ChainWatch

File Processing Software Documentation

6/26/2020

V0.4

TABLE OF CONTENTS

ChainWatch (Beta)

ChainWatch (Beta)

OVERVIEW

ABOUT

This specification outlines the structure and operation of a collection of software written for the Raspberry Pi. This software is intended to provide service workers with an overview of the information captured by SICK Cameras when inspecting chain links at Honda manufacturing facilities. Upon detecting FTP image transfers into a user's home directory, it will store the details of the image encoded in the filename and transfer them to another folder for long term storage. These images will then be viewable through a website hosted on the Raspberry Pi, as well as an overview of pass/fail rates and alerts for links with high failure rates.

SOFTWARE REQUIREMENTS

In order to operate the provided programs, the following third-party software packages must be installed on the Raspberry Pi:

- vsftpd (3.0.3)
- MariaDB (10.3.22)
- python (3.7.0)

Note: Version numbers indicate the versions installed on the system and are not indicative of forwards or backwards compatibility.

PYTHON PACKAGE REQUIREMENTS

The python packages included in ChainWatch requires some third-party packages. These must be installed in either the system or root user's package directory:

- flask
- inotify
- mysql-connector-python
- qrcode

OPERATION

The custom software for this suite is provided as two Python modules, one for processing the FTP images (`filewatcher`) and one for hosting the website (`webserver`). These are each setup as systemd services, meaning the scripts will launch in the background on startup and remain active unless they are disable. All necessary parameters for function are provided in the service file.

PYTHON MODULES

ABOUT

In Python, modules are packages which can be run like a program. ChainWatch relies on two modules, `filewatcher` and `webserver`, for its functionality. `filewatcher` is responsible for handling incoming FTP files and logging them in the database, while `webserver` provides the website as a front end for viewing the data. Modules are run with the command. It is accessible at the following URL:

sudo python3 -m <module> <args>

where `<module>` is the name of the module and `<args>` are any command line arguments. All necessary arguments are handled by the service file.

DEPENDENCIES

Both modules rely on two first party packages, located alongside the modules:

- **`config`**: This package handles configuration values. It ingests a JSON file and stores the associated values. These values can then be reloaded during execution or modified and saved back to the file.
- **`dbconnection`**: This package handles interfacing with the database, removing the need for the modules to understand the layout and structure

FILEWATCHER

QUICK DETAILS

Location: `/usr/local/lib/python3.7/dist-packages/filewatcher`
Service file: `/etc/systemd/system/filewatcher.service`
Service user: `sick1`

ABOUT

`filewatcher` is python module which runs as a systemd service on the Raspberry Pi. Its function is to detect new files in the home directory of the camera's account, record that file in the database, and move it to a specified directory for long term storage. This program is also in charge of managing the database, i.e. removing image files and records when the counts exceed the limits specified in the configuration file.

Note: This module has the capacity to run all tasks in background threads, but this feature has been disabled due to the limited processing power of the Raspberry Pi. Processing images synchronously provides enough speed for normal operation while ensuring the stability of the database.

WEBSERVER

QUICK DETAILS

Location: `/usr/local/lib/python3.7/dist-packages/webserver`
Service file: `/etc/systemd/system/webserver.service`
Service user: `pi`
Port: `5000`

ABOUT

`webserver` is python module which runs on the Raspberry Pi to provide easy access to information about the link diagnostic information. It is accessible at the following URL:

http://<hostname>:5000

PAGES

Index: /

The homepage shows a list of important statistics regarding link inspection.

Grid: /grid

This page displays every link as a square in a large grid, which when clicked will open up the details page for that link (described below). Each square is color coded to display that link's pass rate, with blue representing a 100% pass rate and red representing a 0% pass rate. When the mouse hovers over a square, a pop-up bubble shows the exact percentage. The squares can be sorted be link id or pass rate.

Link Details: /<num>

This page shows a table of the link's images, when they were taken, and whether they failed. The details page for any chain link can be visited by going to that chain link's ID number. For instance, the webpage for the chain link #452 would be

http://<hostname>:5000/452

Database administration commands are provided for deleting individual images or clearing the link from the database entirely. Additionally, a button labeled "View Images" opens up the slideshow page for the link (described below).

