pvWebMonitor Documentation

Release 2016.1025.0-1-ge1796bf

Pete R. Jemian

Mar 16, 2017

Contents

1 Contents 3

2 Indices and tables 27

Python Module Index 29

post EPICS PVs to read-only (static) web page(s)

This package provides a background service that monitors EPICS PVs and writes them into customized HTML files in a WWW server directory. The service can be started and stopped by a manage.csh script for automated startup in a cron task or at system startup.

author Pete R. Jemian

email jemian@anl.gov

docs http://pvWebMonitor.readthedocs.io

git https://github.com/prjemian/pvWebMonitor.git

PyPI https://pypi.python.org/pypi/pvWebMonitor

discussion

version 2016.1025.0

release 2016.1025.0-1-ge1796bf

published Mar 16, 2017

CHAPTER 1

Contents

Overview

The basic flow of data from EPICS to the WWW site is described in the following diagram:

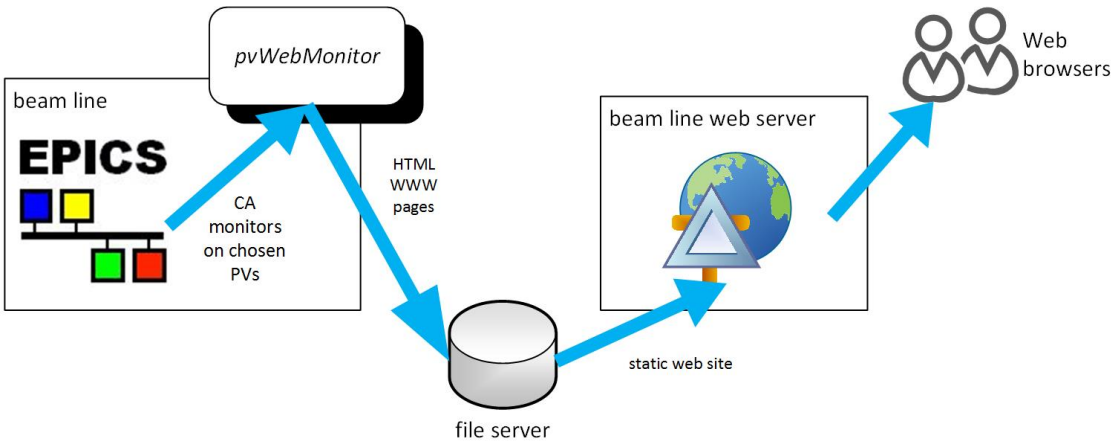


Fig. 1.1: flow of data from EPICS to WWW site

The **pvWebMonitor** service is run on a computer in the same subnet as the EPICS system to be monitored. All configuration files and other resources are placed in a single project directory. **pvWebMonitor** places an EPICS Channel Access monitor on each PV in the *pvlist.xml* file and stores updates in-memory. Periodically, as specified in *config.xml*, **pvWebMonitor** writes the PV values from memory to an XML file (named *rawdata.xml*) in the project directory. Once that XML file is written, **pvWebMonitor** uses *rawdata.xml*[1] with each of the XSLT files[2] in the project directory to create a corresponding HTML file in the project directory. The complete list of HTML files is written into

[1] The *rawdata.xml* file contains all the EPICS PV values, as well as some additional metadata useful in building the WWW site.

[2] Each XSLT files (`*.xsl`) contains the layout of a single HTML page, with additional markup to display the current EPICS PV values (and metadata). The EPICS PV data is provided in *rawdata.xml*.

an *index.html* file in the project directory. Finally, all content in the project directory (except for the *config.xml* file) is copied to the WWW site directory. (Only new content is copied, files that do not change are not re-copied.)

It is important to note the WWW site is written as a *static web site* so that it provides no opportunity to change values in the EPICS system being monitored.

Also, since some browsers do not have XML parsers and thus cannot render XSLT[3], all HTML files are created by **pvWebMonitor**.

Examples

Example (very brief[4]) *rawdata.xml* file:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="short.xsl"?>
<pvWebMonitor version="1">
  <written_by>pvWebMonitor/PvWatch</written_by>
  <datetime>2015-01-15 16:48:36</datetime>
  <pv id="VDM_Stripe" name="prj:m1.RBV">
    <name>prj:m1.RBV</name>
    <id>VDM_Stripe</id>
    <description>VDM_Stripe motor</description>
    <timestamp>2015-01-15 15:30:16.837633</timestamp>
    <counter>2</counter>
    <units>deg</units>
    <value>-1.510</value>
    <raw_value>-1.51</raw_value>
    <format>%.3f</format>
  </pv>
</pvWebMonitor>
```

Each XSLT file describes the format of an HTML page. The XSLT file uses XSL markup to pick EPICS PV values from the XML file. Here's an example that shows the value of the PV `prj:m1.RBV`. (The *id* `VDM_Stripe` is used here as a symbolic reference.):

```
<xsl:value-of select="//pv[@id='VDM_Stripe']/value"/>
```

Here's how to show when that PV value was last updated:

```
<xsl:value-of select="//pv[@id='VDM_Stripe']/timestamp"/>
```

Here's how to show when the EPICS PV data was last posted to the WWW site:

```
<xsl:value-of select="/pvWebMonitor/datetime"/>
```

The XSLT language has many additional functions available to help as your page designs get more complex. Look at the supplied *livedata.xsl* file for additional examples. There are good tutorial web sites available, such as: http://www.w3schools.com/xsl

Here's an example XSLT file using these example lines above (line breaks, `<br />`, were added for clarity):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
```

[3] http://www.w3schools.com/xsl/xsl_server.asp

[4] A more complete example is provided in the *Example* section.

```
    description="simple example XSLT to EPICS PV value">

    <xsl:template match="/">
        <html>
            <body>
                EPICS PV: <xsl:value-of select="//pv[@id='VDM_Stripe']/name"/><br />

                PV value: <xsl:value-of select="//pv[@id='VDM_Stripe']/value"/><br />

                PV last updated: <xsl:value-of select="//pv[@id='VDM_Stripe']/
↪timestamp"/><br />

                HTML file written: <xsl:value-of select="/pvWebMonitor/datetime"/>
            </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```

The XSLT transformation using the XML file above looks like:

```
<html><body>
                EPICS PV: prj:m1.RBV<br>

                PV value: -1.510<br>

                PV last updated: 2015-01-15 15:30:16.837633<br>

                HTML file written: 2015-01-15 16:48:36</body></html>
```

Which shows in a browser:

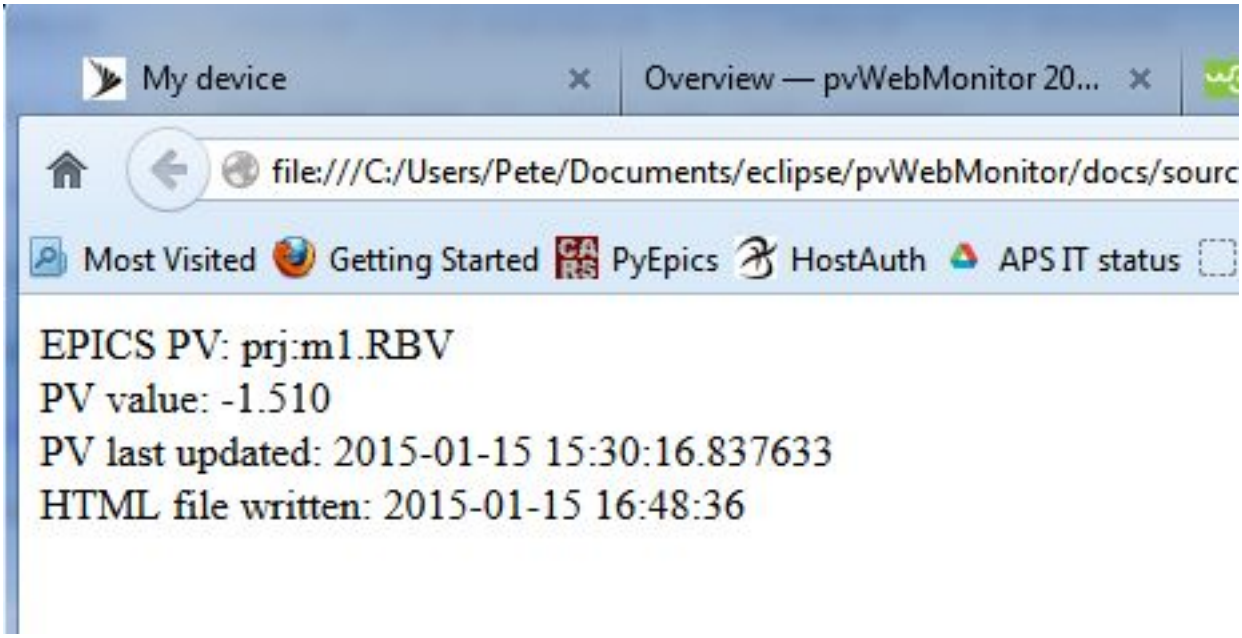


Fig. 1.2: Example HTML web page from above.

Configuration

These are the steps needed to get the **pvWebMonitor** service running on your workstation.

1. install the *pvWebMonitor* package into your Python environment
2. setup the project configuration directory
3. identify the web server directory to be used
4. edit config.xml
5. identify the list of EPICS PVs
6. edit pvlist.xml
7. customize the livedata display file: edit livedata.xsl
8. run the config.xml file
9. watch the log_data.txt file in the project directory

These steps are described in the following sections:

Installation

If you need to install the *pvWebMonitor* package, follow these terse instructions:

```
pip install pvWebMonitor
```

Alternatively, you could clone the GitHub project:

```
git clone https://github.com/prjemian/pvWebMonitor.git
```

Once the installation is complete, the *pvWebMonitor* executable should be ready to use.

Setup a new project directory

The *pvWebMonitor* service is configured by files configured by the user in a *project directory*.

To get default versions of the files, run this command:

```
mkdir path/to/project/directory
pvWebMonitor --setup path/to/project/directory
cd path/to/project/directory
```

where *path/to/project/directory* is either a partial, relative, or absolute path to an existing directory to be used. Once this command has run, the files will be copied to the designated directory. If files with these names already exist, *pvWebMonitor* will stop with an error report and not overwrite the existing files.

file	How is it used?
config.xml	defines user settings for the program
pvlist.xml	declares list of EPICS PVs to be monitored
pvlist.xsl	for easy display of pvlist.xml
livedata.xsl	user-customized display
rawdata.xsl	standard display of all monitored EPICS PVs
manage.sh	shell script to manage the background task

Each of these files will be explained in the coming sections.

The *config.xml* file

The *config.xml* file defines constants needed by the program.

Definitions for each item listed in the table under *pvWebMonitor.read_config.read_xml()* are provided, such as this example:

Example `config.xml` file.

```
<?xml version="1.0" ?>
<pvWebMonitor__config version="1.0.1">

  <!-- PVs to be monitored -->
  <var name="PVLIST_FILE"               value="pvlist.xml" />

  <!-- absolute directory path to WWW site on local file system -->
  <var name="LOCAL_WWW_LIVEDATA_DIR"    value="./" />

  <!-- writing messages to log file -->
  <var name="LOG_INTERVAL_S"            value="300"  type="float" />

  <!-- updates to HTML pages -->
  <var name="REPORT_INTERVAL_S"         value="10"  type="float" />

  <!-- sleeps at end of main loop -->
  <var name="SLEEP_INTERVAL_S"          value="0.1"  type="float" />

  <!-- another logging message interval -->
  <var name="MAINLOOP_COUNTER_TRIGGER"  value="10000" type="int" />

  <!-- files with these name patterns (glob style match)
       will be copied from project dir to www dir
       (upper/lower case variants will be searched as well)

       uses Python ``fnmatch.filter()``
  -->
  <pattern value="*.html" /> <!-- web pages -->
  <pattern value="*.gif" />  <!-- images -->
  <pattern value="*.jpeg" /> <!-- images -->
  <pattern value="*.jpg" />  <!-- images -->
  <pattern value="*.png" />  <!-- images -->
  <pattern value="*.xsl" />  <!-- XML stylesheets -->
  <pattern value="*.txt" />  <!-- text -->
  <pattern value="*.pdf" />  <!-- documentation -->

</pvWebMonitor__config>
```

To use the *pvWebMonitor* service effectively, it is likely you will only need to edit the value for *LOCAL_WWW_LIVEDATA_DIR* which defines the location of the directory used by the web server to serve content.

Preamble

The *config.xml* must be "well-formed XML".

The first line of the file is *always*:

```
<?xml version="1.0" ?>
```

This line declares this to be a file that should be well-formed XML according to the version 1.0 standard.

Root Tag

All well-formed XML files have a single element at the outermost (root) level of the file. In the *config.xml* file, the root element is **pvWebMonitor__config**. Note the closing `</pvWebMonitor__config>` tag at the end of the file.

A version attribute describes this file adheres to the `version="1.0"` definition of *config.xml* files. That definition is described in the XML Schema file *config.xsd* provided in the source code package.

This XML Schema definition is used to validate the *config.xml* when it is read. If there are problems, the first problem discovered will be reported.

The *pvlist.xml* file

The complete list of EPICS Process Variables to be monitored is declared in the *pvlist.xml* file.

Preamble

The *pvlist.xml* must be "well-formed XML". (Google for this term to become informed what this means.)

The first lines of the file are *always*:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="pvlist.xsl" ?>
```

The first line declares this to be a file that should be well-formed XML according to the version 1.0 standard. The second line provides a convenience definition for visualizing the *pvlist.xml* file in a browser that uses the *pvlist.xsl* file to format the XML content into something that humans can more easily read. To do this, the *pvlist.xsl* file must be in the same directory as the *pvlist.xml* file.

Root tag

All well-formed XML files have a single element at the outermost (root) level of the file. In the *pvlist.xml* file, the root element is **pvwatch**. Note the closing `</pvwatch>` tag at the end of the file.

A version attribute describes this file adheres to the `version="1.0"` definition of *pvlist.xml* files. That definition is described in the XML Schema file *pvlist.xsd* provided in the source code package.

This XML Schema definition is used to validate the *pvlist.xml* when it is read. If there are problems, the first problem discovered will be reported.

Inside the root tag, *EPICS_PV* elements describe the PVs to be monitored. Additionally, *definition* tags are provided to describe the terms used.

Describe a PV to Monitor

Each PV to be monitored is declared in the XML file using a line such as this example:

```
<EPICS_PV
  PV="ioc:xps:c0:m1.RBV"
  mne="mr"
  description="motor MR, degrees"
  />
```

This says to monitor the EPICS process variable named `ioc:xps:c0:m1.RBV` and to associate that value with the mnemonic named `mr`. The description text `motor MR, degrees` can be used in displays for this value. The tag `EPICS_PV` describes this as a PV declaration. It must appear in uppercase letters. A complete list of terms is described in the section below: *ref*:pvlist.terms‘.

