PvMail Documentation

Release 0.gc9f879d.dirty

Pete Jemian, APS, ANL <jemian@anl.gov>

May 21, 2019

Contents

Watches an EPICS PV and sends email when value changes from 0 to 1.

author Pete R. Jemian

email jemian@anl.gov

docs http://PvMail.readthedocs.io

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

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

version 3.2.8

release 0.gc9f879d.dirty

published May 21, 2019

Note: While *PvMail* is the name of the Python package, the executable installed in <python>/bin is called `pvMail` using a command line such as:

```
[user@host,518,~]$ pvMail
```

PvMail watches (monitor) an EPICS PV and send an email when the value of that PV changes from 0 to 1.

IMPORTANT:

PvMail *only* triggers when the trigger_PV makes a transition from 0 to 1.

- It will ignore transitions (in floating-point PVs) to 1.0000... that do not come directly from 0.0000... If you wish to watch a PV that presents values other than 0 and 1, then use a calculation PV as the trigger which results in a transition from 0 to 1.
- Reset the value of the trigger_PV to 0 to resume watching for the next triggered event.

The PV being watched (that *triggers* the sending of the email) can be any EPICS record type or field that results in a value of 0 (zero) that changes to 1 (one). This includes these EPICS records (and possibly more): *ai*, *ao*, *bi*, *bo*, *calcout*, *scalcout*, *swait*, ...

When an event causes an email to be triggered, PvMail will retrieve the value of another PV that is the first part of the message to be sent. Additional metadata will be appended to the message.

Note: Email is sent using either a call to a configured SMTP server or the `sendmail` program on the native OS. The sendmail protocol is only supported on Linux systems that provide a `sendmail` program. The SMTP protocol is more general but requires valid credentials on the SMTP server and the credentials must be stored in a local configuration file.

PvMail provides either a command-line interface or a graphical user interface. Both are accessed from the same command, using different command-line options. The command-line version is intended to run as a background program, it has no user interaction but logs all its output into a log file. The GUI version provides a screen to edit each of the parameters before the background process is started. It also provides buttons to start and stop the background process.

1 Overview of Contents

1.1 Contents

Overview

There are now several parts to the PvMail support package.

command	section	description
`pvMail -g`	*pvMail: graphical user interface*	runs the graphical user interface
`pvMail`	*pvMail: command-line interface*	runs the command line interface
`pvMail_mail_config_file`	*ini_config Module*	prints the name of the configuration file
`pvMail_mail_test`	*mailer Module*	tests the emailer and configuration file

One-time steps

Before you can run **pvMail**, you need to configure it.

First, run:

```
pvMail_mail_config_file
```

if you have not already created a configuration file. This command will create the file (if it does not exist) and then print its name to the console.

Edit this file for the particulars of how you want to send your email. Refer to the *ini_config Module* section for additional details about the configuration file.

Next, run:

```
pvMail_mail_test joeuser@example.com
```

(use your own email address, not Joe's) to test that an email can be sent using your configuration. Refer to the *mailer Module* section for additional details about sending email. Run this tool any time you suspect that you cannot send email.

Routine usage

You can run **pvMail** either in command-line mode (foreground or background) or in GUI mode. Follow the links above for more details about each.

pvMail: command-line interface

Basically, you use it either as a background daemon or as a GUI. Call it with a `-g` or `--gui` command line option to force the GUI to run, otherwise you get the background daemon. Either way, it makes a log file (based on PID number) with any program output.

background daemon:

```
pvMail triggerPV messagePV user1@email.domain,user2@host.server &
```

GUI:

```
pvMail triggerPV messagePV user1@email.domain,user2@host.server -g &
```

PvMail uses Matt Newville's PyEpics package for EPICS CA connections and PyQt4 package to manage the GUI.

Tip: Since *PvMail* creates a log file (by default in the current working directory), be sure you start the program from a directory to which you have write access or specify the absolute path to the log file as a command line argument:

```
pvMail -l /path/to/log_file.txt triggerPV messagePV user1@email.domain &
```

Starting PvMail from the command-line

PvMail is started from the command line:

```
$ pvMail pvMail:trigger pvMail:message jemian
```

No program output is printed to the screen. Instead, the output is directed to a log file. Here is an example:

```
INFO:root:(pvMail.py,2011-11-27 19:03:23.072392) ####################################
↪#######################
INFO:root:(pvMail.py,2011-11-27 19:03:23.072826) startup
INFO:root:(pvMail.py,2011-11-27 19:03:23.072897) trigger PV       = pvMail:trigger
```

(continues on next page)

