data Updated: False
Samples Inserted: 50
Chunk 2/2
Sending data...
Response: OK
Live Data Updated: False
Samples Inserted: 50
Sending data...
Response: OK
```

```
Live Data Updated: True
Samples Inserted: 0
Continuous mode. Waiting for new data...
..
Only updating live data
Sending data...
Response: OK
Live Data Updated: True
Samples Inserted: 0
..........
.
Updating from: 2012-07-16 7:21:15+12
Sending data...
Response: OK
Live Data Updated: True
Samples Inserted: 1
...
```

Here the program first sends data out 50 records at a time until the entire local database has been copied to the remote server. In this case the local database only contains 150 records (sent in three chunks).

It then enters continuous mode where it wakes up when ever something changes in the database. The first time it wakes up was caused by something updating the live data so it sends live data only to the remote server. The second time a new sample has been inserted as well as the live data being updated so both of these are sent.

Command-line Arguments

The full list of arguments accepted by db_push is:

Argument	Parameter	Description
`-t` `--database`	dbname	Name of the database to use. Required.
`-n` `--host`	hostname	Database server hostname. Required.
`-u` `--user`	username	Username for database server. Required.
`-p` `--password`	password	Password for database server. Required.
`-s` `--site`	url	Base URL for the web interface.
`-c` `--continuous`		Keep running after sending the initial update.
`-d` `--gpg-home-directory`	directory	GnuPG Home Directory
`-b` `--gpg-binary`	gpg binary	The GnuPG binary to use
`-k` `--key-id`	key id	The key to sign data with

5.4 Web Interface Setup

When database replication is used the web interface is responsible for inserting data into the database. Because of this its database user account must have `UPDATE` permissions on the `live_data` table and `INSERT` permissions on the `sample` table.

When run in this configuration the web interface requires GnuPG to be installed on the system in addition to all the software required normally.

5.4.1 Installing Keys

As described in section 5.1.1 the zxweather database replication feature uses digital signatures to prevent unauthorised people from pushing data into the remote database.

In section 5.3.2 you will have created new keys for signing. A public key will have been generated as part of this process and written to a file (`public_key.asc` in the examples). You must now upload that public key to your web server and load it into the web interfaces GnuPG home directory.

This procedure is similar to what you went through for setting up db_push. First you need to create a new GnuPG home directory on the web server:

```
$ mkdir /opt/zxweather/gnupg_home
```

This directory must be owned by the user the web interface runs as. On debian this is the `www-data` user.

You can then use the *install_key* utility in the `db_push` subdirectory of the zxweather distribution to install your public key:

```
$ python db_push/install_key.py -d /opt/zxweather/gnupg_home \
-f public_key.asc
```

This program will print out a message such as the one below:

```
Key Fingerprint: C7EC0D7AA1BD69BC0671CCA401787FE106493116
Add the fingerprint above to the web interface configuration file
```

Make a note of the key fingerprint as this will be required when you configure the web interface.

5.4.2 Web Interface Configuration

In order to use the database replication feature you must enable it in the Web Interfaces configuration file. This is done in the database_replication group towards the end of the file which will look something like the following:

```
[database_replication]
enable: False
#gnupg_home:
#gpg_binary:
#key_fingerprint:
```

The settings here are:

- enable - If the feature is enabled or not. Set this to "True".
- gnupg_home - The location where GnuPG stores keys, etc. Uncomment this and et it to the path of the gnupg home directory you created (eg, `/opt/zxweather/gnupg_home`)
- gpg_binary - If GnuPG is installed in an unusual location on your system, specify it here.
- key_fingerprint - The signing keys fingerprint. If specified then only data signed with that key will be accepted. If not specified then any data with a valid signature is accepted. You should uncomment this setting and insert the key fingerprint given to you when you installed the public key.

Once this is done the section will look something like:

```
[database_replication]
enable: True
gnupg_home: /opt/zxweather/gnupg_home
#gpg_binary:
key_fingerprint: C7EC0D7AA1BD69BC0671CCA401787FE106493116
```

You should now be able to restart Apache and run the db_push tool to start pushing data into the web interfaces database.

6

Troubleshooting

6.1 Data Logger

Problem	Cause	Solution
The data logger has crashed trying to read from the weather station.	The weather station has crashed (the history function on the weather station console does not work)	Kill the data logger process, hard-reset the weather station (remove and reinstall batteries and USB cable) and then restart the data logger after performing a full update.
The data logger exits with a database error written to the log file	The connection to the database was lost	Check database connectivity and restart the data logger.
	Database parameters are incorrect	Check database parameters and restart the data logger.
	The database was not created correctly.	Recreate the database and restart the data logger.

6.2 Web Interface

This section covers errors you may encounter when viewing pages in the web interface.

Problem	Cause	Solution
The station overview page gives a "404 Not Found" error	The database is empty	Wait for data to appear in the database

6.3 Database Replication

This section covers issues you may encounter with database replication. Database replication itself is covered by chapter 5.

6.3.1 Error Responses

When the remote web interface reports an error it appears in the output from db_push like in the example below:

```
Sending data...
Response: ERROR: error message
Live Data Updated: False
Samples Inserted: 0
```

Message	Cause	Solution
	Key not installed	Install public key (Section 5.4.1)
Verification failed - bad or missing signature. Status: no public key	GnuPG Home Directory Ownership Incorrect	Change owner of web interfaces GnuPG Home directory to the user the web interface runs as
	Configured GnuPG home directory does not exist	Section 5.4.1 covers creating it. Section 5.4.1 covers configuring the web interface.