PyModerator 0.3
User's Guide

Contents

Introduction

PyModerator is used to moderate Usenet newsgroups. It is written in Python, a high level scripting language. Python has been ported to many different operating systems, and PyModerator should run on any system that can run Python and the Python GUI module Tkinter. Fortunately, the list of systems PyModerator should run on includes the most popular: most Unix systems, most Microsoft Windows systems, and the Apple Macintosh (although use on the Macintosh has never been tested).

A single moderator (solo moderation) or a group of cooperating moderators (co-moderation) may use PyModerator to moderate one or more newsgroups. While it can be used very effectively to solo moderate, many of its features were specifically designed to aid co-moderation.

PyModerator is really composed of two processing pieces: a client Graphical User Interface (GUI) process and a server (central repository) process. The two processes communicate via a TCP/IP socket, which allows them to run on different machines. The server process listens on a TCP/IP service port for client connections. A single server can handle any number of clients and therefore any number of concurrently active moderators. If you have several co-moderators and one of them can leave their machine on (and accessible to the net), then that moderator can run the server process continuously and all the co-moderators can connect to that central server using only the client program. PyModerator features are best utilized if one person hosts a server (runs serverStart.py) and each of the co-moderators run a client (runs clientStart.py). Otherwise each co-moderator can run in solo-moderation mode by running startBoth.py which starts a local server and a local client. But if each co-moderator runs a local server, it will unfortunately segment the archives across all their machines.

Getting Started

A note on terminology: I will be using the word *directory* synonymously with the word *folder* in several places below. The word folder is Microsoft's term for a Unix directory.

1. Installation

The first step is to download the PyModerator.zip file and then unzip the file in a drive or file system with enough room to hold archived messages (which may accumulate to many megabytes after a few years). Unzipping PyModerator.zip will create a PyModerator directory (with some subdirectories). In general you should install to a user account that has the minimum authorizations needed to run (it is recommended that you not run either the client or server as root on Unix systems).

2. Starting and Stopping the Server

If you are a co-moderator and someone else will be running the PyModerator server, you may skip to step 3 "Starting and Setting Up the Client" below.

If you are solo moderating a newsgroup or are will be the moderator hosting the PyModerator server, then you are responsible for starting the server, making it "visible" on the Internet to the other moderators, and keeping it running either all the time or at times previously arranged.

On Windows systems: Open the PyModerator folder and double-click on serverStart.py. This will open an MS-DOS text display window. You may iconify it to move it out of the way. To stop the server you may click on the close window button. Confirm closure when asked. Do *not* stop the server when you believe it is performing disk activity! You can corrupt the data files if you do that.

On Unix systems: Log in using the account you want to run the server under and cd into the PyModerator directory. Then issue the following command to run the PyModerator server in the background:

```
nohup serverStart.py &
```

To stop the server you may issue a kill command to kill that Python process. Do *not* kill the server when you believe it is performing disk activity! You can corrupt the data files if you do that.

On Unix and Windows: The server will create a ServerDB directory if it does not already exist and initialize all the files needed in that directory. As newsgroups are added the server will create subdirectories named the same as the newsgroup names.

3. Starting and Setting Up the Client

On Windows systems: Open the PyModerator folder and double-click on clientStart.py. This will open an MS-DOS text display window and the PyModerator window.

On Unix systems: Log in using the Unix user ID you want to run the client under and cd into the PyModerator directory. Then issue the following command to run the PyModerator client:

```
clientStart.py
```

On Unix and Windows: You may need to modify the font size in order to view the entire application window on your screen. See section 8 "Changing the Font Size" on page 21 for details.

The client will create a ClientDB directory if it does not already exist and initialize all the files needed in that directory.

Before the client can connect to the server you need to tell the client the IP address of the server, among other details. From the main menu select **Manage → Login Fields** and enter the IP address of the PyModerator server in the Service Host field. A comprehensive discussion on what to enter is provided in the "Login Fields…" section on page 7. Once you have made the appropriate changes, you are ready to log into the server. Select **Session → Login** from the main menu and proceed to log in (see the "Login…" section on page 6 for more information).

4. Server and Newsgroup First Steps

When you are logged in as root for the first time your next step is to select the menu item **Manage → Server Info…** and set the fields in that dialogue window accordingly. See the “Server Info…” section on page 9 for more information.

You are now ready to create newsgroup and moderator records. To create a newsgroup record select menu item **Manage → Newsgroups…** and see the “Newsgroups…” section on page 10 for information on what you need to put in the fields. To create moderator records select menu item **Manage → Moderators…** and see the “Moderators…” section on page 13 for information on what you need to put in the fields.

You are now ready to have moderators moderate! Log out and log back in using one of the new moderator accounts and select the newsgroup to moderate from the **Moderate** menu. Be sure to review this document to get acquainted with all aspects of PyModerator since you will almost certainly want to do things like create rejection types and perform other maintenance activities.

The Dropdown Menu Selections

The main menu bar at the top of the PyModerator client has five drop down menus: **Session**, **Manage**, **Message**, and **Moderate** on the left and Help on the far right. The menu selections are explained in the next sections.

1. Session

The **Session** drop down menu provides options for starting and stopping the login session between the client and the server, as well as updating the display.

a. Login…

You must login to the server before attempting to perform any moderation. You must have properly filled in the required fields in the **Manage → Login Fields** dialogue box first before attempting to use this selection. Once selected, the PyModerator client will attempt to connect to the server (which normally takes a fraction of a second) and if successful will display the **Connect and Login to Server** dialogue box. The ***Moderator ID*** field’s default value is set to the value that was entered into the **Manage → Login Fields** dialogue, but you may change it. You should then enter the proper password and click on OK; otherwise click on Cancel to abort the login. The first time a server is started it creates a moderator ID *root* that has no password. One of the first tasks the root moderator should do is change that password to something more secure!