(continued from previous page)

```
INFO:root:(pvMail.py,2011-11-27 19:03:23.073323) message PV       = pvMail:message
INFO:root:(pvMail.py,2011-11-27 19:03:23.073401) email list       = ['jemian']
INFO:root:(pvMail.py,2011-11-27 19:03:23.073463) log file         = logfile.log
INFO:root:(pvMail.py,2011-11-27 19:03:23.073667) logging interval = 5.0
INFO:root:(pvMail.py,2011-11-27 19:03:23.073735) sleep duration   = 0.2
INFO:root:(pvMail.py,2011-11-27 19:03:23.073795) interface        = command-line
INFO:root:(pvMail.py,2011-11-27 19:03:23.073855) user             = jemian
INFO:root:(pvMail.py,2011-11-27 19:03:23.073952) host             = como-ubuntu64
INFO:root:(pvMail.py,2011-11-27 19:03:23.074053) program          = ./pvMail.py
INFO:root:(pvMail.py,2011-11-27 19:03:23.074124) PID              = 8903
INFO:root:(pvMail.py,2011-11-27 19:03:23.074196) do_start
INFO:root:(pvMail.py,2011-11-27 19:03:23.074280) test connect with pvMail:message
INFO:root:(pvMail.py,2011-11-27 19:03:23.445334) test connect with pvMail:trigger
INFO:root:(pvMail.py,2011-11-27 19:03:23.468540) passed basicChecks(), starting␣
↪monitors
INFO:root:(pvMail.py,2011-11-27 19:03:23.477917) checkpoint
INFO:root:(pvMail.py,2011-11-27 19:03:27.373142) pvMail:trigger = 1
INFO:root:(pvMail.py,2011-11-27 19:03:27.373908) SendMessage
INFO:root:(pvMail.py,2011-11-27 19:03:27.374199) sending email to: jemian
INFO:root:(pvMail.py,2011-11-27 19:03:27.374716) mail -s "pvMail.py: pvMail:trigger"␣
↪jemian < /tmp/pvmail_message.txt
INFO:root:(pvMail.py,2011-11-27 19:03:27.538022) message(s) sent
INFO:root:(pvMail.py,2011-11-27 19:03:28.092551) checkpoint
INFO:root:(pvMail.py,2011-11-27 19:03:29.440516) pvMail:trigger = 0
```

The program starts, reports its configurations, and connects with the EPICS PVs, and then goes into a background mode. A checkpoint (command-line option `-i`) is reported periodically. The default is 5 seconds. This may be changed to 10 minutes or longer for production use, but is always specified in seconds.

Observe that, in the above example, the trigger PV changed from 0 to 1 at 19:03:27.373142 (and back to 0 at 19:03:29.440516). The change at ~19:03:27 triggered PvMail to send an email as configured. For now, the code writes the text of the email to a temporary file (command-line option `-m`, default is "/tmp/pvmail_message.txt"). In this example, the message reads:

```
pvMail default message

user: jemian
host: como-ubuntu64
date: 2011-11-27 19:03:27.374135
program: ./pvMail.py
PID: 8903
trigger PV: pvMail:trigger
message PV: pvMail:message
recipients: jemian
```

The message shows up in the mail browser (here my Linux `mail` program):

```
jemian@como-ubuntu64$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/jemian": 3 messages 3 new
>N  1 jemian@como-ubunt  Sun Nov 27 18:27   25/730   pvMail.py: pvMail:trigger
 N  2 jemian@como-ubunt  Sun Nov 27 18:58   25/730   pvMail.py: pvMail:trigger
 N  3 jemian@como-ubunt  Sun Nov 27 19:03   25/730   pvMail.py: pvMail:trigger
```

The full message, as seen in the mail browser is:

```
Message 3:
From jemian@como-ubuntu64 Sun Nov 27 19:03:27 2011
Envelope-to: jemian@como-ubuntu64
Delivery-date: Sun, 27 Nov 2011 19:03:27 -0600
To: jemian@como-ubuntu64
Subject: pvMail.py: pvMail:trigger
From: Pete R Jemian <jemian@como-ubuntu64>
Date: Sun, 27 Nov 2011 19:03:27 -0600

pvMail default message

user: jemian
host: como-ubuntu64
date: 2011-11-27 19:03:27.374135
program: ./pvMail.py
PID: 8903
trigger PV: pvMail:trigger
message PV: pvMail:message
recipients: jemian
```

Starting PvMail from the command-line at the APS

At the APS, Enthought Python Distribution is installed on the /APSshare partition available to all beam lines.

Here is a command to run PvMail and get the help message:

```
/APSshare/epd/rh5-x86_64/bin/pvMail -h
```

or the 32-bit version:

```
/APSshare/epd/rh5-x86/bin/pvMail -h
```

Note: Support at APS for both RHEL5 and RHEL6 use the same Enthought Python Distribution.

