# waveserv Documentation

***Release 1.2***

**Brendan Smithyman**

July 12, 2012

# CONTENTS

The **waveserv** program provides a web-based interface designed to simplify monitoring FULLWV and OMEGA. It is specifically designed for use with this software, but is open-source and may be modified or reused freely. While **waveserv** is less capable than a full seismic processing suite, it is compatible with a wide variety of devices and web browsers. It also takes advantage of compression and advanced web techniques to make it very easy to monitor your server/cluster/workstation over a slow Internet connection.

The command-line program **waveserv** indexes all of the relevant files in the OMEGA/FULLWV project directory and provides them to the web browser of your choice. This could (by default) be a browser on the same computer, or it could be a browser running on a computer half way around the world. The interface is lightweight, and works fine on laptops, workstations, phones and tablets. The server interface has been tested on Mac OS X and Linux, but will most likely work on any machine with a functional Python environment. It is based on the development web server from the Django Project.



The **waveserv** package also makes use of the `pygeo` Python module, and in particular the `pygeo.fullpy` interface for reading *projnm.ini* files and `pygeo.segyread.SEGYFile` interface for reading the SEG-Y format datafiles used by OMEGA/FULLWV. `pygeo` is released under the GNU Lesser General Public License.

Contents:

CHAPTER

ONE

# INSTALLATION

**waveserv** can be accessed via Subversion using the following command:

```
svn co https://skylab.bitsmithy.net/svn/public/waveserv/trunk waveserv-base
```

This checks out the program source code into a directory called **waveserv-base**, using the SVN *trunk* version.

---

**Note:** The *trunk* should always be (more or less) functional, but likely does not include the most up-to-date developments. For the development version, issue:

```
svn co https://skylab.bitsmithy.net/svn/public/waveserv/branches/devel waveserv-devel
```

However, note that the development version is not guaranteed to do anything other than possibly melt your computer; use at your own risk.

---

In order to make **waveserv** available system-wide, you need to add the **waveserv-base** directory to the system search path. For users of **bash** (Mac OS and Linux default in most cases), add the following to your *~/.bash_profile* or *~/.bashrc* file:

```
export PATH="/path/to/waveserv-base:$PATH"
```

For users of **csh**, add the following to your *~/.cshrc* file:

```
setenv PATH "/path/to/waveserv-base:$PATH"
```

## 1.1 Custom Libraries

**waveserv** requires the `pygeo` Python module in order to run. To get a copy, issue the command:

```
svn co https://skylab.bitsmithy.net/svn/public/pygeo/trunk pygeo-base
```

Place the directory *pygeo-base* on your Python search path; in **bash**, this would look something like:

```
if [ -n "$PYTHONPATH" ]; then
  export PYTHONPATH="/path/to/pygeo-base:$PYTHONPATH"
else
  export PYTHONPATH="/path/to/pygeo-base"
fi
```

## 1.2 Generic Libraries

This requires the following Python libraries installed on the system, and is tested with Python 2.6 or 2.7:

| Package | Description | Minimum Version |
|---|---|---|
| python | Python Interpreter | Tested with 2.6+ |
| python-numpy | Numerical Python | Untested |
| python-django | Django Web Framework | 1.2 |
| python-matplotlib | Matlab-like plotting environment | Untested |
| python-imaging | Python Imaging Library (PIL) | Untested |

The names above correspond to Debian/Ubuntu packages. If you are using Linux, I would expect all of the generic libraries to be in the package manager, but if not (e.g. CentOS has problematic Python support) they could be installed using another method like "easy_install". I have not tested with older versions of numpy and matplotlib, but I'm fairly sure that the Django framework needs to be at minimum v1.2.

For Ubuntu/Debian users, you will need to run:

```
sudo apt-get install python python-numpy python-django python-matplotlib python-imaging
```

For CentOS users, you will need to run:

```
su
yum install python numpy Django python-matplotlib python-imaging
```

---

**Note:** If you are using Mac OS X (or if you want to use Python on MS Windows), I highly recommend the commercial EPD (Enthought Python Distribution). This is free for academic use, and reasonably priced for industry use given the level of support offered. Additionally, there is a free version that most likely would suffice for casual use. This can also be extremely handy for poorly-behaved Linux distributions. Enthought supports Python development and open source.