Once a valid moderator ID and password have been entered, the PyModerator server will connect to the POP mailbox, read any new messages that have been sent to the moderator's initial newsgroup (if any), and then send the information to the client for display. These operations may take a few seconds so a delay at login may be noticeable.

b. Refresh from Server

The PyModerator client only updates the whole display when you perform some action that changes the state of one or more messages on the server. Likewise, new messages are downloaded from the POP mailbox only when a moderator logs in. To get the server to reread the POP mailbox and have the client update its view from the server, you may select this option at any time. It is normally fairly quick.

c. Logout

You may use this selection if you wish to keep the PyModerator client running but wish to logout and disconnect from the server. A slight time saver over using Quit and restart should you wish to logout using one moderator ID and log back in under another moderator ID.

d. Quit

You may use this selection if you wish to logout and disconnect from the server and also wish to terminate the client program. This is the normal way to exit. The close window button in the upper right corner of the application window will perform the same operation.

2. Manage

Aspects of the system that only occasionally need management changes have been collected under the **Manage** drop down menu.

a. Login Fields…

Before you can log in, you must first tell the client what machine the server is running on (so it can connect), plus a few other pieces of information. The information you enter into these fields is stored on the client machine in the file `ClientDB/client.cfg`. This is not a text file, so will look like gibberish in a text editor.

This selection brings up the **Edit Default Login Fields** dialogue box and the explanation of what goes into each of the fields is as follows:

Moderator ID

You may place in this field the moderator ID that you most often login as. This is what shows up by default in the **Login** dialogue box. If no one has ever logged into the server before, you will need to log in using the moderator ID *root*.

Service Host

This is the IP address of the machine where the PyModerator server is running. It may be entered as a fully qualified host name (e.g. fredsmachine.somedomain.com) or using a dotted IP address (e.g. 123.45.67.89). If you are solo moderating, or are for any reason running the server on the same machine as the client, then the IP address alias for local host, 127.0.0.1, may be entered. This is the default value.

Service Port

By default the PyModerator server listens on port 11556 for client connection requests. This value should normally not be changed unless the server's listening port has been changed. The most common reasons for changing this are that the port is already in use by another program on the server machine (rare) or that the server machine is behind a firewall and only certain port numbers are allowed open.

NNTP Host

You will also see a request for the NNTP host in the **Manage → Server Info** dialogue box. Why does PyModerator seem to be asking for the same information in two places? Because it is being stored in two different places! The value entered in this field is stored on the client machine in the `ClientDB/client.cfg` file and the value entered in the **Server Info** ***NNTP Host*** field is stored on the server machine in the `ServerDB/server.cfg` file. Suppose three moderators, Curly, Moe, and Larry, are co-moderating a newsgroup and Larry is hosting the PyModerator server they all connect to. When Curly goes to approve a post, should it be posted from the client program via Curly's Usenet service provider or from the server program via Larry's Usenet service provider? With PyModerator you can choose to do it either way; in the **Manage → Newsgroups** dialogue you would set the ***Post From Server*** field to *Yes* if you wish to have approved messages posted from the server using the server's NNTP host information and set to *No* if messages are to be posted from the clients using the client NNTP host information. If you are solo moderating (or wish to always have messages posted from the server), you should enter the NNTP information in the **Manage → Server Info** dialogue box, set ***Post From Server*** to *Yes* for your newsgroup(s), and leave all the NNTP fields in the Login Fields dialogue box at their default values.

NNTP Port

This should left at the default value of 119 unless your client is connecting to an NNTP server listening on a non-standard port.

NNTP User

If the NNTP server that the client connects to requires user and password authentication, then the user ID or name should be entered in this field. If no authentication is being used you *must* leave this blank.

NNTP Password

Enter the password for the user ID or name entered for the above field. If no authentication is being used you *must* leave this blank.

SMTP Host

When you use PyModerator to reply to posters by e-mail, or reject messages using rejection reasons that generate e-mail to the submitter, then the mail will be sent from the client machine to the destination address by way of an e-mail relay machine called an SMTP host. The ***SMTP Host*** field is the name of the machine that is used for outbound mail. You must enter the name of the SMTP relay host either as a fully qualified host name (e.g. mail.somedomain.com or smtp.somedomain.com) or as a dotted IP address (e.g. 123.45.67.89).

b. Server Info…

When you log in as *root* or with a superuser moderator ID, you may select the **Edit Server's Connection Information** dialogue box to modify the following fields on the server:

NNTP Host

This is the NNTP host to which approved messages are posted. This is only used if the ***Post From Server*** field for the newsgroup in question is set to *Yes*. See **NNTP Host** on page 8 for a more detailed discussion.

NNTP Port

This should left at the default value of 119 unless the PyModerator server is connecting to an NNTP server listening on a non-standard port.

NNTP User

If the NNTP server that the PyModerator server connects to requires user and password authentication, then the user ID or name should be entered in this field. If no authentication is being used you *must* leave this blank.

NNTP Password

Enter the password for the user ID or name entered for the above field. If no authentication is being used you *must* leave this blank.

SMTP Host

This should be set to the SMTP relay host IP address for outbound e-mail from the PyModerator server. There is only one case in which the server will ever send e-mail: when it has a scheduled posting to make but is not allowed to post! If the scheduled posting capability is being used for messages on a newsgroup (see **Periodic Posts…** on page 17) and the ***Post From Server*** field for the newsgroup in question is set to *No*, then the only way for the server to "post" is to e-mail the post to the newsgroup's submission address. It will do this as soon as it discovers the scheduled posting time has elapsed. It is then up to the moderator to approve the periodic post.