---

command-line parameters

usage

When PvMail is started from the command line with no additional parameters:

```
$ pvMail

usage: pvMail [-h] [-l LOG_FILE] [-i LOGGING_INTERVAL]
                  [-r SLEEP_DURATION] [-g] [-v]
                  trigger_PV message_PV email_addresses
pvMail: error: too few arguments
```

This is the *usage* message. It tells us we must supply three positional arguments: `trigger_PV message_PV email_addresses`.

positional argument: `trigger_PV`

EPICS process variable name to watch using a CA monitor. When `trigger_PV` makes a transition from 0 (zero) to 1 (one), then get the string from the `message_PV` and send an email to all of the `email_addresses` on the list.

positional argument: `message_PV`

EPICS process variable name pointing to a (short) message that will be used as the first part of the email message to be sent.

positional argument: `email_addresses`

List of email addresses, separated by commas if more than one. For example, `user1@email.domain, user2@host.server` will send one email to `user1@email.domain` and another email to `user2@host.server`.

Note: At Argonne, it is possible to send email to a pager using the email address `####@pager.anl.gov` and the pager number. Be sure not to use a preceding `4-` or the email will not be deliverable.

---

option: `--version` or `-v`

The current version of the program can always be printed using the `-v` or `--version`. With this option, the program prints the version number and then quits.

```
$ pvMail --version
3.0-663
```

option: `--help` or `-h`

It may be easier to review the short help instructions for command-line options:

```
$ ./pvMail --help
usage: pvMail [-h] [-l LOG_FILE] [-i LOGGING_INTERVAL]
                 [-r SLEEP_DURATION] [-g] [-v]
                 trigger_PV message_PV email_addresses

Watch an EPICS PV. Send email when it changes from 0 to 1.

positional arguments:
  trigger_PV           EPICS trigger PV name
  message_PV           EPICS message PV name
  email_addresses      email address(es), comma-separated if more than one

optional arguments:
  -h, --help           show this help message and exit
  -l LOG_FILE          for logging program progress and comments
  -i LOGGING_INTERVAL  checkpoint reporting interval (s) in log file
  -r SLEEP_DURATION    sleep duration (s) in main event loop
  -g, --gui            Use the graphical rather than command-line interface
  -v, --version        show program's version number and exit
```

option: `--gui` or `-g`

This command line option is used to start the GUI (see *pvMail: graphical user interface*). If either GUI option is used, then the positional arguments (`triggerPV messagePV email@address`) are optional.

option: `-l LOG_FILE`

Both the command-line and GUI versions of PvMail log all program output to a log file. If a LOG_FILE is not specified on the command line, the default file will be `pvMail-PID.log` in the current directory where *PID* is the process identifier of the running `pvMail` program.

Note: If the LOG_FILE already exists, new information will be appended. It is up to the account owner to delete a LOG_FILE when it is no longer useful.

The PID number is useful when you wish to end a program that is running as a background daemon. The UNIX/Linux command is:

```
kill PID
```

option: `-i LOGGING_INTERVAL`

units seconds

When a program runs in the background, waiting for occasional activity, there is often some concern that the program is actually prepared to act when needed. To offset this concern, PvMail will report a *checkpoint* message periodically (every LOGGING_INTERVAL seconds, default is every 5 minutes) to the LOG_FILE. The program ensures that LOGGING_INTERVAL is no shorter than 5 seconds or longer than 1 hour.

option: `-r SLEEP_DURATION`

units seconds

For operation as a background daemon process, the command-line version must check periodically for new EPICS CA events, using a call to `epics.ca.poll()`. In between calls, the application is told to sleep for SLEEP_DURATION seconds. The default SLEEP_DURATION is 0.2 seconds and is limited to values between 0.1 ms and 5 s.

pvMail: graphical user interface

The *PvMail* program GUI is started from the command line with the `-g` or `--gui` command-line options. If either GUI option is used, then the positional arguments (`triggerPV messagePV email@address`) are optional. Without either GUI option, the command-line interface is started.:

```
$ pvMail -g &
```

Tip: Usually, you want to run the GUI as a background task by appending the ampersand (`&`) on the command line, as shown above.

