

ABOVE Server Documentation

Version 1

Casey Daniel

July 10, 2015

Contents

1	Introduction	3
2	Overview and Data Flow	3
3	RTEMP	3
4	Directory Tree	3
5	TCP Sever	4
6	File Manager	4
7	CDF and Summary Plots	6
7.1	Summary Plots	6
7.2	CDF	6
8	SD reader	6
9	FTP Hosting	7
10	Shell Scripts	7
11	HTML Page	7
12	Installation	7
13	Appendix	8
13.1	Dameon	8
13.2	Crontab	8

1 Introduction

This documentation is meant to be a user guide for the ABOVE server files. These files include the TCP server, Initial Ingestion, plot generation, and shell scripts. All source files can be found on the server on `/data/vlf/src/`.

2 Overview and Data Flow

For a brief overview of the data flow of the project. There are two primary flows starting at the instrument. Data is collected by the FPGA and all data collected is saved to an unformatted SD card attached to the instrument. Selections of the data are also sent back to the server located at the University of Calgary over WiFi or 3G modems. The amount and intervals that the data is sent over 3G is dependent on the data volume, and is typically selected to be under the 6GB data plan.

Data sent over 3G is received by the TCP server, written in Python. The instrument divides the data into chunks, and then sent. This server catches the incoming data, and saves the chunk files. These files are then picked up, combined into a larger file containing the entire data set, and sends data to RTEMP where we can monitor the instruments easily. These files are plotted and converted to CDF by MATLAB scripts. These are then hosted on `data.phys.ucalgary.ca` by Darren Chaddock.

The SD card data is saved, and then sent back to the University. There is a C++ script that will extract the data from the SD card and saved locally. These files can then be copied to the server, where MATLAB scripts can once again plot and convert to CDF, and then hosted.

3 RTEMP

RTEMP is the Real Time Environment Monitoring Platform, currently run by Darren Chaddock. Data is sent from the File Manager script. This data can be seen at <http://above.rtemp.ca/> The important bits of data displayed here is the current SD fill, and the packet age. When a site goes down it's best to check on the server to make sure that the server on that port has not crashed, and is running. If so then the most likely cause is either low battery or low cell reception.

4 Directory Tree

Table 1 provides a summary of the relevant directories and information on where to find certain files.

root	<i>/data/vlf/</i>
source directory	<i>/data/vlf/src</i>
raw data	<i>/data/vlf/RawData</i>
malformed files	<i>/data/vlf/malformedFiles</i>
data tree	<i>./dataType/Year/Month/Day/Site/Hour</i>
chunk files	<i>/data/vlf/chunks/datatree</i>
full data files	<i>/data/vlf/full_files/</i>
logs	<i>/data/vlf/logs</i>
summary plots	<i>/data/vlf/summaryPlots/datatree</i>
cdf files	<i>/data/vlf/cdf/datatree</i>

Table 1: Directory List

5 TCP Sever

The Python TCP server it's self is written in Python 2.7, and is needed to run. In the source directory the server is managed by the executable of the form source *directory/tcp_server/tcp_server_PORTNO.py* where PORTNO is the desired port number. This executable looks for one of 3 commands, start, stop, or restart. (i.e. *./tcp_server_26000.py start*)

If another port is desired a few changes need to be made to the python file. The file needs to be renamed with the new port number, and near the top of the file, the variable *PORT_NUMBER* changed to match.

The server flow is described in figure 1, once I finish making it. This will detail the communications flow between the client on the FPGA and the server.

The client will send over chunk files over the internet connection, and will deposit them in the raw data directory.

6 File Manager

The File Manager, as found in ingestion software directory, and is responsible for file management. It runs the same way the tcp server does, and the same commands and format are used. This file looks through the rawData directory and collects the chunk files produced by the server and combines them into a full file, and deposit in the full file tree.

The File Manager is run on a dameon (see appendix) script. When the *clean_up()* method is called, a search of the raw data directory is done. A check for a set of data files is done. If a set contains the expected number of files, as set in the header of the files, they are combined and a full file is saved in the full file directory, and the chunks are then moved to the chunks directory.