You must enter the name of the SMTP relay host either as a fully qualified host name (e.g. mail.somedomain.com or smtp.somedomain.com) or as a dotted IP address (e.g. 123.45.67.89).

Idle Timeout (secs)

When a client that is connected and logged in to the server has been idle (i.e. no data traffic between client and server) longer than the number of seconds indicated in this field, the server will log the user out and drop the connection. *Keep in mind that if a moderator spends too much time composing an e-mail reply or otherwise reading or editing a submitted post, no traffic is happening between the client and server and the user could be timed out and their work lost!* It is therefore advised that this value be increased, perhaps to an hour (3600). Initial default value is 15 minutes (900 seconds).

c. Change Password…

The **Change Moderator's Password** dialogue box allows a moderator to change his or her password or a superuser to change anyone's password (but only root may change root's password). Enter the ID of the moderator whose password is to be changed and the new password into the ***Password*** and ***Password Again*** fields (since you can't see what you type in password fields, entering the password twice confirms that you've not accidentally mistyped the password the first time).

d. Newsgroups…

This brings up the **Select Newsgroup to Manage** dialogue box that displays a scrollable list of the names of newsgroups that have been created on the server for moderation and allows you to create, edit, or delete newsgroups from the server's list. Clicking on the *Cancel* button closes this dialogue box.

To create a newsgroup record: you need to click on the *Create* button. This will bring up the **Manage Newsgroup** dialogue box. The values you need to put in the requested fields are described below. Once you have filled in the fields, you may click on *OK* to create the newsgroup moderation record or you may click on *Cancel* to abort the operation. Only a superuser may create a newsgroup.

To edit a newsgroup record: you must click on the row you want and then click on the *Edit* button or you may double-click on the row you want. Either sequence will bring up the **Manage Newsgroup** dialogue and will display the current field values. Modify any of the fields and click *OK* to make your changes permanent or *Cancel* to abort the changes. Only a superuser may edit a newsgroup.

To delete a newsgroup record: you must click on the row you want to delete and then click on the *Delete* button. A confirmation dialogue box will request whether you really want to do this. Only a superuser may delete newsgroups.

The **Manage Newsgroup** dialogue box fields and their meanings are:

Newsgroup ID

This is a required field and *must* be set to the *name* of the newsgroup as it appears in the NNTP server. For example, if the newsgroup to be moderated is misc.business.moderated, then this field must be set to misc.business.moderated, not mbm, or m.b.moderated, or any other variant.

POP Host

This must be set to the name of the POP server to which newsgroup submissions are eventually e-mailed. You must enter the name of the POP host either as a fully qualified host name (e.g. pop3.somedomain.com or mail.somedomain.com) or as a dotted IP address (e.g. 123.45.67.89).

POP Port

This is the POP server’s listening port. Normally POP servers listen on port 110. Generally you should keep this field at the default value of 110 unless you know that the POP server is listening on a non-standard port number.

POP User ID

This is the user ID that must be sent to the POP server to authenticate and log in to download incoming e-mail submissions. It is often (but not always) the same value as the user name of the submission e-mail address. That is, if the submission address for misc.business.moderated is mbm@somedomain.com, then the value of this field should probably be set to mbm.

POP Password

This is the password that must be sent to the POP server to authenticate the user ID. This has to be saved in plain text in the `ServerSB/server.cfg` file, so some care should be taken to make sure that file is not easily accessible (e.g. for example, it should not be in a shared network folder).

Allow Cross Posts

If you have established a policy of not allowing cross posting between newsgroups, then you should click on the Yes button for this field. Then whenever a message is received that is cross-posted, the ***Bad Cross Posts*** row will display in angry red the names of the cross-posted newsgroups. *This is strictly an advisory notification; it is up to the moderator to reject the post, approve it as is, or trim the newsgroups header.*

Round Robin Assignment

When co-moderating a newsgroup, you may wish to have the load fairly distributed to each of the moderators. In that case you should select the *Yes* button for this field. Incoming messages are assigned to each non-vacationing moderator in turn (much as cards are dealt out to card players). *Assignments are strictly advisory; that is, any moderator may ignore the assignments and perform actions on any message in the newsgroup*. If the *No* button is selected then messages are assigned to the *root* user and any moderator may act on any message.

New Material Yellow %

This is the new material yellow warning threshold field. For newsgroups that have a policy of rejecting or trimming posts that contain excessive quoted material, PyModerator provides a warning indicator for such posts. If the computation of new material is below the percentage indicated in this field (but above the red percentage), the background of the ***New Material*** display field will turn yellow. The default value for this field is 25%. *This is strictly an advisory notification; it is up to the moderator to determine what action should be taken with the message.*

Any line that begins with the characters ">" or "|" is considered quoted material. All other lines, including attribution lines and signatures, are considered new material. White space characters, such as spaces, tabs, line feeds, and carriage returns, are ignored in the computation. So blank lines or heavy indentation will be ignored in both the new and quoted sections. In other words only the printing characters are included in the computations.

New Material Red %

This is the new material red warning threshold field. See the discussion above in the ***New Material Yellow %*** field for background. When the computation of new material is below the percentage indicated in this field, the background of the ***New Material*** display field will turn red. The default value for this field is 10%.

Post From Server

When a message is approved, it may be posted to Usenet using either the Usenet news server specified by either the PyModerator client or the server. A more complete discussion may be found for the **NNTP Host** field on page 8. Click on *Yes* to post messages from the PyModerator server to the NNTP host indicated by the PyModerator server’s data file or *No* to post messages from the PyModerator client to the NNTP host indicated by the moderator’s PyModerator client data file.