At minimum, it is required to provide the *PV*, *mne*, and *description* attributes.

The order of the attributes is not important, they can be given in any order. Also, the spacing between attributes is not important. The entire *EPICS_PV* element can be specified on one line or broken across several lines.

Keep the description text short. Longer descriptions, including those with line breaks, are less useful in creating display screens.

The closing tag

Note that `/>` is used to close the `EPICS_PV` element. It is equivalent to use:

```
<EPICS_PV
  PV="ioc:xps:c0:m1.RBV"
  mne="mr"
  description="motor MR, degrees"
  ></EPICS_PV>
```

but this is not advised since no content is allowed for EPICS_PV elements, only attributes.

Terms

attribute	definition
mne	one-word mnemonic reference used in python and xslt code (mne should be unique for each EPICS_PV)
PV	EPICS process variable name (must be used in only one EPICS_PV)
description	useful text informative to others
display_format	(optional, default="%s") PVs will be formatted for display with this string
ignore	(optional, default="false") this PV is ignored if value is not “false”

These two declarations are equivalent:

```
<EPICS_PV PV="ioc:xps:c0:m1.RBV" description="motor MR, degrees" display_format="%.6f
↪" mne="mr"/>
```

```
<EPICS_PV
  PV="ioc:xps:c0:m1.RBV"
  description="motor MR, degrees"
  display_format="%.6f"
  mne="mr"
  />
```

Removing declarations

Sometimes, it is necessary to stop watching a certain PV. There are three ways to do this. It can be commented out using XML comments, it can be marked to *_ignore_* it, or the declaration could be deleted. We'll describe the first two cases.

Comment out in XML

To comment out using an XML comment (<!-- -->), take this code:

```
<EPICS_PV PV="ioc:m1" mne="m1" description="motor 1" />
```

and surround it with XML comment tags, such as:

```
<!--
<EPICS_PV PV="ioc:m1" mne="m1" description="motor 1" />
-->
```

XML comment tags can be used to block out many *EPICS_PV* declarations at once.

Marking with *_ignore_* attribute

To mark a single *EPICS_PV* declaration to be ignored, take this code:

```
<EPICS_PV PV="ioc:m1" mne="m1" description="motor 1" />
```

and add the `_ignore_="true"` attribute, such as:

```
<EPICS_PV _ignore_="true" PV="ioc:m1" mne="m1" description="motor 1" />
```

The *_ignore_* attribute can be given in any order. The value *true* may be upper or lower case but must be enclosed by double quotes.

Each PV to be ignored using the *_ignore_* attribute must have its own *_ignore_* attribute. You cannot mark a whole block of *EPICS_PV* elements with a single *_ignore_* attribute.

Example *pvlist.xml* file

An example of such a file is shown below.

Example *pvlist.xml* file. You can edit this file with a text editor.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="pvlist.xsl" ?>

<!-- You can edit this file with a text editor -->

<pvwatch version="1.0">

  <EPICS_PV PV="ioc:alldone"
      description="IOC motors moving"
      mne="ioc_alldone"
```

```
    />

  <EPICS_PV PV="APS:SRcurrentAI"
      description="APS storage ring current, mA"
      display_format="%.2f"
      mne="SR_current"
      />

  <EPICS_PV PV="ID:Energy"
      mne="Und_E"
      description="ID E, keV"
      display_format="%.4f"
      />
  <EPICS_PV PV="dcm:BraggEAO"
      mne="DCM_E"
      description="DCM E, keV"
      display_format="%.4f"
      />

  <EPICS_PV PV="PSS:STA_A_FES_OPEN_PL.VAL"
      description="white shutter opened"
      mne="white_shtr_opened"
      />
  <EPICS_PV PV="PSS:STA_B_SBS_OPEN_PL.VAL"
      description="mono shutter opened"
      mne="mono_shtr_opened"
      />
  <EPICS_PV PV="PSS:D_BEAM_READY"
      _ignore_="true"
      description="PSS: D Beam Ready"
      mne="D_beam_ready"
      />