If a full set of files is not found, and the time stamp is over one hour old, then the chunk files that are found are then moved to the malformed files directory. File Manager is also responsible for sending RTEMP Information. These packets

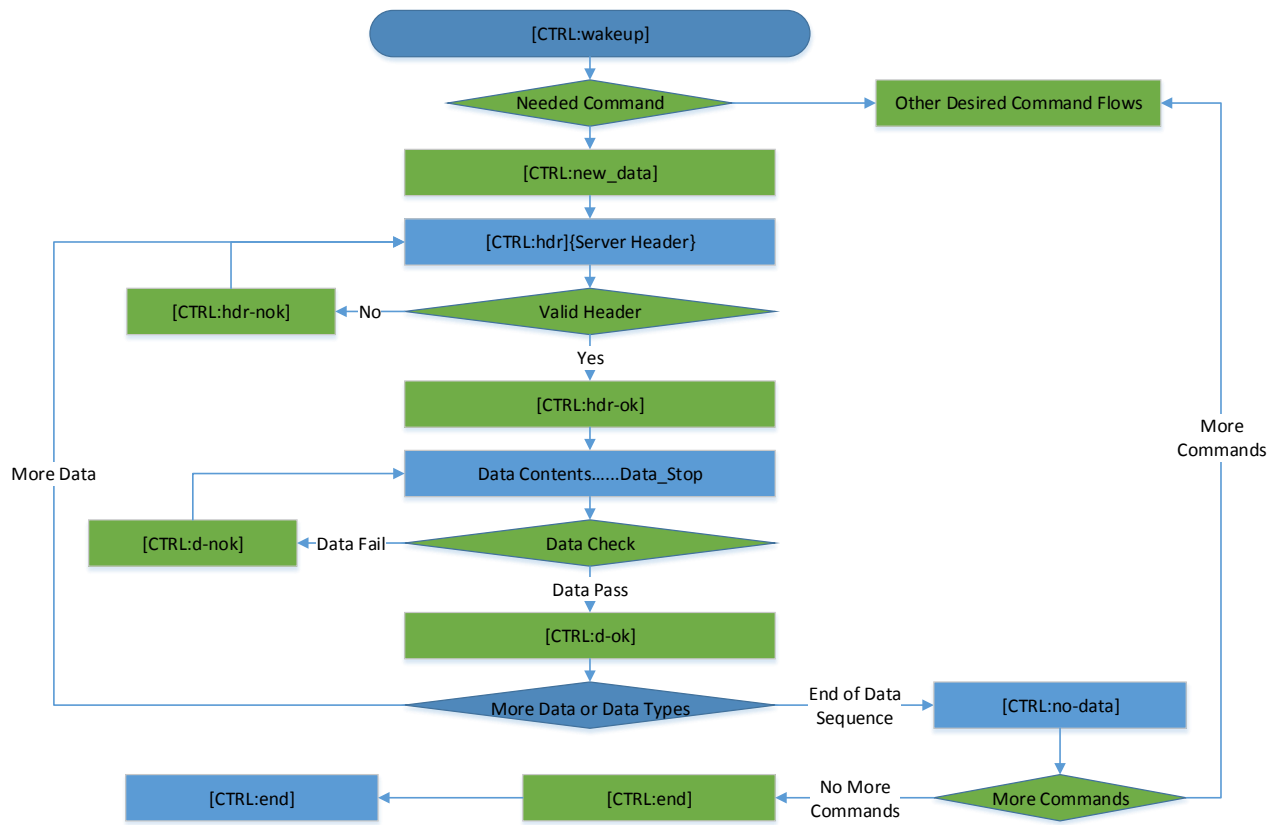


Figure 1: Visual of the connection process. Blue represents information sent by the client, and green represents information or decisions made by the server.

are formed and sent from this script, for information on format please contact Darren Chaddock.

After the files are formed, Eric Davis' script for the phase data is then started in a separate thread. Once the clean up is complete, a timer is set for 5 minutes for clean up to be called again.

7 CDF and Summary Plots

The creation of summary plots and CDF files are handled using Matlab. Going forward there will be one Matlab function for the CDF generation, summary plot. There will be one of each of these for each file type.

These functions get called by other Matlab scripts that search for the relevant file types in the data tree. Once found they will unpack the data, and pass it off to the two functions. Two versions of this script will exist. The first one will be called by a crontab (see appendix) and will only look in the current and previous days directory for any files that have not been processed. The second will look through the full data directory to find

All of these can be run from the command line using the command 'Matlab -nodesktop -nodisplay -r run "/path/to/script/scriptname.m"'. It's recommended that when running the regeneration scripts that you move the other Matlab scripts out of the directory to ensure they are not run. Matlab is a resource hog, and leaving these scripts in place can cause a bottle neck on the CPU. There is also a shell script regenerations.sh in src/shellScripts/ directory that will manage these scripts.