Create Time

This is a display-only field that indicates when the newsgroup record was created.

Creator ID

This is a display-only field that indicates which moderator created this newsgroup moderation record.

Moderators

A button is displayed for every moderator record that has been created on the PyModerator server. You may select or deselect the moderators for a newsgroup by clicking on these buttons. Buttons will appear when moderators have been created and disappear when moderators have been deleted. You will need to close and reopen the **Manage Newsgroup** dialogue to pick up changes to the list of moderators.

e. Moderators…

This brings up the **Select Moderator to Manage** dialogue box that displays a scrollable list of moderator IDs that have been created on the server and allows you to create, edit, or delete moderators from the server’s list. Clicking on the *Cancel* button closes this dialogue box.

To create a moderator record: you must be a superuser and you need to click on the *Create* button. This will bring up the **Manage Moderator** dialogue box. The values you need to put in the requested fields are described below. Once you have filled in the fields in the **Manage Moderator** dialogue, you may click on *OK* to create the moderator or you may click on *Cancel* to abort the action.

To edit a moderator record: you must click on the row you want and then click on the *Edit* button or you may double-click on the row you want. Either sequence will bring up the **Manage Moderator** dialogue and will display the current field values. Modify any of the fields and click *OK* to make your changes permanent or *Cancel* to abort the changes. Moderators may always change their own record (except they can’t change their user type!) but only a superuser may change another moderator’s type.

To delete a moderator record: click on the row you want to delete and then click on the *Delete* button. A confirmation dialogue box will request whether you really want to do this. The root user can never be deleted, and only a superuser may delete moderators.

To explain the **Manage Moderator** dialogue box fields, we assume it is your moderator record that is being created or edited:

Moderator ID

This is your login ID. It may be any value but is conventionally set to some value related to your name or any other preferred “handle”. Moderator IDs must be unique. The moderator ID *root* is special: it may not be deleted and has a handful of special capabilities no other user type has.

E-Mail From Address

This is your e-mail address. *This is a required field.* It is essential and is used in two places: in the “From” header on e-mails, and in the “Approved” header on all approved posts. If it is not filled in then approved postings will have blank values in the “Approved” header and NNTP servers will tend to silently ignore such postings. So please fill this in.

Full Name

This is your name, as it will appear on “From” headers in e-mails.

On Vacation

When you plan to be away from moderation duties for any extended period of time, you can set this value to *Yes*. If you are co-moderating one or more newsgroups where round robin assignment is active, setting this to *Yes* will take you out of the round robins. A “**Vac**“ indicator will also appear next to your moderator ID in the moderator status row to inform your fellow co-moderators that you are unavailable.

User Type

Moderators come in three types: *Moderator*, *Superuser*, and *Guest*. A *Guest* may log in and out and view messages, but is otherwise prohibited from doing any other action. A *Moderator* is allowed to do all the normal moderation activities, but a handful of dangerous actions are not allowed to plain moderators. A *Superuser* can do almost anything, except for a couple actions that are reserved only to the *root* user.

Create Time

This is a display-only field that indicates when the moderator record was created.

Moderates

A button is displayed for every newsgroup record that has been created on the PyModerator server. Select or deselect those newsgroups that you wish to moderate by clicking on the appropriate buttons.

Last Login At

This is a display-only field that shows when you last successfully logged in.

Last Logout At

This is a display-only field that shows when you last logged out.

f. Update Valid Groups…

To assist moderators in knowing whether messages are being cross-posted to another moderated newsgroup or to an invalid newsgroup, the PyModerator server maintains a list of valid newsgroups. This list is initially empty and must be populated from the NNTP server using this menu selection. Every six months, or as conditions warrant, this menu selection should be run. The newsgroup list can be quite long and it can take many minutes to update the valid group. And in the mean time this action will preempt all other actions from all other clients from occurring. This makes the server appear "frozen" to other users while the action is in progress. Therefore you should perform this action only as needed.

Only a superuser may execute this command.

3. Message

Aspects related to how messages should be manipulated or that are only occasionally used are gathered under this menu selection.

a. Statistics…

This displays the number of times each moderator has rejected, limboed, and approved messages, showing the breakdown by each rejection type. Rows and columns are added only as moderators actually use the various rejection types.

b. Rejection Types…

This brings up the **Rejections** dialogue box that displays a scrollable list of rejection IDs that have been created on the server and allows you to create, edit, or delete rejection reasons from the server's list. Clicking on the *Cancel* button closes this dialogue box.

To create a rejection type record: you need to click on the *Create* button. This will bring up the **Manage Rejection** dialogue box. The values you need to put in the requested fields are described below. Once you have filled in the fields in the **Manage Rejection** dialogue, you may click on *OK* to create the rejection reason or you may click on *Cancel* to abort the action. When you create a rejection reason, a new button is created in the rejection button row with the rejection ID, so it is a good idea to keep the rejection ID short. Guests may not create rejection records.

To edit a rejection type record: you must click on the row you want and then click on the *Edit* button or you may double-click on the row you want. Either sequence will bring up the **Manage Rejection** dialogue that will display the current field values. Modify any of the fields and click *OK* to make your changes permanent or *Cancel* to abort the changes. Only superusers and the original creator of a rejection record may edit that record.

To delete a rejection type record: you must click on the row you want to delete and then click on the *Delete* button. A confirmation dialogue box will request whether you really want to do this. Once deleted, the corresponding button is removed from the rejection button row. Only superusers and the original creator of a rejection record may delete that record.