</pvwatch>
```

The *rawdata.xsl* file

The *rawdata.xsl* file is used to format the complete list of monitored EPICS PV values into an HTML file for display from the web server. It should not be necessary to edit this file.

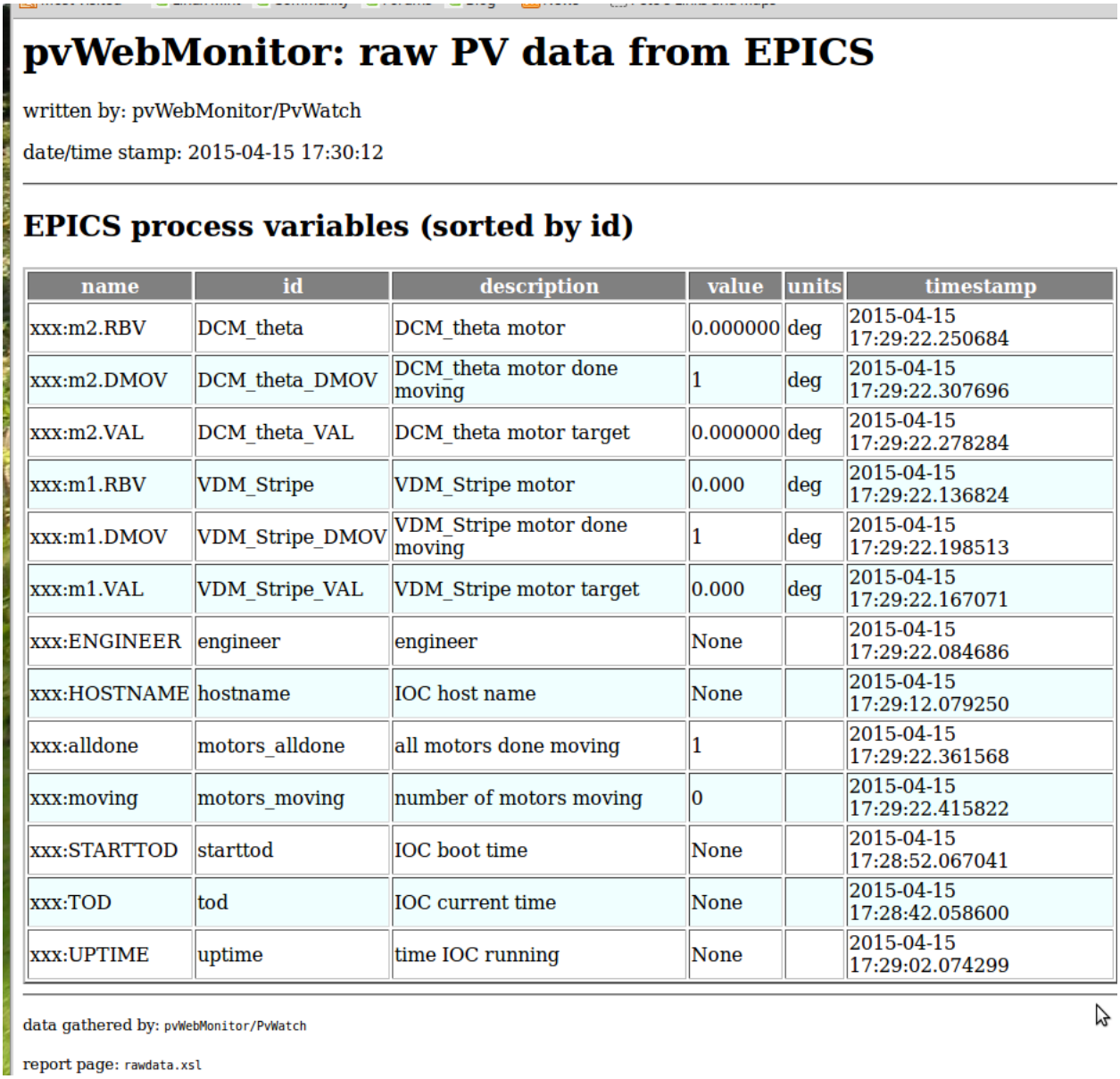
The *livedata.xsl* file

The *livedata.xsl* file is used to format select monitored EPICS PV values into an HTML file for display from the web server. It is intended that the user will modify this file for the desired display features.

Note: need to provide instructions and references on editing XSLT for displaying PVs.

The *manage.sh* file

explanation ...



pvWebMonitor: raw PV data from EPICS

written by: pvWebMonitor/PvWatch

date/time stamp: 2015-04-15 17:30:12

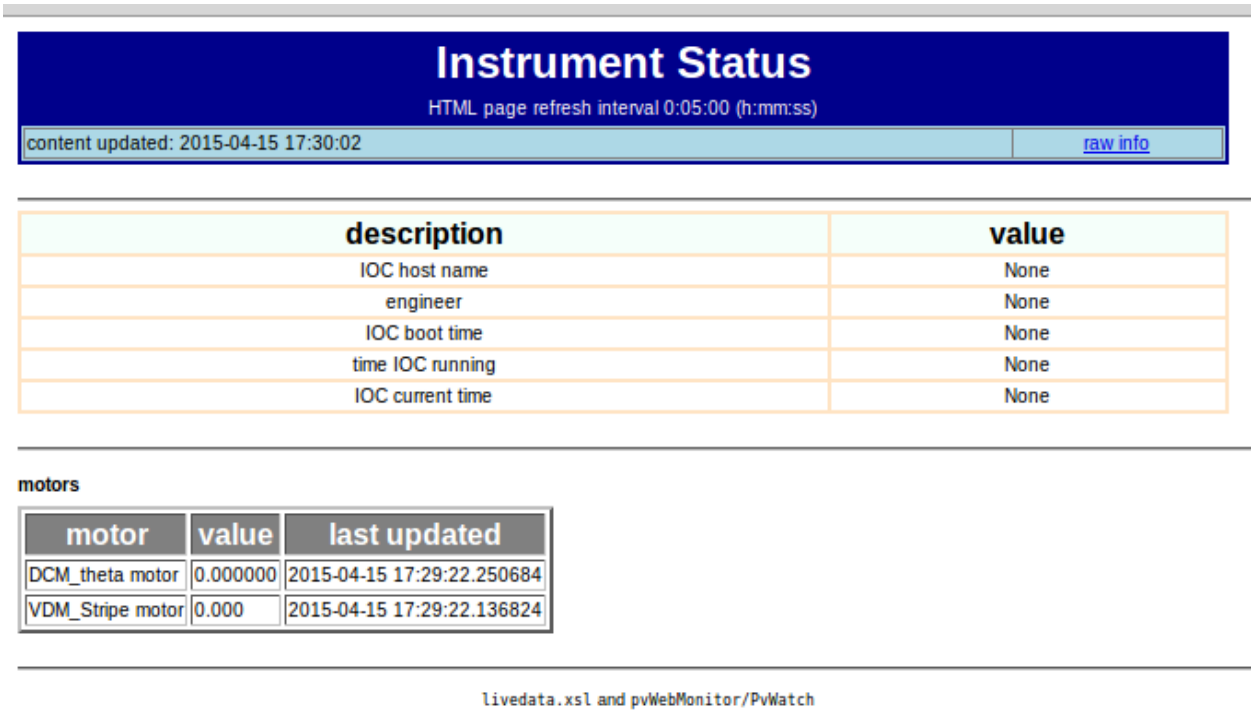
EPICS process variables (sorted by id)

name	id	description	value	units	timestamp
xxx:m2.RBV	DCM_theta	DCM_theta motor	0.000000	deg	2015-04-15 17:29:22.250684
xxx:m2.DMOV	DCM_theta_DMOV	DCM_theta motor done moving	1	deg	2015-04-15 17:29:22.307696
xxx:m2.VAL	DCM_theta_VAL	DCM_theta motor target	0.000000	deg	2015-04-15 17:29:22.278284
xxx:m1.RBV	VDM_Stripe	VDM_Stripe motor	0.000	deg	2015-04-15 17:29:22.136824
xxx:m1.DMOV	VDM_Stripe_DMOV	VDM_Stripe motor done moving	1	deg	2015-04-15 17:29:22.198513
xxx:m1.VAL	VDM_Stripe_VAL	VDM_Stripe motor target	0.000	deg	2015-04-15 17:29:22.167071
xxx:ENGINEER	engineer	engineer	None		2015-04-15 17:29:22.084686
xxx:HOSTNAME	hostname	IOC host name	None		2015-04-15 17:29:12.079250
xxx:alldone	motors_alldone	all motors done moving	1		2015-04-15 17:29:22.361568
xxx:moving	motors_moving	number of motors moving	0		2015-04-15 17:29:22.415822
xxx:STARTTOD	starttod	IOC boot time	None		2015-04-15 17:28:52.067041
xxx:TOD	tod	IOC current time	None		2015-04-15 17:28:42.058600
xxx:UPTIME	uptime	time IOC running	None		2015-04-15 17:29:02.074299

data gathered by: pvWebMonitor/PvWatch

report page: rawdata.xsl

Fig. 1.3: Example *rawdata.html*, generated from the *rawdata.xsl* file.



Instrument Status

HTML page refresh interval 0:05:00 (h:mm:ss)

content updated: 2015-04-15 17:30:02 | raw info

description	value
IOC host name	None
engineer	None
IOC boot time	None
time IOC running	None
IOC current time	None

motors

motor	value	last updated
DCM_theta motor	0.000000	2015-04-15 17:29:22.250684
VDM_Stripe motor	0.000	2015-04-15 17:29:22.136824

livedata.xsl and pvWebMonitor/PvWatch

Fig. 1.4: Example *index.html*, generated from the *livedata.xsl* file.

Example shell script to manage the `pvWebMonitor` process either as a startup or a background daemon.

```
#!/bin/bash
# init file for pvWebMonitor
#
# chkconfig: - 98 98
# description: pvWebMonitor WWW page update script for NAME_YOUR_SYSTEM_HERE
#
# processname: pvWebMonitor_MAKE_THIS_NAME_UNIQUE