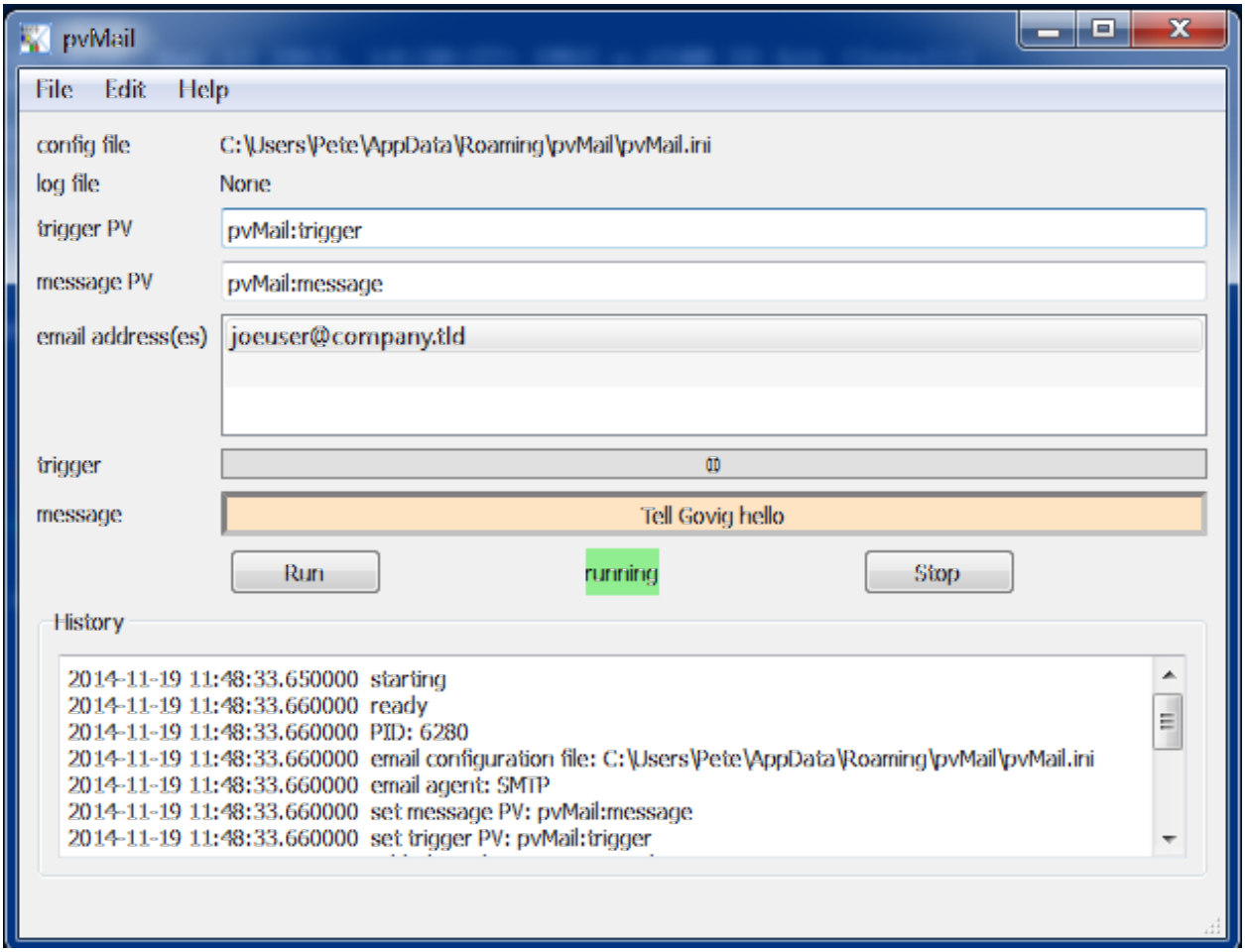

Fig. 1: GUI of the *PvMail* application

The GUI provides editable text entry widgets for each of the required command-line terms (a.k.a. *positional arguments*): `trigger_PV message_PV email_addresses`. The list of email addresses is separated. The GUI provides a tool to add additional address or remove addresses.

The GUI also shows (using *running* or *stopped* text) whether or not the PV monitor process is running.

> **Warning:** If either of the PVs fail to connect, it is likely that the GUI may become confused whether or not it is running. In such cases, press the *Stop* button, then press the *Run* button to try to restart monitoring.

All *PvMail* monitoring will be stopped if the GUI window is closed. At present, there is no feature to detach or reattach a monitor set. Also, *PvMail* can only monitor a single set of PVs using the current design. A request to enhance this capability is on the TODO list (see *TODO items for future releases*).

At the bottom of the GUI panel, a status log is shown. These are the contents of the log file.

Tip: Since *PvMail* creates a log file (by default in the current working directory), be sure you start the program from a directory to which you have write access or specify the absolute path to the log file as a command line argument:

```
pvMail -g -l /path/to/log_file.txt &
```

Send test email

Under the *File* menu, there is an item to `send test email` which attenpts to send a test email using the current settings as described in the configuration file.

EPICS test database

To test the program during its development, a test database (*test.db*) was prepared. The database creates two PVs:

`pvMail:trigger` the PV to watch

`pvMail:message` the message to be sent

starting: softIoc

Start the database by adding it to an existing EPIC IOC configuration or by starting a soft IOC using the `softIoc` program softIOC from EPICS base. Here is an example of how that looks from a Linux command shell:

```
$ softIoc -d test.db
Starting iocInit
############################################################################
## EPICS R3.14.12 $Date: Wed 2010-11-24 14:50:38 -0600$
## EPICS Base built Feb 27 2011
############################################################################
iocRun: All initialization complete
epics>
```

Note: Here, the shell prompt is signified by the `$` symbol.

---

watching: camonitor

Once the EPICS IOC is started and the PVs are available, it is possible to watch them for any changes from the command line using the `camonitor` camonitor application from EPICS base:

```
$ camonitor pvMail:trigger pvMail:message
    pvMail:trigger                 <undefined> off UDF INVALID
    pvMail:message                 <undefined> pvMail default message UDF INVALID
```

Note: Do not be concerned about the `UDF INVALID` notices, they will disappear once the PVs have been written to at least once.