The Manage Rejection dialogue box fields and their meanings are:

Rejection ID

This is a unique ID assigned to the rejection type. It may be any value, but short one or two word descriptive titles are best. These should be kept short since the ID appears in the rejection buttons and screen real estate is at a minimum. Some common IDs are "Spam", "Off Topic", "Other", "Other (No reply)", and so on. PyModerator will automatically place the word "**Reject:**" before the ID when it creates the button, therefore you should not prefix your rejection ID with that word.

E-Mail Response Template

If a standard rejection notice is to be e-mailed to the poster, the text of that response should be entered in this text box. E-mail will be sent if the ***Default Action*** field is set to ***Limbo & Reply*** or ***Reject & Reply***. At the time a message is rejected and an e-mail notification is to be sent, a dialogue box will show the rejection e-mail message and you will have an opportunity to modify the e-mail text to personalize it or cancel the action. Typical personalized text you might want to include would be specific suggestions on what the poster can do to make the message approvable or to sign your name.

Default Action

You must pick a rejection action from one of the following buttons:

> ***Limbo & Reply***: This will move the message into the **Limbo** list, mark its status as in limbo, and send the poster an e-mail message. The e-mail will include the text from the ***E-Mail Response Template*** and the original e-mail will be appended. This action is useful for rejection types dealing with borderline cases where a person may be given an opportunity to justify why their message should be approved. It might also be useful for a rejection type that

automatically sends a courtesy signal to a poster that their message is being delayed while the moderator ponders what to do with it.

Reject & Reply: This will remove a message from the **Review** or **Limbo** list, mark its status as rejected, and send the poster an e-mail message. It is the most common rejection action and is therefore the default setting.

Reject: This will remove a message from the **Review** or **Limbo** list and mark it as rejected. Note that it does *not* send any e-mail. The most common uses for this action are for rejecting spam and rejecting messages where a notice to the poster would be redundant or unnecessary. It is useful to have at least one rejection type that covers such miscellaneous summary rejections.

Create Time

This is a display-only field that indicates when the rejection type record was created.

Created By

This is a display-only field that indicates which moderator created this rejection type record.

c. Periodic Posts…

This brings up the **Periodic Message Postings** dialogue box that allows a superuser to select previously approved messages for automatic repeated posting. It is typically used to periodically post newsgroup FAQs and posting policies. Any number of messages may be posted, each at different frequencies and phases.

To have a message posted at repeated intervals, select it from the **Review**, **Limbo**, **Archive**, or **Search Archive** results list. Then click on the *Add* button. The message will then be added to the **Repeated Message** list. By default the message will be set to repeat every 30 days, starting 30 days from the point in time you added it to the **Repeated Message** list. To change the repetition time, click in the ***Repetition time, in days*** field and enter how many days apart you wish to repeat the posting. You may also click on the ***Next send time, in days*** field to set when the new cycle will begin. Fractional days may be entered in either field. When you have set the fields to the values you want, you *must* click on the *Update* button to store your changes.

You may change the ***Repetition time, in days*** or ***Next send time, in days*** fields at any time. Just remember to click on the *Update* button to store the changes.

If you wish to remove a message from the **Repeated Message** list, click on that message’s row and then click on the *Drop* button.

When you have completed your view or changes to the **Periodic Message Postings** dialogue, you may close it by clicking on the *Close* button.

d. Delete…

This is used to delete a message entirely from the archive. It may only be used by superusers and should be used with great caution. However, it will not delete the message ID. The only time you should use this is to save disk space, particularly to delete very large messages such as certain types of spam.

4. Moderate

You must use this drop down menu to switch between the newsgroups you wish to moderate. The entry ***None*** may be selected to deselect from moderating any newsgroup. ***None*** is the default entry when a moderator is created and one of the first things you need to do on your first login is to select the newsgroup you wish to moderate. On subsequent logins you will find yourself moderating the last newsgroup you selected on this drop down menu. If the only entry you can select on this menu is ***None*** then you have not been assigned any newsgroups to moderate.

5. Help

This provides some basic online help. The **About** selection provides version number and copyright information, if any. The ***Balloon Help*** button toggles the balloon help text boxes on and off. When the mouse pointer is paused over some screen features, balloon help text boxes will appear that attempt to describe those features (such as buttons, menus, and so on).

The Command Bar Buttons

With the exception of the **Search Archive** button, pressing a button on the command bar (**Approve**, **Limbo**, etc.) will apply that operation to the *currently selected message*. The message that appears in the main window is considered the currently selected message.

1. Approve

This will approve the currently selected message so long as it is in the review or limbo state. Messages that have been previously rejected cannot be approved. To approve a previously rejected message, you will need to first move it into limbo.

2. Limbo

This will place the currently selected message into the limbo state. This will prompt for a limbo comment that will be placed in the **Event Detail** record for the limbo event. Messages that are in the review state or have been previously rejected may be placed into limbo. The limbo state is intended as a temporary holding state for messages that moderators wish to ponder over. How moderators wish to use limbo, if at all, is up to them.

3. Reassign

Whether or not you use round robin assignment, you may use this command to reassign any message (even those not assigned to you) to any other moderator ID, including root. This button brings up the **Reassign Message to Selected Moderator** dialogue box that displays a scrollable list of moderator IDs. Click on the moderator ID to which you want to assign the message to and then click on *OK*. Otherwise click on *Cancel* to close the window. *Keep in mind that assignments are strictly advisory; PyModerator does not restrict actions on the message to the assigned moderator.*

4. Add Event Note

This allows you to add an event note to the message's **Event History**. It brings up the **Enter Event Detail, If Any** dialogue box. Enter the text of the note into the window and click on *OK* to save your note or *Cancel* to abort the action. An event note may be used for various purposes, including notes to oneself or queries or comments to one's co-moderators.