PROJECT_DIR=/tmp/pv
MANAGE=${PROJECT_DIR}/manage.sh
LOGFILE=${PROJECT_DIR}/log-manage.txt
PIDFILE=${PROJECT_DIR}/pid.txt
CONFIGFILE=${PROJECT_DIR}/config.xml
EXECUTABLE_SCRIPT=/home/oxygen/JEMIAN/Apps/anaconda/bin/pvWebMonitor
RETVAL=0


get_pid(){
    cd ${PROJECT_DIR}
    PID=`/bin/cat ${PIDFILE}`
    return $PID
}


check_pid_running(){
```

```
    get_pid
    if [ "${PID}" == "" ]; then
        # no PID in the PIDFILE
        RETVAL=1
    else
        RESPONSE=`ps -p ${PID} -o comm=`
        if [ "${RESPONSE}" == "pvWebMonitor" ]; then
            # PID matches the pvWebMonitor profile
            RETVAL=0
        else
            # PID is not pvWebMonitor
            RETVAL=1
        fi
    fi
    return $RETVAL
}


start(){
    cd ${PROJECT_DIR}
    ${EXECUTABLE_SCRIPT} ${CONFIGFILE} 2>&1 >> ${LOGFILE} &
    PID=$!
    /bin/echo ${PID} > ${PIDFILE}
    /bin/echo "# [$0 `/bin/date`] started ${PID}: ${EXECUTABLE_SCRIPT}" 2>&1
↪>> ${LOGFILE} &
    /bin/echo "# [$0 `/bin/date`] started ${PID}: ${EXECUTABLE_SCRIPT}"
}


stop(){
    get_pid
    check_pid_running

    if [ $RETVAL == 1 ]; then
        /bin/echo "# [$0 `/bin/date`] not running ${PID}: ${EXECUTABLE_
↪SCRIPT}" 2>&1 >> ${LOGFILE} &
    else
        kill ${PID}
        /bin/echo "# [$0 `/bin/date`] stopped ${PID}: ${EXECUTABLE_SCRIPT}"
↪2>&1 >> ${LOGFILE} &
        /bin/echo "# [$0 `/bin/date`] stopped ${PID}: ${EXECUTABLE_SCRIPT}"
    fi
    /bin/cp -f /dev/null ${PIDFILE}
}


restart(){
    stop
    start
}


checkup(){
    #=====================
    # call periodically (every 5 minutes) to see if pvWebMonitor is running
    #=====================
    #        field      allowed values
    #      -----      --------------
```

```
    #       minute      0-59
    #       hour        0-23
    #       day of month    1-31
    #       month       1-12 (or names, see below)
    #       day of week     0-7 (0 or 7 is Sun, or use names)
    #
    # */5 * * * * /tmp/pv/manage.sh checkup 2>&1 > /dev/null


    get_pid
    check_pid_running
    if [ $RETVAL == 0 ]; then
        echo "# [$0 `/bin/date`] running fine, so it seems" 2>&1 > /dev/null
    else
        echo "# [$0 `/bin/date`] could not identify running process ${PID}, starting new process" 2>&1 >> ${LOGFILE}
        start
    fi
}