---

CHAPTER

TWO

# COMMAND LINE INTERFACE

The **waveserv** CLI is a Python program that loads up a specially modified web server. However, instead of hosting a web site, this interface looks for Waveform Tomography project files and generates an interactive listing.

You may run **waveserv** by navigating to the directory containing FULLWV/OMEGA files and typing **waveserv [projnm]**, where **[projnm]** is the name associated with the ***.ini** file. It can also auto-detect this if there is only one project in the directory, but this fails (for example) during inversion because of the generation of **tmp.*** files.

Access the *Graphical User Interface* by pointing a web browser at http://localhost:1503/ (by default), or use the command-line interface to change the port ("waveserv –help"). If the server is not running on the local machine (i.e., your web browser is not on the machine that OMEGA/FULLWV runs on), you can specify the bind address as something other than **127.0.0.1**. This is a potential security issue, so in most cases it is better to use VPN (Virtual Private Networking, e.g., OpenVPN) or SSH-tunnel methods to access it from the local machine. To set up a SSH tunnel, issue the following command on the machine running the web browser:

```
ssh -f -N -L 1503:localhost:1503 [remotehost]
```

In this case, connect to the address http://localhost:1503/ as if you were running **waveserv** on the same computer.

> **Warning:** **waveserv** is based on the Django test web server. This is designed for development use, and is not recommended for use on public networks (viz., the Internet). Django is a popular project, and its vulnerabilities may be well-known; **waveserv** is susceptible to the same issues. By default, **waveserv** is only accessible from the local computer, and so is only accessible to other users with accounts on the server/workstation. The *-b* option should only be used on a secure network.