5. E-Mail Poster

This allows you to send an e-mail reply to the message's poster. If there is a "Reply-To" header, it sends to that address, otherwise it send to the address in the "From" header. Clicking this brings up the **E-Mail Reply** dialogue box. It includes the text of the original message, which you may edit or delete. Enter the text of your message, modify any of the headers as appropriate, and click on *OK* to send the message or *Cancel* to abort the action. The message will be e-mailed through your (client's) SMTP mail relay host. If the e-mail bounces, the bounce will go to the address that you set in the "From" header.

6. Search Archive

This allows you to search the message archives. Having access to thousands of messages but no way to quickly find items of interest would make having an archive almost useless. This command gives you pattern matching search capabilities. Clicking on this button brings up the **Search Archive** dialogue box. At its simplest, you would type in a search expression (often just a word or a name) in the ***Search Expression*** field and click on the *Search* button. After a few moments the messages that had text that matched the search string would display in the scrollable **Results** window. Click on the rows in the **Results** window for the messages that you are interested in viewing.

The pattern match search is performed on every message, so if the archive is very large, the search can take some time. To speed up searching you should select only those parts of a message record that are relevant. For example, if you are looking for all messages whose subject line contains "Three stooges are bad comedy" you should only select *Incoming Headers* in the ***Search In*** field. You would then enter that phrase as your search string. Another example: you have created a rejection ID called Spam for spam messages and you wish to search all messages that were rejected for that reason. You would select only *History Events* in the ***Search In*** field and put the phrase "Reject: Spam" in the ***Search Expression*** field, then click on the *Search* button. After a few moments you would be displayed all the messages that were rejected as spam.

Search expressions can be quite complex, but the most common are simple keyword searches. Generally, space characters *are* significant. Also, an expression may match something longer; e.g. the search for "eject" would match "eject", "ejection", "reject", "rejection", and so on. If you wish to find messages that match one of several possible words, you must add the "|" (vertical bar symbol) between them. So "Curly|Moe|Larry" would find all messages in which any one of those names appeared.

The search mechanism is based on the Python "re" module. It is not possible to cover here all the possible search expressions that are allowed by that module, so I refer you to two online resources:

The *Regular Expression Syntax* section in The *Python Library Reference*:

http://www.python.org/doc/current/lib/re-syntax.html

A. M. Kuchling's introduction to the subject, *Regular Expression HOWTO*, may be easier to read and may be found online at:

http://py-howto.sourceforge.net/regex/regex.html

The meaning of the **Search Archive** dialogue box fields are as follows:

Search Expression

This is the search pattern you are looking for matches on. The valid syntax is discussed above.

Ignore Case

If you wish to have the search match the case of the characters exactly as you entered them, then you would click on *No*. Most of the time you don't care about whether the words are in upper or lower case, so the default is set to *Yes*, indicating a match will be considered made if the words are the same, even if the case of the characters don't match.

Search In

This allows you to search in different parts of the message record. It is normally the case that you want to search only in the ***Incoming Headers*** and the ***Incoming Bodies*** of the messages, so these are set by default. Searching in both the outgoing bodies and headers and the incoming bodies and headers is often expensive in computation time and generally redundant.

Please note that when searching ***Incoming Headers*** and ***Outgoing Headers***, the header type (e.g. Subject, Received, etc.) is not included in the search.

When selecting the ***History Events*** button, the text in all the rows in the **Event History** window and all the corresponding **Event Detail** text is searched in all messages.

Results

This is the scrollable list of messages that matched the search expression. To view a matching message, click on the row of interest to display that record.

Note that the *Close* button closes the **Search Archive** window but it does not clear the results. Next time you click on the **Search Archive** button, the window will show the results from the previous search. To replace the results, merely enter new search criteria and click on the *Search* button.

The Rejection Buttons

Below the command button line are displayed the buttons for the rejection IDs that have been created for this newsgroup. They are listed in alphabetic order.

The Main Display Windows

7. Copy and Paste under MS Windows and X Windows

A few notes are in order on copying and pasting of text since some conventions of PyModerator are slightly nonstandard. Use the usual left mouse-button dragging under MS Windows or X Windows to highlight text you want to copy (or block delete). On MS-Windows use control-C to copy the highlighted region to the internal copy buffer and control-V to paste from copy buffer. Under X-Windows highlighted text is automatically placed into copy buffer and you use the middle mouse button or control-V to paste from the copy buffer. *Note that copying from some text windows is currently broken on the MS Window version; this will be corrected in the next release of PyModerator.*

8. Changing the Font Size

You may find that the PyModerator GUI display is either too large to fit on your screen or that the character font is too small. To adjust the size of the display fonts, and in turn the size of the display windows, you must create a font configuration file. You will find a sample configuration file called `optionDB_sample` in the PyModerator source directory. Copy it to another file called `optionDB.txt` and open that text file up in a text editor such as WordPad on MS Windows or vi or emacs on Unix systems. In it you should find the following:

```
*Font:                  Arial 8 bold
*EntryField.Entry.Font: Courier 8
*Listbox*Font:          Courier 8
*Text*Font:             Courier 8

# *Font:                    Arial 10 bold
# *EntryField.Entry.Font:   LinePrinter 10
# *Listbox*Font:            LinePrinter 10
# *Text*Font:               LinePrinter 10
```

The lines that begin with the "#" character are comment lines. These lines indicate the font type (e.g. Arial, Courier, LinePrinter, Times Roman, etc.) and the font size (e.g. 8, 9, 10, 12, etc.) for each of the major text display classes. You may wish to experiment with the font types (use fixed width fonts for those classes of text that were originally specified with Courier) as well as the sizes. Feel free to use the comment character at the beginning of any lines you wish to have ignored. Not all font sizes are available on all platforms, so some font size changes may be rounded to the nearest available font size. Or a default font type may be used when a non-existent font name is entered. Since the optionDB.txt file is read only when the PyModerator client is started, you will need to stop and restart the client each time you make a change to the file.