Image Slideshow: /slideshow/<num>

This page allows a user to browse a link's images. The left and right images for each loop are shown as slides, and buttons let the user navigate backwards and forwards through each loop, inspecting the link over time.

Settings Configuration: /configure

This page displays the configuration values for `filewatcher` and `webserver` in an editable form. This allows the values to be modified in the browser without needing to directly interface with any files on the Raspberry Pi. It also provides functions for managing the image database, such as resetting the contents. This page also provides administrative commands like rebooting the webserver and clearing the database.

Documentation: /spec

This link provides a copy of this file, viewable in browser for any operational instructions or documentation for future software support.

DATABASE

CONFIGURATION

User: `ChainWatch`
Password: `service`
Database: `ChainWatch`

ABOUT

The details of each link are stored in a MySQL database on the Raspberry Pi. This database is managed using MariaDB. The database contains two tables, `images` and `past_failures`, which store information about the processed images.

The database's time zone must be set to UTC, as this is what `webserver` expects the **time** column to be in for both tables.

IMAGES

ABOUT

This table stores the most recent images from the cameras. Once a maximum number of images (set by a configuration value in `filewatcher`) has been exceeded, the oldest images for that camera are removed from the database. If these inspections were failures, they are transferred to `past_failures`.

DOCUMENT FORMAT

The information of each link is stored in the collection with the following format:

- **img_id** `int(11)`: The primary key of the database. Autoincrementing.
- **link_id** `int(11)`: The ID of the link.
- **loop_count** `int(11)`: The index of loop the image was taken on.
- **camera** `varchar(100)`: The name of the camera that took the image.
- **passed** `tinyint(1)`: Treated as a Boolean value, a '0' signals a crack was found in the image.
- **time** `datetime`: The time that the file was added to the database. This is automatically added.
- **filepath** `varchar(100)`: The absolute path to the image file.

PAST_FAILURES

ABOUT

This table stores a link's failed inspections after they have been trimmed from `images`. These images do not factor into a link's failure rate. Like `images`, these entries will be removed from the database after a link's maximum number has been reached.

DOCUMENT FORMAT

The information of each link is stored in the collection with the following format:

- **img_id** `int(11)`: The primary key of the database. Autoincrementing.
- **link_id** `int(11)`: The ID of the link.
- **loop_count** `int(11)`: The index of loop the image was taken on.
- **camera** `varchar(100)`: The name of the camera that took the image.
- **passed** `tinyint(1)`: Treated as a Boolean value, a '0' signals a crack was found in the image. Since this table collects failed inspections, this will always be `False`.
- **time** `datetime`: The time that the file was added to the database. This is automatically added.
- **filepath** `varchar(100)`: The absolute path to the image file.

SYSTEMD

ABOUT

systemd is a Linux software suite used for managing system components. ChainWatch uses it to automatically run the necessary python modules on startup. In the event that any modules need to be restarted or shutdown, the commands below can be used. These instructions, or "services", are defined by `.service` files, located at `/etc/systemd/system/` and named after the module they are designed to run.

Note: `<SERVICE>` refers to the filename WITHOUT the extension, which is the module's name.

COMMANDS

SERVICE COMMANDS

`sudo systemctl start <SERVICE>`

Start the execution of an inactive service. This has no effect if the service is already running.

`sudo systemctl restart <SERVICE>`

Quit and restart the execution of a service

`sudo systemctl stop <SERVICE>`

Quit the execution of service

`sudo systemctl status <SERVICE>`

Print the current status of the service

`sudo systemctl enable <SERVICE>`

Set the service to run on startup. This does not start the service if it is inactive.

`sudo systemctl disable <SERVICE>`

Sets the service to no longer run on startup. This does not prevent the service from being manually started later or quit the service if it was already running.

FILES

STORAGE

After processing, `filewatcher` moves all image files to the '`/srv/ftp`' directory. They are stored with their original filename appended with the current POSIX timestamp.

RETRIEVAL

Files are served from the Raspberry Pi's self-hosted website. They are stored under the URL:

http://<hostname>/img/<filename>

where `<hostname>` is the Raspberry Pi's IP address on the network and `<filename>` is the image's filename. In instances where the filename is unknown, an index of every image for a given link is listed on the webpage:

http://<hostname>/<link_id>

where `<link_id>` is the link's numerical id.

ChainWatch (Beta)