## 2.1 Options

**-version**
**show program version number and exit**

**-h, -help**
**show a help message and exit**

**-v, -verbose**
**display additional information**

**-b** ADDR, **-bind**=ADDR
**server address to bind [Default: 127.0.0.1:1503]**

**-d** FILE, **-db**=FILE
**filename for server database [Default: .waveserv.db]**

**-g, -serverdebug**

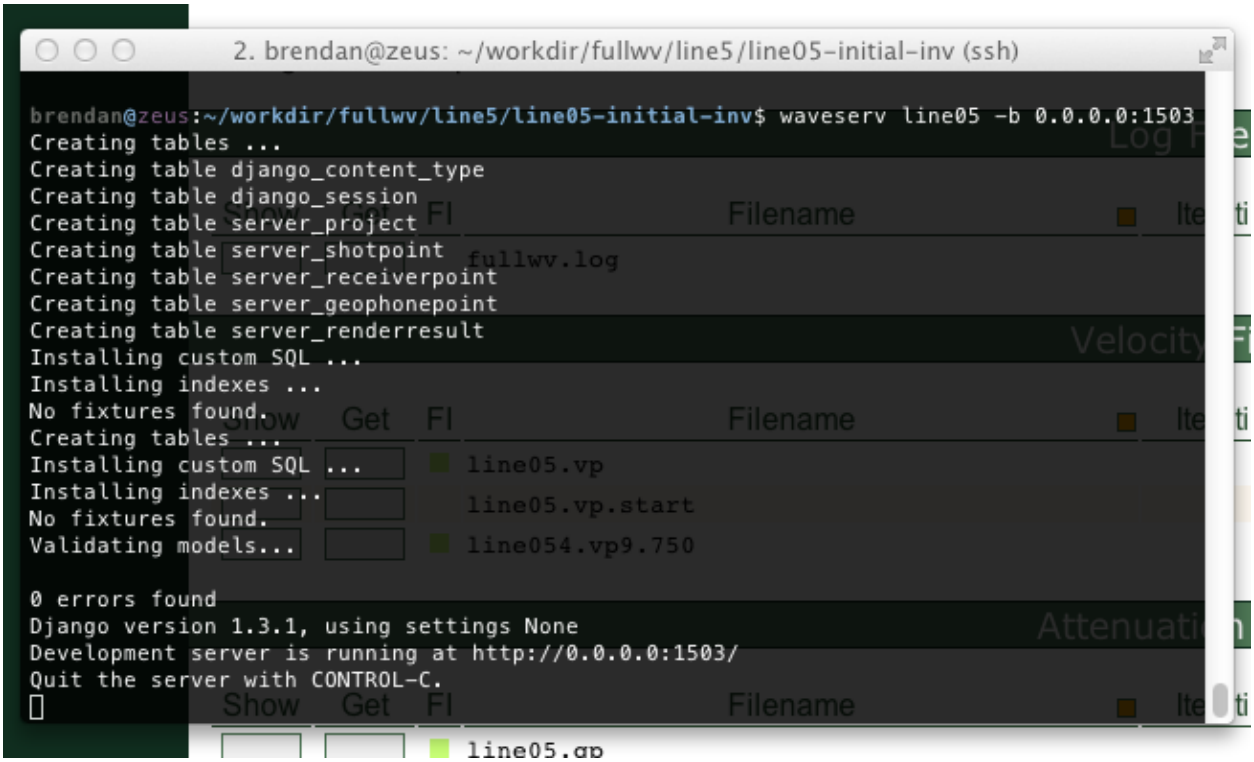


Figure 2.1: Example execution of **waveserv**, for a project named *line05*.

**enable debugging for web server**

**-k, -keep**
**keep temporary files on exit [Default: False]**

**-p** DIBUPATH, **-dibupath**=DIBUPATH
**path for temporary image files [Default: .buffer]**

CHAPTER

THREE

# GRAPHICAL USER INTERFACE

The **waveserv** program directly indexes and reads the (SEG-Y) files in the directory working directory on the server, handling endian and floating point conversion, and provides a directory listing through the web-browser interface. The web-based GUI is rendered by the **waveserv** executable, which is started using a *Command Line Interface*.

With the **waveserv** server running, navigate to http://localhost:1503/index to show the directory listing:

Log Files

| Show | Get | FI | Filename | Iteration | Frequency | Size | Modified |
|---|---|---|---|---|---|---|---|
| | | | fullwv.log | | | 7.0 MB | 2012-06-15 14:09:19 PDT |

Velocity Files

| Show | Get | FI | Filename | Iteration | Frequency | Size | Modified |
|---|---|---|---|---|---|---|---|
| | | | line05.vp | | | 513.3 KB | 2012-06-14 19:52:09 PDT |
| | | | line05.vp.start | | | 513.3 KB | 2012-06-09 10:38:55 PDT |
| | | | line054.vp9.750 | 4 | 9.750 | 513.3 KB | 2012-06-14 19:52:09 PDT |

Attenuation Files

| Show | Get | FI | Filename | Iteration | Frequency | Size | Modified |
|---|---|---|---|---|---|---|---|
| | | | line05.qp | | | 513.3 KB | 2012-06-14 19:52:09 PDT |
| | | | line05.qp.start | | | 513.3 KB | 2012-06-09 10:38:43 PDT |
| | | | line054.qp9.750 | 4 | 9.750 | 513.3 KB | 2012-06-14 19:52:09 PDT |

iVelocity Files

| Show | Get | FI | Filename | Iteration | Frequency | Size | Modified |
|---|---|---|---|---|---|---|---|
| | | | line054.vpi9.750 | 4 | 9.750 | 513.3 KB | 2012-06-14 19:52:09 PDT |

Figure 3.1: Directory listing rendered by **waveserv**.

Each type of file is grouped under a descriptive heading, and the entries under each heading are sorted. Several pieces of information are available for each entry, including the *filename*, *size*, and *modification time*. Additionally, for files that correspond to a particular *frequency* or *iteration*, this information is extracted from the filename. Each of these headings can be used as a sort key; simply click the corresponding check box (beside the column heading) to change the sort key. The current sort key is indicated by a box that is coloured differently from the others. New in version 1.1: An indicator shows the freshness of each file graphically. Each entry has two buttons, labelled *Show* and *Get*. Clicking the *Get* button simply downloads the original file. The *Show* button beside an entry displays a graphical representation of the file. This takes you to http://localhost:1503/show/filename for a file *filename*, which is a web address that calls a plotting method, which generates a PNG image.