On startup the PyModerator client looks for a font database file called optionDB, optionDB.txt, or optionDB_sample, in that order. As soon as it finds one of those files it opens it and sets the fonts accordingly.

9. Moderator Login Status Row

Below the rejection buttons (if any) there appears the moderator login status row. This displays some status info on all the co-moderators assigned to the newsgroup. The syntax and meaning of each part of the status box is as follows:

("**S**" | "**M**" | "**G**") "**:**" <*Moderator ID*> ("**on**" | "**left**") <*number*> "**hrs ago.**" ["**(Vac)**"]

If the indicated user is a superuser the first letter is **S**, if the user is a "normal" moderator it is **M**, and if the user is a guest it is **G**. The user's moderator ID is then shown. If the user is currently logged in, the text in their status block is brown and it displays how many hours ago the user logged in; e.g. "Jim on 0.4 hrs ago." If the user has logged out, it indicates how many hours have elapsed since their logout. And if the user has clicked on the ***On Vacation*** button for their moderator record, the **(Vac)** part is displayed.

10. The Left Column: Message Selection Lists

The three column windows on the left side of the PyModerator application window are all selection lists. Clicking on a row in any of them loads that message for display.

a. Review

When messages are read from the POP server their state is set to review and they are displayed in the **Review** selection window. Messages that have been assigned to you will always be displayed at the top and will be sorted in message ID order. Messages assigned to others will be sorted first by moderator ID and sub-sorted by message ID.

The first column shows the assigned moderator ID (root if no round robin active). The second column is the message ID. The third column shows the first 12 characters of the subject line.

b. Limbo

This shows the messages that are in the limbo state. Layout and ordering is otherwise identical to the **Review** selection window/

c. Archive

This shows the entire archive of message IDs. Because there may be thousands of message records in the archive, it only loads and displays detail 30 rows at a time. The column format is as follows:

```
MM/DD hh:mm Nnnn Ttttttt [Aaaaaaaaaaaaaaaaaaaaaaaaa] Sssssssssssssssssssssss
```

Where "`MM/DD hh:mm`" is the month, day, hour, and minute that the message was placed into the archive. It is *not* the date and time the message was sent or received by the POP server. "`Nnnn`" is the message ID. "`Ttttttt`" is the current state of the message and is one of **Review**, **Approved**, **Rejected**, or **Limbo**. "`Aaaaaaaaaaaaaaaaaaaaaaaaa`" isn't a scream; it is the first 25 characters of the poster's name and e-mail address (but it is truncated to the nearest whole word). "`Sssssssssssssssssssssss`" isn't the sound of a memory leak; it is the first 24 characters of the subject line.

Because the columns are too wide to fit in the archive window, you must use the horizontal scroll bar to scan the rows to read them. The vertical and horizontal scroll bars stay where you place them, so if you wish to scan for messages from a certain poster, you can move the horizontal scroll bar to bring the names into view and then scroll vertically to scan.

You can load rows that have not been loaded, (dashes appear in the date and time columns) simply by clicking on them. That will load the message selected for display and also load in 30 rows in the Archive Window from that message and on down. This is called on demand row loading and is a super clever feature of PyModerator that cuts down on the network traffic needed to support the archive window (at least the author thinks so).

11. The Center Column: The Selected Message

The center and right columns of the application screen display the contents of the message record. Obviously there is more to a message than just the headers and the message body that the poster sent. A message also has a history and a processing state that moderators may change.

a. Message Received As

This is a scrollable display-only window that displays the message as it was originally received. It shows all the headers and the body of the message. However, it does rearrange the headers in the following way: it places the “Received:” headers at the top if they are not already at the top (but preserves their original order), and places the remainder of the headers below them but in alphabetic order. The body of the message is left unchanged.

b. New Material/Bad Cross Posts

The **New Material** line shows what percentage of the body of the message appears to be new material. Lines beginning with the “>” or “|” characters are considered quoted text (old material) and all other lines are considered new material. Further details on this field may be found in the ***New Material Yellow %*** section on page 12.

The **Bad Cross Posts** line displays the newsgroups to which the post appears to be improperly cross-posted to. After each newsgroup name you will see a colon followed by a letter, such as **Y**, or **M**, or sometimes even the word **None**. These are newsgroup status indicators and their meanings are as follows:

Y: The newsgroup exists and is not moderated.

N: The newsgroup exists but may not be posted to. Typically applies only to the control newsgroups.

M: The newsgroup exists but is moderated.

None: The newsgroup does not exist.

If the list of bad cross posts is longer than what will fit in the width of the text line, you can scroll it left and right. Place the mouse cursor in the **Bad Cross Posts** line and then press and hold down the middle mouse button (or the left and right buttons simultaneously if you do not have a middle mouse button) and move the mouse left and right. This should move the text left and right also.

c. Will Be Approved As/Approved As/Was Rejected

The title of this section depends on the current processing state of the message. If the message is currently in review or limbo, the title will be **Will Be Approved As**. If the message’s current state is approved, then the title will be **Approved As**. And if the message’s current state is rejected, the title will be **Was Rejected** and the headers and body will display as blank.

Only the headers shown (Subject, From, Reply-To, Organization, References, Newsgroups, and Followup-to) will be copied from the incoming post to the outgoing message. If no Newsgroups header appears (because the message was e-mailed), the current newsgroup name is automatically copied to that header. At this time PyModerator does not copy any "X-*" style extension headers to outgoing messages. You may modify any of these headers before approving a post, but in general you should keep such edits to a minimum to avoid confrontations with submitters. Generally only Newsgroups and Followup-to are likely to need editing.