The crontab calls a shell script stored in src/shellScripts/. The shell scripts then call the Matlab scripts.

7.1 Summary Plots

The current plots are two spectrograms, one of each antenna, of the full 0-75kHz range. Below is a spectrogram of 0-10kHz, followed by the plotted time series.

More types of plots will be added as new types of data are available.

7.2 CDF

The CDF files are ISTP compliant, and follow the required attributes and naming conventions. Again only one type of data is currently hosted as only one is available. More details about CDF can be found at <http://cdf.gsfc.nasa.gov/> and ISTP compliance at http://spdf.gsfc.nasa.gov/istp_guide/istp_guide.html

8 SD reader

The SD reader application is a c++ script that will read the contents of the SD card, and place them in the directory tree with a given root directory. To

run this program in the command line `"../sdCard if=/path/to/InputFile.dat od=/root/of/directory/tree"`. There are also 2 optional commands. `-hf` will fix the header values. The early versions of the SD card had a few fields with incorrect information. `-sd SD_ SIZE` will allow you to change the SD card size. The default is 128GB, however if the card is a different size, then a different one can be use. It's recommended to direct the output to a log file by attaching `">log.txt"` to the end of the command.

Before running, the SD card should be copied into a separate file. The mounting point for the SD on a unix system can be found by looking through `/dev/` using `diskutil`. Once the mount point is found, the command `dd if=/path/of=out/file.dat` can be used to transfer the documents. The `/out/file.dat` can then be fed into the SD reader.

9 FTP Hosting

Files and images will be hosted by Darren Cahddock via anonymous FTP.

10 Shell Scripts

There are a few shell scripts. These primarily handle the generation of summary plots. There is also a reboot script that is called by the crontab on boot to restart the server and the file manger. There is also a `root_reboot` shell script that is called by the root user on boot to start the FTP server.

11 HTML Page

In the summary plots page directory sit a simple HTML page to be able to view the summary plots from the browser. It consists of a simple form with the year, month, day, year, hour, site, and a few buttons to be able to flip through both the availability charts and the summary plots.

The year is not automatically generated, and every year the year needs to be added to the form.

Below the form is the java script that will react to the buttons and pull up the requested image from the FTP server, and display an error message if they are not available.

12 Installation

To install the server software on a new server, all the files should be placed in a source directory. All the directories in the directory table should be worked out on the new server, and updated in any python and Matlab script. These directories are commonly named and labeled near the top of the files.

Ensure both python and Matlab are installed. The python packages numpy, scipy, and matplotlib also need to be installed.

The TCP server should be started on the desired ports, and the sites can be remotely configured with the updated ports and IP addresses. Following which the file manager should be started.

Cronjob's calling the shell scripts should be set up. It's recommended that the availability plots and the summary plots generation script be timed so they are called at different times to prevent Matlab becoming a huge resource hog. The current set up can be seen in the crontab section, however this can be tailored as desired.

The HTML page also needs to be updated with the new IP address. It's listed as a constant in the java script. The site is currently hosted by Darren Chaddock, and the new HTML page should be sent to him to be updated.

13 Appendix

13.1 Dameon

When a program is run normally from the command line, the process is attached to the terminal, meaning that when you close the terminal the process is also stopped. This obviously causes a problem when trying to run programs indefinitely. To take the process away from the terminal and pass it into the background is called dameon.

In the python scrips I have found a class that through a simple import and modification of a few components will handle this task easily. More on how this is done can be found from the source.

13.2 Crontab

Here is the current crontab setup to be used as an example. For more options on crontab, please consult their documentation.

```
Main User Crontab
1,30,50 * * * * /data/vlf/src/shellScripts/matlabSummaryPlot.sh
&> /data/vlf/logs/plottingscript.log
@reboot /data/vlf/src/shellScripts/reboot.sh &>/dev/null
15,45 * * * * /data/vlf/src/shellScripts/matlabAvailabilityPlotter.sh
&> /data/vlf/logs/avail_log.log
```

If you wish to see the output of these scripts, the file can be adjusted.