---

changing a PV: caput

You can test changing the value of the trigger PV using the `caput` caput application from EPICS base:

```
$ caput pvMail:trigger 1
    Old : pvMail:trigger                off
    New : pvMail:trigger                on
```

changing a PV: dbpf (in the IOC shell)

You can change the value of the trigger PV using the `dbpf` dbpf command in the IOC shell:

```
$ dbpf("pvMail:trigger", 1)
"on"
$ dbpf("pvMail:trigger", 0)
"off"
```

test.db

Here is the full listing of the test EPICS database used for program development.

```
# EPICS database to use while testing and developing pvMail.py code

#  /APSshare/epics/base-3.14.12.1/bin/linux-x86-el5-debug/softIoc -d test.db
#
#  IOC:     softIoc -d test.db
#  client:  camonitor pvMail:{trigger,message}
#  pvMail:  pvMail.py  pvMail:trigger pvMail:message prjemian@gmail.com,jemian@anl.gov

record(bo, "pvMail:trigger")
{
        field(DESC, "trigger PV")
        field(ZNAM, "off")
        field(ONAM, "on")
}
record(stringout, "pvMail:message")
{
        field(DESC, "message to be sent by email")
```

(continues on next page)

(continued from previous page)

```
        field(VAL,  "pvMail default message")
}


# Copyright (c) 2014, UChicago Argonne, LLC.  See LICENSE file.
```

PvMail as a Python package

This section provides the source code documentation. The documentation here may be of little or no use to the casual user of this software.

installation

The PvMail project can be installed as a Python package.

1. Checkout the project from subversion
2. Change into the project working directory
3. Run `setup.py install`

starter program

Once the PvMail project has been installed as a package, the PvMail application can be run from the command line (assuming that the python executable is on the execution path):

```
pvMail
```

PvMail source code documentation

`cli` Module

Source code documentation for EPICS `cli`

Watch an EPICS PV. Send email when it changes from 0 to 1

class PvMail.cli.**PvMail**(*config=None*)
Bases: `threading.Thread`

Watch an EPICS PV (using PyEpics interface) and send an email when the PV changes from 0 to 1.

basicChecks()
check for valid inputs, raise exceptions as discovered, otherwise no return result

do_restart()
restart watching for triggers

do_start()
start watching for triggers

do_stop()
stop watching for triggers

receiveMessageMonitor (*value*, ***kw*)
respond to EPICS CA monitors on message PV

receiveTriggerMonitor (*value*, ***kw*)
respond to EPICS CA monitors on trigger PV

testConnect (*pvname*, *timeout=5.0*)
create PV, wait for connection, return connection state (True | False)

adapted from PyEpics __createPV() method

PvMail.cli.**SendMessage** (*pvm*, *agent_db*, *reporter=None*)
construct and send the message

Parameters **pvm** (*obj*) – instance of PvMail object on which to report

PvMail.cli.**cli** (*results*, *config=None*)
command-line interface to the PvMail class

Parameters

- **results** (*obj*) – default parameters from argparse, see main()
- **config** (*obj*) – email configuration from ini_config.Config()

PvMail.cli.**getUserName** (*db*)

PvMail.cli.**gui** (*results*, *config=None*)
graphical user interface to the PvMail class

Parameters

- **results** (*obj*) – default parameters from argparse, see main()
- **config** (*obj*) – email configuration from ini_config.Config()

PvMail.cli.**logger** (*message*)
log a message or report from PvMail

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

PvMail.cli.**main** ()
parse command-line arguments and choose which interface to use

uic_gui Module

Source code documentation for EPICS uic_gui

pvMail: just the GUI

Run the Graphical User Interface for PvMail using PyQt4 from a .ui file with the uic subpackage.

class PvMail.uic_gui.**EmailListModel** (*input_list*, *parent=None*, **args*)
Bases: PyQt4.QtCore.QAbstractListModel

data (*self*, *QModelIndex*, *role: int = Qt.DisplayRole*) → QVariant

flags (*self*, *QModelIndex*) → Qt.ItemFlags

rowCount (*self*, *parent: QModelIndex = QModelIndex()*) → int

setData(*self*, *QModelIndex*, *QVariant*, *role: int = Qt.EditRole*) → bool

class PvMail.uic_gui.**PvMailSignalDef**

Bases: PyQt4.QtCore.QObject

Define the signals used to communicate between the threads.