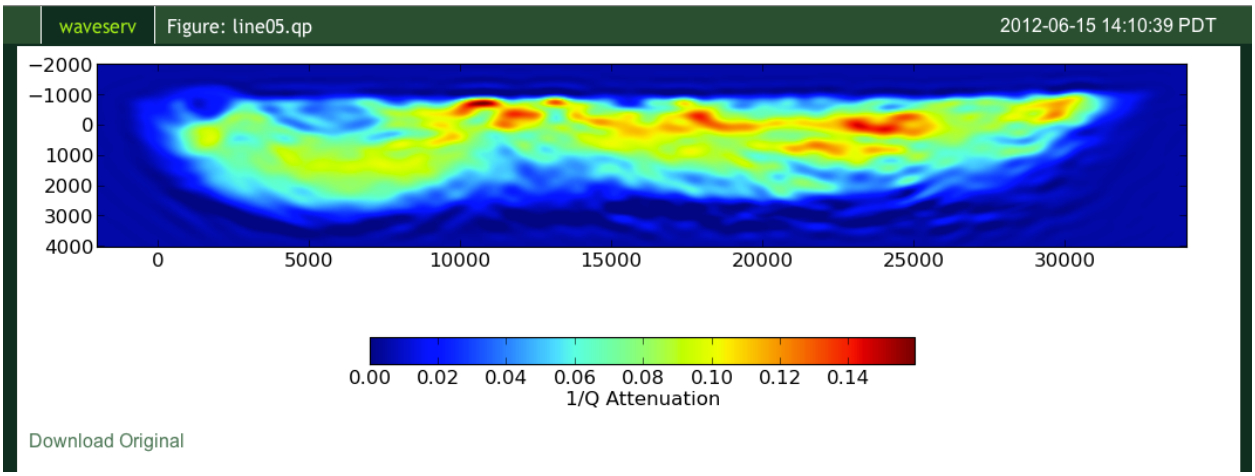


Figure 3.2: Example plot for an attenuation (1/Q) model, for the project *line05*.

CHAPTER

FOUR

# DEVELOPMENT

To date, all development of **waveserv** has been by me alone (Brendan Smithyman). I'd be more than pleased to have someone else interested in the development, so please feel free to send questions, comments or fixes to me at bsmithyman@eos.ubc.ca.

## 4.1 Main Executable

The **waveserv** executable is written in Python, and uses `optparse` to generate the *Command Line Interface*. It also creates a minimal database for the `django` test web server on the fly, using a temporary `sqlite3` database to store the settings and data. It imports the `django.core.management` interface, and uses this to construct a new `django` project in the working directory. The `pygeo.fullpy` module is used to access the **projnm.ini** file from the working directory, and read the information. This is stored in the `django` database. While the current version of the interface does not implement this, there are provisions for storing source and receiver information, which would make it possible to plot sources and receivers over model images (or to create maps and geometry plots).

The vast majority of the web-server code is handled by the `django` test web server, which is called using the `django.core.management` interface near the end of the **waveserv** source. Unless otherwise specified, temporary files are deleted at the end of the execution.

## 4.2 Server Code

The main logic of the **waveserv** program is managed by the server-side code in the **server/** directory:

**urls.py** This file determines the URL (Uniform Resource Locator) structure of the web-based interface. This determines which Python function handles each URL (e.g., /index, /show/..., /download/...).

**models.py** This file defines the database model for **waveserv**, which is necessary for storing information about the project (i.e., from the **projnm.ini** file). The database is also used to store information for caching images; if the PNG image for a given geophysical datafile is more up-to-date than the file itself, a cached version is used rather than re-generating the figure.

**views.py** This file contains code that deals with most/all of the *web-server* aspects of **waveserv**. This includes data processing for the listing page, the actual *rendering* of figures, handling downloads, caching, etc. The intent is for this to include most of the processing that is web-interface specific, and as such it makes heavy use of `django` design techniques. With the exception of some initial data management code, this file is made up of a series of *views*, which handle the server-side processing for web addresses listed in **urls.py**.

**handlers.py** This file contains code that deals with most/all of the *geophysical* aspects of **waveserv**. This includes processing files, interfacing with geophysical data formats (viz., SEG-Y), generating figures in an abstract sense, and filesystem actions. The file management is heavily dependent on the `re` module (for *regular expression*

parsing); **waveserv** knows how to handle a particular file type because of the lookup tables in this source file. In principle, this could be used to generate plots separately from the web interface (e.g., for scripted generation of figures in conjunction with document preparation).

**settings.py** This file is not actually used by **waveserv** at all, but is required if you wish to test code using the regular `django` test server.

**manage.py** This file is not actually used by **waveserv** at all, but is required if you wish to test code using the regular `django` test server.

## 4.3 Media and Templates

The web interface is built using the `django` templating language, saved in a series of ***.html** files in the **templates/** directory. These control most of the structural aspects of the GUI, and are populated with data using server-side Python scripting. The visual elements are controlled mainly by a stylesheet stored in the **media/** directory.

CHAPTER

FIVE

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

# INDEX

## Symbols



## W