case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    restart
    ;;
  checkup)
    checkup
    ;;
  *)
    echo $"Usage: $0 {start|stop|restart|checkup}"
    exit 1
esac
```

Example

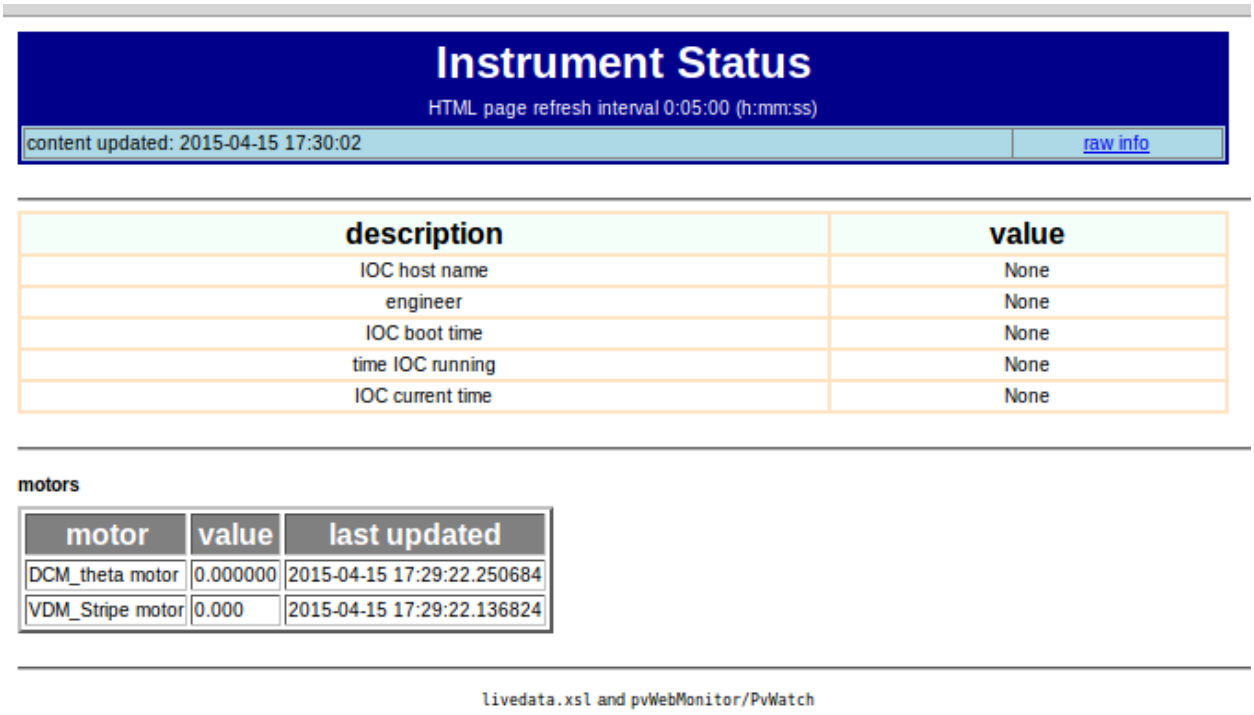
Livedata Report

file <web_site>/index.html

XSLT `livedata.xsl`

raw data `rawdata.xml` (*Example Raw Data for Reports*)

(To view an archive copy of this HTML page in your browser, click here –> `livedata.html`)



Instrument Status

HTML page refresh interval 0:05:00 (h:mm:ss)

content updated: 2015-04-15 17:30:02 | raw info

description	value
IOC host name	None
engineer	None
IOC boot time	None
time IOC running	None
IOC current time	None

motors

motor	value	last updated
DCM_theta motor	0.000000	2015-04-15 17:29:22.250684
VDM_Stripe motor	0.000	2015-04-15 17:29:22.136824

livedata.xsl and pvWebMonitor/PvWatch

Fig. 1.5: Example user report of the raw data monitored by an instance of *pvWebMonitor*.

Example Raw Data for Reports

file <web_site>/rawdata.html

XSLT `rawdata.xsl`

raw data `rawdata.xml` (*Example Raw Data for Reports*)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="pvlist.xsl"?>
<pvWebMonitor version="1">
  <written_by>pvWebMonitor/PvWatch</written_by>
  <datetime>2015-01-15 16:48:36</datetime>
  <pv id="DCM_theta" name="prj:m2.RBV">
    <name>prj:m2.RBV</name>
    <id>DCM_theta</id>
    <description>DCM_theta motor</description>
    <timestamp>2015-01-15 15:30:16.891366</timestamp>
    <counter>2</counter>
    <units>deg</units>
    <value>-9.840000</value>
    <raw_value>-9.84</raw_value>
    <format>%.6f</format>
  </pv>
  <pv id="DCM_theta_dmov" name="prj:m2.DMOV">
    <name>prj:m2.DMOV</name>
    <id>DCM_theta_dmov</id>
    <description>DCM_theta motor done moving</description>
    <timestamp>2015-01-15 15:30:16.864203</timestamp>
    <counter>2</counter>
    <units>deg</units>
```

```
    <value>1</value>
    <raw_value>1</raw_value>
    <format>%s</format>
  </pv>
  <pv id="VDM_Stripe" name="prj:m1.RBV">
    <name>prj:m1.RBV</name>
    <id>VDM_Stripe</id>
    <description>VDM_Stripe motor</description>
    <timestamp>2015-01-15 15:30:16.837633</timestamp>
    <counter>2</counter>
    <units>deg</units>
    <value>-1.510</value>
    <raw_value>-1.51</raw_value>
    <format>%.3f</format>
  </pv>
  <pv id="VDM_Stripe_dmov" name="prj:m1.DMOV">
    <name>prj:m1.DMOV</name>
    <id>VDM_Stripe_dmov</id>
    <description>VDM_Stripe motor done moving</description>
    <timestamp>2015-01-15 15:30:16.810400</timestamp>
    <counter>2</counter>
    <units>deg</units>
    <value>1</value>
    <raw_value>1</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai0" name="ino:cr:ai0">
    <name>ino:cr:ai0</name>
    <id>ai0</id>
    <description>Arduino AI0</description>
    <timestamp>2015-01-15 16:48:35.481271</timestamp>
    <counter>1808</counter>
    <units>V</units>
    <value>0.0439882697947</value>
    <raw_value>0.0439882697947</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai0_mean" name="ino:cr:ai0:mean">
    <name>ino:cr:ai0:mean</name>
    <id>ai0_mean</id>
    <description>Arduino mean AI0</description>
    <timestamp>2015-01-15 16:48:35.987300</timestamp>
    <counter>7901</counter>
    <units>V</units>
    <value>0.0467644183773</value>
    <raw_value>0.0467644183773</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai1" name="ino:cr:ai1">
    <name>ino:cr:ai1</name>
    <id>ai1</id>
    <description>Arduino AI1</description>
    <timestamp>2015-01-15 16:43:42.932475</timestamp>
    <counter>6</counter>
    <units>V</units>
    <value>0.0</value>
    <raw_value>0.0</raw_value>
    <format>%s</format>
```

```
  </pv>
  <pv id="ai1_mean" name="ino:cr:ai1:mean">
    <name>ino:cr:ai1:mean</name>
    <id>ai1_mean</id>
    <description>Arduino mean AI1</description>
    <timestamp>2015-01-15 15:30:17.053715</timestamp>
    <counter>2</counter>
    <units>V</units>
    <value>0.0</value>
    <raw_value>0.0</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai2" name="ino:cr:ai2">
    <name>ino:cr:ai2</name>
    <id>ai2</id>
    <description>Arduino AI2</description>
    <timestamp>2015-01-15 16:47:54.988769</timestamp>
    <counter>1319</counter>
    <units>V</units>
    <value>2.64418377322</value>
    <raw_value>2.64418377322</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai2_mean" name="ino:cr:ai2:mean">
    <name>ino:cr:ai2:mean</name>
    <id>ai2_mean</id>
    <description>Arduino mean AI2</description>
    <timestamp>2015-01-15 16:48:35.998230</timestamp>
    <counter>7020</counter>
    <units>V</units>
    <value>2.64418377322</value>
    <raw_value>2.64418377322</raw_value>
    <format>%s</format>
  </pv>
  <pv id="ai3_mean" name="ino:cr:ai3:mean">
    <name>ino:cr:ai3:mean</name>
    <id>ai3_mean</id>
    <description>Arduino mean AI3</description>
    <timestamp>2015-01-15 16:48:33.503247</timestamp>
    <counter>5977</counter>
    <units>V</units>
    <value>2.61974584555</value>
    <raw_value>2.61974584555</raw_value>
    <format>%s</format>
  </pv>
  <pv id="arduino_rate" name="ino:cr:rate">
    <name>ino:cr:rate</name>
    <id>arduino_rate</id>
    <description>Arduino update rate</description>
    <timestamp>2015-01-15 16:48:36.004167</timestamp>
    <counter>6214</counter>
    <units>1/s</units>
    <value>2142.0</value>
    <raw_value>2142.0</raw_value>
    <format>%s</format>
  </pv>
  <pv id="engineer" name="prj:ENGINEER">
    <name>prj:ENGINEER</name>
```

```
    <id>engineer</id>
    <description>engineer</description>
    <timestamp>2015-01-15 15:30:16.782947</timestamp>
    <counter>2</counter>
    <units></units>
    <value>engineer</value>
    <raw_value>engineer</raw_value>
    <format>%s</format>
  </pv>
  <pv id="hostname" name="prj:HOSTNAME">
    <name>prj:HOSTNAME</name>
    <id>hostname</id>
    <description>IOC host name</description>
    <timestamp>2015-01-15 15:30:16.755911</timestamp>
    <counter>2</counter>
    <units></units>
    <value>gov.aps.anl.gov</value>
    <raw_value>gov.aps.anl.gov</raw_value>
    <format>%s</format>
  </pv>
  <pv id="motors_alldone" name="prj:alldone">
    <name>prj:alldone</name>
    <id>motors_alldone</id>
    <description>all motors done moving</description>
    <timestamp>2015-01-15 15:30:16.919038</timestamp>
    <counter>2</counter>
    <units></units>
    <value>1</value>
    <raw_value>1</raw_value>
    <format>%s</format>
  </pv>
  <pv id="motors_moving" name="prj:moving">
    <name>prj:moving</name>
    <id>motors_moving</id>
    <description>number of motors moving</description>
    <timestamp>2015-01-15 15:30:16.945825</timestamp>
    <counter>2</counter>
    <units></units>
    <value>0</value>
    <raw_value>0</raw_value>
    <format>%s</format>
  </pv>
  <pv id="starttod" name="prj:STARTTOD">
    <name>prj:STARTTOD</name>
    <id>starttod</id>
    <description>IOC boot time</description>
    <timestamp>2015-01-15 15:30:16.701840</timestamp>
    <counter>2</counter>
    <units></units>
    <value>01/12/2015 12:56:08</value>
    <raw_value>01/12/2015 12:56:08</raw_value>
    <format>%s</format>
  </pv>
  <pv id="tod" name="prj:TOD">
    <name>prj:TOD</name>
    <id>tod</id>
    <description>IOC current time</description>
    <timestamp>2015-01-15 16:48:35.521618</timestamp>
```

```
    <counter>4701</counter>
    <units></units>
    <value>01/15/2015 16:48:35</value>
    <raw_value>01/15/2015 16:48:35</raw_value>
    <format>%s</format>
  </pv>
  <pv id="uptime" name="prj:UPTIME">
    <name>prj:UPTIME</name>
    <id>uptime</id>
    <description>time IOC running</description>
    <timestamp>2015-01-15 16:48:35.521851</timestamp>
    <counter>4701</counter>
    <units></units>
    <value>3 days, 03:52:27</value>
    <raw_value>3 days, 03:52:27</raw_value>
    <format>%s</format>
  </pv>
</pvWebMonitor>
```

(To view an archive copy of this HTML page in your browser, click here –> `rawdata.html`)

Source Code

pvWebMonitor Package: `main` Module

pvWebMonitor.main

USAGE:

```
jemian@gov:~$ pvWebMonitor
usage: pvWebMonitor [-h] [-l LOG_FILE] [-v] xml_config_file
pvWebMonitor: error: too few arguments
```

HELP:

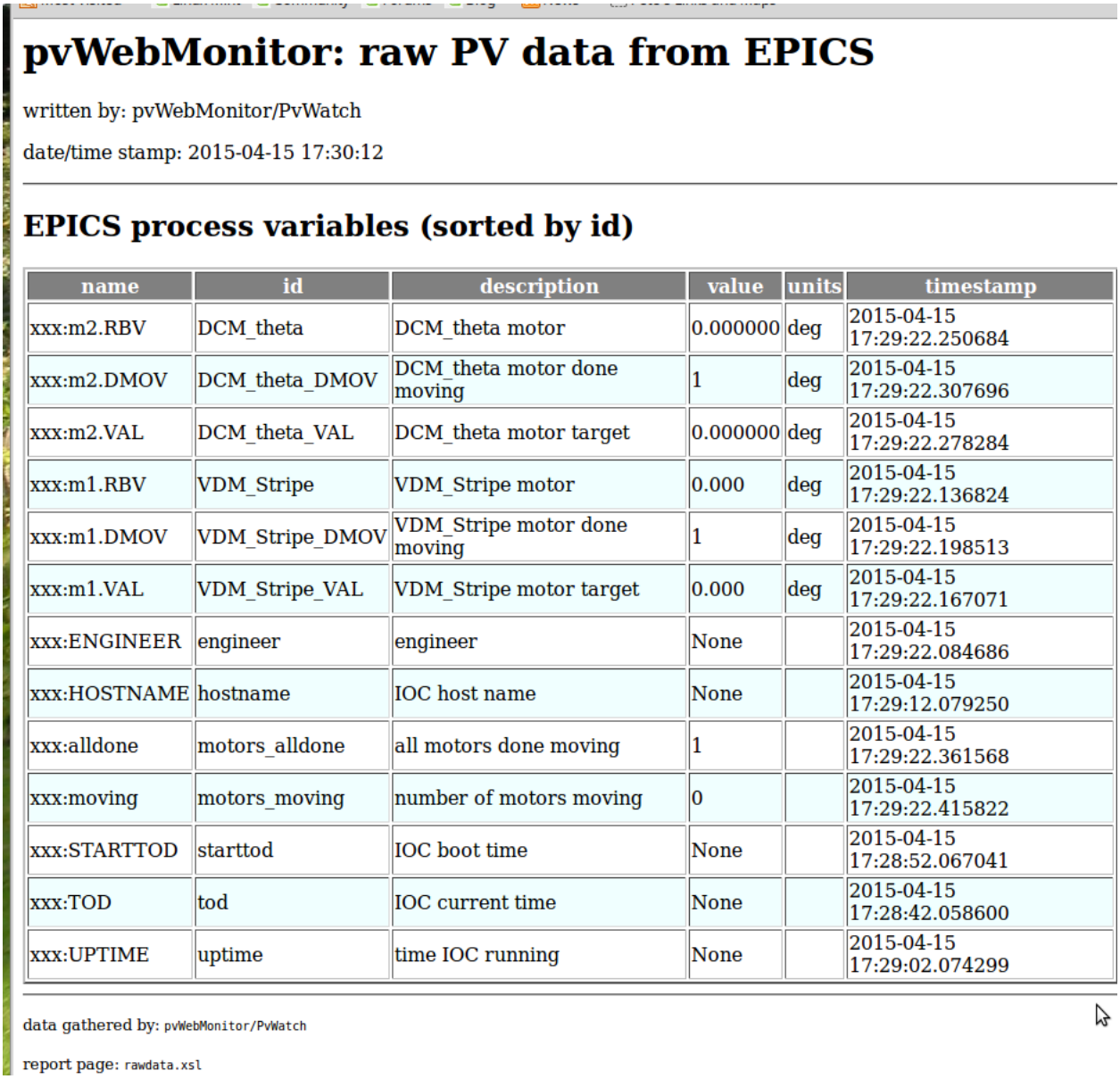
```
jemian@gov:~$ pvWebMonitor -h
usage: pvWebMonitor [-h] [-l LOG_FILE] [-v] [--setup SETUP] xml_config_file

pvWebMonitor: post EPICS PVs to read-only web page

positional arguments:
  xml_config_file       XML configuration file

optional arguments:
  -h, --help            show this help message and exit
  -l LOG_FILE, --log_file LOG_FILE
                        log file
  -v, --version         show program's version number and exit

getting started (none of the above):
  --setup SETUP         setup a new project directory
```



pvWebMonitor: raw PV data from EPICS

written by: pvWebMonitor/PvWatch

date/time stamp: 2015-04-15 17:30:12

EPICS process variables (sorted by id)

name	id	description	value	units	timestamp
xxx:m2.RBV	DCM_theta	DCM_theta motor	0.000000	deg	2015-04-15 17:29:22.250684
xxx:m2.DMOV	DCM_theta_DMOV	DCM_theta motor done moving	1	deg	2015-04-15 17:29:22.307696
xxx:m2.VAL	DCM_theta_VAL	DCM_theta motor target	0.000000	deg	2015-04-15 17:29:22.278284
xxx:m1.RBV	VDM_Stripe	VDM_Stripe motor	0.000	deg	2015-04-15 17:29:22.136824
xxx:m1.DMOV	VDM_Stripe_DMOV	VDM_Stripe motor done moving	1	deg	2015-04-15 17:29:22.198513
xxx:m1.VAL	VDM_Stripe_VAL	VDM_Stripe motor target	0.000	deg	2015-04-15 17:29:22.167071
xxx:ENGINEER	engineer	engineer	None		2015-04-15 17:29:22.084686
xxx:HOSTNAME	hostname	IOC host name	None		2015-04-15 17:29:12.079250
xxx:alldone	motors_alldone	all motors done moving	1		2015-04-15 17:29:22.361568
xxx:moving	motors_moving	number of motors moving	0		2015-04-15 17:29:22.415822
xxx:STARTTOD	starttod	IOC boot time	None		2015-04-15 17:28:52.067041
xxx:TOD	tod	IOC current time	None		2015-04-15 17:28:42.058600
xxx:UPTIME	uptime	time IOC running	None		2015-04-15 17:29:02.074299

data gathered by: pvWebMonitor/PvWatch

report page: rawdata.xsl

Fig. 1.6: Example report of the raw data monitored by an instance of *pvWebMonitor*.

VERSION:

```
jemian@gov:~$ pvWebMonitor -v
2015.0112.0
```

pvWebMonitor.main.**main**()
entry point for the command-line interface

pvWebMonitor Package: pvwatch Module

pvWebMonitor.pvwatch

exception pvWebMonitor.pvwatch.**CouldNotParseXml**
Bases: exceptions.Exception

Could not parse XML file

exception pvWebMonitor.pvwatch.**PvNotRegistered**
Bases: exceptions.Exception

pv not in pvdb

class pvWebMonitor.pvwatch.**PvWatch**(*configuration*)
Bases: object

Core function of the pvWebMonitor package

To call this code, first define configuration=dict() with terms as defined in read_config.read_xml(), then statements such as:

```
watcher = PvWatch(configuration)
watcher.start()
```

EPICS_monitor_receiver(**args*, ***kws*)
Response to an EPICS (PyEpics) monitor on the channel

add_file_pattern(*pattern*)
add pattern as an additional file extension pattern

Any file with extension matching any of the patterns in self.upload_patterns will copied to the WWW directory, if they are newer.

add_pv(*mne*, *pv*, *desc*, *fmt*, *as_string*)
Connect to a EPICS (PyEpics) process variable

buildReport()
build the report

get_pvlist()
get the PVs from the XML file

report()
write the values out to files

The values of the monitored EPICS PVs (the "raw data") is written to an XML file. This file is then used with one or more XSLT stylesheets to create HTML pages. An overall "home page" (index.html) is created to provide a table of contents of this static web site.

start()
begin receiving PV updates and posting new web content

update_pvdb (*pv*, *raw_value*)
log PV value to the cache in pvdb

Parameters

- **pv** (*str*) – name of EPICS PV
- **raw_value** (*obj*) – could be str, float, int, or ...