EPICS_monitor

class PvMail.uic_gui.**PvMail_GUI**(*ui_file=None*, *logger=None*, *logfile=None*, *config=None*, **args*, ***kw*)

Bases: object

GUI used for pvMail, based on PyQt4

appendEmailList(*email_addr*)

doAbout(**args*, ***kw*)

doClose(**args*, ***kw*)

doRun(**args*, ***kw*)

doSendTestMessage()

doStop(**args*, ***kw*)

doUrl()

getEmailList()
the complete list of email addresses

getEmailList_Stripped()
the list of email addresses with empty items removed

getMessagePV()

getTriggerPV()

logfile_to_history()

onMessage_gui_thread(*value*)

onMessage_pv_thread(*value=None*, **args*, ***kw*)

onTrigger_gui_thread(*value*)

onTrigger_pv_thread(*value=None*, *char_value=None*, **args*, ***kw*)

setEmailList(*email_list*)

setMessagePV(*messagePV*)

setStatus(*message*)

setTriggerPV(*triggerPV*)

show()

PvMail.uic_gui.**main**(*triggerPV*, *messagePV*, *recipients*, *logger=None*, *logfile=None*, *config=None*)

ini_config Module

Application defaults are stored in a file compatible with ConfigParser.Config. This file is to be stored in a *secure* location such that it is accessible and readable only to the account user (to the extent possible on the operating system).

The application configuration settings file *pvMail.ini* is stored in a directory that depends on the operating system, as selected by the `os.name` value, as shown in this table:

os.name	path
posix	`*$HOST/.pvMail/pvMail.ini*`
nt	`*%APPDATA%\\\\pvMail\\\\pvMail.ini*`

The user can override this path by defining the **PVMAIL_INI_FILE** environment variable to point to the desired application configuration settings file. This definition must be made before the call to *PvMail.ini_config.Config*.

If the application configuration settings file does not exist, a default one will be created on the first call to *PvMail.ini_config.Config*. This default configuration file is only a template and **must be modified** with the user's settings before email can be sent successfully.

OBJECTIVE

The main reason why an application configuration settings file is needed is to supply the configuration to send email from `PvMail.pvMail` through an SMTP server.

example application configuration settings (`pvMail.ini`) file:

```
[header]
application = pvMail
written = 2014-11-10 16:16:10.751682

[mailer]
mail_transfer_agent = sendmail

[SMTP]
connection_security = STARTTLS
password = keep_this_private
user = joeuser
server = smtp.server.org
port = 465

[sendmail]
user = joeuser
```

OVERVIEW

The *PvMail.ini_config.Config* class reads the entire contents of the application configuration settings file and copies that to a dictionary in the class: *self.agent_db*. Each of the sections in the file (such as *SMTP*, *sendmail*) comprise subdictionaries with *key* = *value* content. The *header* section contains metadata about the file and is not read. The *mailer* section has a key *mail_transfer_agent* that indicates which mail transport agent[1] will be used to send the email. Current choices available are: *SMTP* or *sendmail* (supported on Linux only).

Comments in the application configuration settings file will be ignored and will not be written back to the file if the file is rewritten from `ini_config.Config.write()`. A tricky way to preserve *comment* information is to write the comment as if it were a variable to be set inside a section, or possible an entire section. Such as:

[1] MTA: https://en.wikipedia.org/wiki/Message_transfer_agent

```
[SMTP]
server = smtp.mycompany.com
user = j.o.e.user@mycompany.com
password = keep_this_private
hint = comment: use your email as "user" name
```

or:

```
[comment]
comment_2 = this is also a comment
```

It is possible to define other sections, such as to preserve the content of two different SMTP configurations. For example:

```
[header]
application = pvMail
written = 2014-11-09 11:47:47.709000

[mailer]
mail_transfer_agent = SMTP

[SMTP]
server = smtp.mycompany.com
user = j.o.e.user
password = keep_this_private
port = 465
connection_security = STARTTLS

[work-SMTP]
server = smtp.mycompany.com
user = j.o.e.user
password = keep_this_private
port = 465
connection_security = STARTTLS

[gmail-SMTP]
server = smtp.googlemail.com
user = joeuser@gmail.com
hint = use your gmail account as "user" name
password = keep_this_private
port = 587
authentication = Normal password
connection_security = STARTTLS

[sendmail]
user = joeuser
```

To manage between multiple *SMTP* configurations, copy the settings from the desired section and replace the content of the *SMTP* section. The above example is configured for the work email SMTP server.

KEYWORDS

These keywords (exact spelling) are recognized (others are ignored):

server IP name or address of email server

user username accepted by *server* to send an email

password (optional) if required by SMTP server

port port number to be used

authentication `Normal password`

connection_security `STARTTLS` (SSL/TLS is not available via smtplib)

WRITING THE CONFIGURATION FILE

Under normal use, the application configuration settings file is only read. It is possible to create a new configuration file (in the default location) by running the *`PvMail.ini_config`* program directly from the command line. A new file will be created if none existed. If the file already exists, it will not be modified. The only output from this program will be the absolute path name to the application configuration settings file.

It is possible to edit this file with any text editor.