pvWebMonitor Package: read_config Module

read XML configuration file for *pvWebMonitor* package

pvWebMonitor.read_config.**patterns_1_0** (**args*)
config_1_0 relies on a pre-defined list of file match patterns

pvWebMonitor.read_config.**patterns_1_0_1** (**args*)
config_1_0_1 uses a user-defined list of file match patterns

pvWebMonitor.read_config.**read_xml** (*xml_file*)
return the configuration details as a dictionary

Parameters **return** – dictionary

the dictionary WILL contain these definitions for use by pvwatch.PvWatch():

dictionary key	example (type)	description
PVLIST_FILE	pvlist.xml	PVs to be monitored
LOCAL_WWW_LIVEDATA_DIR	./localwww	absolute path to local directory with "web site"
LOG_INTERVAL_S	300 (float)	writing messages to log file
REPORT_INTERVAL_S	10 (float)	updates to HTML pages
SLEEP_INTERVAL_S	0.1 (float)	sleeps at end of main loop
MAINLOOP_COUNTER_TRIGGER	10000 (int)	another logging message interval
PATTERNS	*.html	upload all files that match these patterns

pvWebMonitor Package: setup Module

setup a new project directory

pvWebMonitor.setup.**get_key_value** (*key*, *txt*)
find the assignment line in txt: key=value, and return value

pvWebMonitor.setup.**main** (*new_directory*)
setup a new project directory in *new_directory*

new_directory must exist and not contain any of the files to be copied into it.

Parameters **new_directory** (*str*) – name of existing directory

pvWebMonitor.setup.**modify_manage_script** (*filename*)
customize the manage.sh script for the current setup

pvWebMonitor Package: utils Module

pvWebMonitor.utils

pvWebMonitor.utils.**copyToWebServer**(*local_file*, *web_site_path*)
copy local file to web server directory (on a local file system)

This copy routine assumes that it is not necessary to use scp to copy the file.

pvWebMonitor.utils.**getTime**()
simple wrapper for common timenow() function

pvWebMonitor.utils.**logException**(*troublemaker*)
write an exception report to the log file

Parameters **troublemaker** (*obj*) – instance of Exception

pvWebMonitor.utils.**logMessage**(*message*)
log a message or report from pvWebMonitor

Parameters **message** (*str*) – words to be logged

pvWebMonitor.utils.**validate**(*xml_tree*, *xml_schema_file*)
validate an XML document tree against an XML Schema file

Parameters

- **xml_tree** (*obj*) – instance of etree._ElementTree
- **xml_schema_file** (*str*) – name of XML Schema file (local to package directory)

pvWebMonitor.utils.**writeFile**(*output_file*, *contents*)
write contents to file

Parameters

- **output_file** (*str*) – file to be written (path is optional)
- **contents** (*str*) – text to write in *output_file*

pvWebMonitor.utils.**xslt_transformation**(*xslt_file*, *src_xml_file*, *result_xml_file*)
transform an XML file using an XSLT

Parameters

- **xslt_file** (*str*) – name of XSLT file
- **src_xml_file** (*str*) – name of XML file
- **result_xml_file** (*str*) – name of output XML file

CHANGES

2016.1025.0 revise the versioning process

2016.1003.2 issue #22: correct version number shown now

2016.0907.0 issue #18: check XSLT files for syntax errors, issue #19: let user choose to write waveform strings as string or array of integers, issue #20: add username and host to logging messages

2016.0516.2 #16: accept both version 1.0 & 1.0.1 config.xml files

2016.0427.1 #12: user can add additional file extension patterns, improve the setup of manage.sh on Linux

2016.0414.2 #9: resolve ValueError when creating XML declaration

2015.0117.0 #6: rename project to pvWebMonitor

2015.0116.0 #13: management shell script now uses /bin/bash

2015.0115.0 #4: refactor XSLT infrastructure and web site

2015.0114.1 include XML infrastructure in package

2015.0114.0 packaging update

2015.0113.1 add –setup to fill a new project directory with needed files

2015.0113.0 validate all XML files and raise exceptions if invalid

2015.0112.2 documentation at ReadTheDocs, package at PyPI, code at GitHub

2015-01-09 v1.0.0 initial conversion from USAXS livedata project

Software License

```
Copyright (c) 2009-2016, UChicago Argonne, LLC

All Rights Reserved

pvWebMonitor

Advanced Photon Source, Argonne National Laboratory


OPEN SOURCE LICENSE

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.  Software changes,
   modifications, or derivative works, should be noted with comments and
   the author and organization's name.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. Neither the names of UChicago Argonne, LLC or the Department of Energy
   nor the names of its contributors may be used to endorse or promote
   products derived from this software without specific prior written
   permission.

4. The software and the end-user documentation included with the
   redistribution, if any, must include the following acknowledgment:

   "This product includes software produced by UChicago Argonne, LLC
   under Contract No. DE-AC02-06CH11357 with the Department of Energy."

**********************************************************************

DISCLAIMER

THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF ANY KIND.

Neither the United States GOVERNMENT, nor the United States Department
```

```
of Energy, NOR UChicago Argonne, LLC, nor any of their employees, makes
any warranty, express or implied, or assumes any legal liability or
responsibility for the accuracy, completeness, or usefulness of any
information, data, apparatus, product, or process disclosed, or
represents that its use would not infringe privately owned rights.

*************************************************************************
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

This documentation was built Mar 16, 2017

Python Module Index

p

pvWebMonitor.main, 20
pvWebMonitor.pvwatch, 22
pvWebMonitor.read_config, 23
pvWebMonitor.setup, 23
pvWebMonitor.utils, 23

Index

A

add_file_pattern() (pvWebMonitor.pvwatch.PvWatch method), 22
add_pv() (pvWebMonitor.pvwatch.PvWatch method), 22

B

buildReport() (pvWebMonitor.pvwatch.PvWatch method), 22

C

config.xml, 6
copyToWebServer() (in module pvWebMonitor.utils), 23
CouldNotParseXml, 22

E

EPICS_monitor_receiver() (pvWebMonitor.pvwatch.PvWatch method), 22

G

get_key_value() (in module pvWebMonitor.setup), 23
get_pvlist() (pvWebMonitor.pvwatch.PvWatch method), 22
getTime() (in module pvWebMonitor.utils), 24

L

livedata.xsl, 6
log_data.txt, 6
logException() (in module pvWebMonitor.utils), 24
logMessage() (in module pvWebMonitor.utils), 24

M

main() (in module pvWebMonitor.main), 22
main() (in module pvWebMonitor.setup), 23
modify_manage_script() (in module pvWebMonitor.setup), 23

P

patterns_1_0() (in module pvWebMonitor.read_config), 23
patterns_1_0_1() (in module pvWebMonitor.read_config), 23
project configuration directory, 6
project directory, 3
pvlist.xml, 6
PvNotRegistered, 22
PvWatch (class in pvWebMonitor.pvwatch), 22
pvWebMonitor.main (module), 20
pvWebMonitor.pvwatch (module), 22
pvWebMonitor.read_config (module), 23
pvWebMonitor.setup (module), 23
pvWebMonitor.utils (module), 23

R

read_xml() (in module pvWebMonitor.read_config), 23
report() (pvWebMonitor.pvwatch.PvWatch method), 22

S

start() (pvWebMonitor.pvwatch.PvWatch method), 22

U

update_pvdb() (pvWebMonitor.pvwatch.PvWatch method), 22

V

validate() (in module pvWebMonitor.utils), 24

W

web server directory, 6
writeFile() (in module pvWebMonitor.utils), 24

X

xslt_transformation() (in module pvWebMonitor.utils), 24