Tip: It is advised to set the permissions on the application configuration settings file so that only the owner can read the file (owner: read+write). One way to do this on a linux system:

```
[joeuser] $ /path/to/PvMail/ini_config.py
/home/joeuser/.pvMail/pvMail.ini
[joeuser] $ chmod 600 /home/joeuser/.pvMail/pvMail.ini
```

It is also advisable to restrict access to the parent directory of this file (owner: read+write+executable), such as this linux command:

```
[joeuser] $ chmod 700 /home/joeuser/.pvMail
```

On Windows, the default file might be: `C:\\Users\\JoeUser\\AppData\\Roaming\\pvMail\\pvMail.ini`.

It is possible to provide a custom editor (command-line or GUI) for the application configuration settings file. For now, a text editor will suffice.

ALTERNATE CONFIGURATION FILE

An alternate application configuration settings file may be used by setting the **PVMAIL_INI_FILE** environment variable with the absolute file path to the desired file.

Source Code Documentation

handle application configuration settings in a .ini file

To identify the configuration file (and create if it does not exist already):

```
[joeuser] $ pvMail_mail_config_file
/home/joeuser/.pvMail/pvMail.ini
```

class PvMail.ini_config.**Config**

Bases: object

get()
return the chosen configuration dictionary

read()
read the configuration file

setAgent(*agent*)
choose the mail transfer agent

write()
(re)write the configuration file

exception PvMail.ini_config.**NoConfigFile**
Bases: exceptions.Exception

exception PvMail.ini_config.**Unknown_MTA**
Bases: exceptions.Exception

PvMail.ini_config.**main**()

mailer Module

agent	description
sendmail	(linux-only) uses either */usr/lib/sendmail* or */usr/bin/mail*
SMTP	uses smtplib[1]

TESTING THE CONFIGURATION

It is possible to test the email sending using the configuration file. (Alternatively, the GUI has a *File* menu item to send a test email.) First, the help message for the command:

```
[joeuser] $ pvMail_mail_test --help

usage: pvMail_mail_test [-h] recipient [recipient ...]

test the email sender from PvMail 3.1.0

positional arguments:
  recipient   email address(es), whitespace-separated if more than one

optional arguments:
  -h, --help  show this help message and exit
```

To test the email sending using the configuration file:

```
[joeuser] $ python ./mailer.py joeuser@example.com
```

An email message is sent from *joeuser* to joeuser@example.com:

```
1 To: joeuser@example.com
2 Subject: PvMail mailer test message: sendmail
3 Date: Tue, 11 Nov 2014 13:17:31 -0600 (CST)
4 From: joeuser@example.com
```

(continues on next page)

[1] *smtplib*: https://docs.python.org/2/library/smtplib.html

(continued from previous page)

```
5
6  This is a test of the PvMail mailer, v3.1.0
7  For more help, see: http://PvMail.readthedocs.org
```

Source Code Documentation

send a message by email to one or more recipients (by SMTP or sendmail)

exception PvMail.mailer.**MailerError**
Bases: exceptions.Exception

PvMail.mailer.**main**()
user on-demand test of the mailer module and configuration

PvMail.mailer.**sendMail_SMTP**(*subject*, *message*, *recipients*, *smtp_cfg*, *sender=None*, *logger=None*)
send email message through SMTP server

Parameters

- **subject** (*str*) – short text for email subject
- **message** (*str*) – full text of email body
- **recipients** (*[str]*) – list of email addresses to receive the message
- **smtp_cfg** (*dict*) – such as returned from *PvMail.ini_config.Config.get*

  server required - (*str*) SMTP server

  user required - (*str*) username to login to SMTP server

  port optional - (*str*) SMTP port

  password optional - (*str*) password for username

  connection_security optional - (*str*) **STARTTLS** (the only choice, if specified)
- **sender** (*str*) – “From” address, if *None* use *smtp_cfg[‘user’]* value

EXAMPLE:

```
>>> import PvMail.ini_config
>>> smtp_cfg = PvMail.ini_config.Config().get()
>>> recipients = ['joe@gmail.com', 'sally@example.org']
>>> subject = 'SMTP test message'
>>> message = PvMail.ini_config.__doc__
>>> sendMail_SMTP(subject, message, recipients, smtp_cfg)
```

PvMail.mailer.**sendMail_sendmail**(*subject*, *message*, *recipients*, *sendmail_cfg*, *sender=None*, *logger=None*)
send an email message using sendmail (linux only)

Parameters

- **subject** (*str*) – short text for email subject
- **message** (*str*) – full text of email body
- **recipients** (*[str]*) – list of email addresses to receive the message

- **sendmail_cfg** (*dict*) – such as returned from *PvMail.ini_config.Config.get*

  user required - (*str*) username to for sendmail (or similar) program

- **sender** (*str*) – "From" address, if *None* use *smtp_cfg['user']* value
- **logger** (*obj*) – optional message logging method

EXAMPLE:

```
>>> import PvMail.ini_config
>>> sendmail_cfg = PvMail.ini_config.Config().get()
>>> recipients = ['joe@gmail.com', 'sally@example.org']
>>> subject = 'sendmail test message'
>>> message = PvMail.ini_config.__doc__
>>> sendMail_sendmail(subject, message, recipients, sendmail_cfg)
```

PvMail.mailer.**send_message**(*subject*, *message*, *recipients*, *config*)

send an email message

Parameters

- **subject** (*str*) – short text for email subject
- **message** (*str*) – full text of email body
- **recipients** (*[str]*) – list of email addresses to receive the message
- **config** (*dict*) – such as returned from *PvMail.ini_config.Config*

More Information

Functionally based on pvMail UNIX shell script written in 1999.

Summary

Watches an EPICS PV and sends email when it changes from 0 to 1. PV value can be either integer or float.

Note: When "running", wait for trigger PV to go from 0 to 1. When that happens, fetch mail message from message PV. Then, send that message out to each of the email addresses. The message content is prioritized for view on a small-screen device such as a pager or a PDA or smartphone.

---

version control repository

The PvMail project is hosted on GitHub (https://github.com/prjemian/pvMail). You may check out the entire project source code github repository:

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

GitHub has additional advice for alternative methods.

Documentation

Documentation for the PvMail project, maintained using Sphinx (http://sphinx.pocoo.org), is available from:

- http://PvMail.readthedocs.org

TODO items for future releases

see https://github.com/prjemian/pvMail/issues

Authors

author Kurt Goetze (original version)

author Pete Jemian (this version)

organization AES/BCDA, Advanced Photon Source, Argonne National Laboratory

Requirements

requires EPICS system (http://www.aps.anl.gov/epics) with at least two process variables (PVs) where the "Trigger PV" toggles between values of 0 and 1 and the "SendMessage PV" contains a string to send as part of the email message.

requires PyEpics (http://cars9.uchicago.edu/software/python/pyepics3/)

requires PyQt4 (https://wiki.python.org/moin/PyQt)

Change History

3.2.8

- #15 [bcdaqwidgets](https://pypi.python.org/pypi/PvMail) now on PyPI
- #14 pick up GitHub tag versions with [versioneer](https://github.com/warner/python-versioneer)
- #12 build for RTD in a conda environment

v3.2.6 (2015.04.13) bcdaqwidgets has replaced PySide with PyQt4, do that here

v3.2.5 (2014.12.05) make URL be an active (QPushButton) link in About box

v3.2.4 (2014-11-24) Log status updates from `pvMail.PvMail()` in the GUI

v3.2.3 (2014-11-19) email sent in separate thread, PV content shown in GUI when connected

v3.2.0 (2014-11-18) refactor GUI to Qt4 framework

v3.1.0 (2014-11-11) add optional SMTP email support and configuration file

v3.0.5 (2014-11-06) move project to https://github.com/prjemian/pvMail

v3.0.4 (2014-11-05) make docs build at http://pvmail.readthedocs.org

v3.0.3 (2014-07-10)

- Resolve CA monitor problems
- update packaging to contemporary practice

- handle change in PyEpics internal cache of monitored PVs

v3.0.2 (2013-10-18) Simplify startup of `pvMail` by installing launcher as `<python>/bin/pvMail` as part of `setup.py` tasks

v3.0.1 (2012-09-07)

- new home for documentation
- check for existence of sendmail program before trying

v3.0 (2012-06-19) release of new code

2011-11-23 prj complete rewrite using PyEpics and combined GUI (Traits) and CLI

2009-12-02 prj converted to use wxPython (no Tkinter or Pmw)

2005.09.07 kag Initial alpha version. Needs testing.

License

```
Copyright (c) 2009-2017, UChicago Argonne, LLC

All Rights Reserved

PvMail

BCDA, 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
```

(continues on next page)

(continued from previous page)

```
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.

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

1.2 Glossary

CA EPICS Channel Access protocol

CLI command-line interface

EPICS http://www.aps.anl.gov/epics

GUI graphical user interface

IOC EPICS Input/Output Controller, the EPICS server

message PV EPICS PV that provides the text to be sent by email, additional metadata is appended to this text

OS operating system

PV EPICS process variable

PvMail Name of the Python package that provides the `pvMail` executable

pvMail Name of the `pvMail` executable

PyEpics Python package to manage connections with PVs served by an EPICS IOC

PyQt4 Python framework for GUI and other Qt components

trigger PV EPICS PV that signals an email is to be sent

1.3 Dependencies

This software was built with various standard Python packages available in Python 2.7.

Additionally, this program uses:

***PyEpics* (EPICS interface)** http://cars9.uchicago.edu/software/python/pyepics3/

***PyQt4* (Qt binding)** https://wiki.python.org/moin/PyQt

Both of these are available for easy_install from the Python Package Index (http://pypi.python.org/pypi).

Installation

The most recent public release of this software is available from the Python Package Index[1] by either:

```
[joeuser] $ pip install PvMail
```

[1] https://pypi.python.org/pypi/PvMail

or:

```
[joeuser] $ easy_install PvMail
```

while the source is available from GitHub: https://github.com/prjemian/pvMail

Python Module Index

p

Index

P

R

S

T

U

W