The body of the message may be edited before approval, but you should of course limit such edits to those few things that are considered acceptable for your newsgroup.

You may use copy and paste in the message body window as well as highlighting sections and deleting them with backspace. Standard cursor movements with mouse clicks and arrow keys are possible, and auto word wrap is also operative.

Warnings: if you take too much time editing a message, you risk timing out and losing your work. If you click on another message in the message selection lists (or even the same message you are editing) you will also lose your work!

12. The Right Column: The Selected Message's History

The right column shows the major events that have happened to a message, starting from when it arrived in the review list and was assigned a moderator.

a. Event History

This is a scrollable list of events that have happened to the message. You click on the event row you want more information on and any additional details will appear in the **Event Detail** window. The format of each row is:

`MM/DD hh:mm mmmmmmmm eee…`

Where `MM/DD hh:mm` is the month, day, hour, and minute that the event occurred. The `mmmmmmmm` is the ID of the moderator who caused that event (or to whom the event was assigned by the system). The `eee…` is the event type. These are the following event types:

Assigned

Note

Email

Reject: *Rejection ID*

Approved

Limbo

Reassign

Scheduled Post

Scheduled Email

b. Event Detail

This displays event detail, if any, of the selected event row in the **Event History** window.

Security Information

Communication between the PyModerator client and server is unencrypted. Login passwords and POP e-mail access passwords will travel across the wire as clear text and could be intercepted. The POP e-mail user ID and password are stored unencrypted in the client's and server's data files since these must eventually be extracted as clear text. However, login passwords are stored encrypted via a one-way hash. The current version of the client does not store login passwords and forces the moderator to enter it at least once each time the client is started. Commands and command arguments are sent as pickled data using the Python cpickle module. Execution of arbitrary Python strings has been explicitly avoided in the data transfer and the author of the cpickle module believes the pickling and un-pickling of data streams should be secure against malicious attack. Commands are dispatched via a lookup table, not via the eval or equivalent Python modules that could allow execution of malicious code. However, as a standard precaution it is strongly suggested that PyModerator *not* be run as a root owned process on Unix. In general it is also advised to avoid unfettered access to machines on which clients are run.

Bugs, Deficiencies, and Future Direction

A number of bug fixes and requested enhancements are on my to-do list, listed below. I had planned to indicate their probabilities of being fixed along with their priorities, but on further reflection I have decided that I would be over committing myself. I've been working on this project at a glacial pace since my first burst of activity and unfortunately I don't see that changing any time soon. They'll happen when they happen. Contributions of code changes or fixes are always welcome (if you can make head or tail of my coding) but it is probably a good idea to check with me first before committing too much time to some enhancement or involved fix.

1. If you are in the middle of adding moderator comments to a message or have edited the headers and then select another message, you lose all your work on the message. One way to avoid that problem would be to create a new message view window when a list item is double-clicked. That way each message view window would maintain state information independently. However, client side locking would need to be changed since it assumes only one message is locked at any given time. Or maybe edits saved on server side??
2. Option whereby an e-mail is sent to a poster when a moderator has added moderator comments. (Or subsume this into standard action processing as suggested in the next to-do item.)
3. The number of available actions that are available to rejections is too limited. Allow for more permutations. E.g.: any combination of the following: Radio button for one of: E-mail Always, E-Mail Edits, E-Mail Never; Radio button to

move to one of: Reject, Limbo, Approve, Review. This would no longer be editing rejection types but adding standardized action buttons. A generalization to arbitrary actions into which even the current Approve button may also be subsumed.

4. When clicking on "EMail Poster" it deletes the moderator's edits to the post.
5. After processing a message, keep ordinal of current row and selection window the same.
6. Busy cursor doesn't work during long processes.
7. Alt-keys aren't working for menus.
8. Option to indicate how long one will be on vacation would be helpful.
9. When people log in, tell them how long it has been since their last log in.
10. When time logged out exceeds 24 hours, switch from "hours logged out" to "days logged out".
11. Some of the windows, like "Reassign Message" and "Edit Moderator" don't go away the way you'd expect when you'd selected an item from them.
12. Automatic logout of an old login when logging in after a network problem.
13. Currently the old login takes precedence.
14. UTF-8 strings make Unicode aware versions of Python puke. Must change octet string to Unicode string, which can't be sent via sockets.
15. Archive search on server should check for I/O after each file scan to allow other connections to have activity.
16. Vacation button should be easier to get at.
17. Clean up message view. Just show headers that were or could have been approved and the body as it could have been approved or was approved. Add button to show original message and a button to show differences, if applicable. Put quoted % and bad cross posts messages into a read only scrollable text "info" window.
18. Resizing of window areas. Should be persistent, or at least have an option to save a configuration.
19. Have option to allow X-* headers to be included in approved headers. People are using X-Face and other fancy things and want these to be passed through.
20. Have option to allow Message-ID from incoming messages to be used for the approved messages or allow PyModerator generated IDs or no ID (i.e. like now, where the NNTP server assigns the message ID if it doesn't see one).
21. Fix bug where MS Windows can't select text. This is turning out to be harder than expected. Works fine in X-Windows. Broken widgets in MS.
22. Make sure update of valid newsgroups does proper NNTP login! The server the newsgroup list is extracted from should mimic how postings work; e.g. if "Post

from Server" is No, it is the client side that must get the list of valid newsgroups and send it on to the server!

23. Have option to add a standard set of moderator specified headers to all approved messages, with the option of editing before approval.
24. Ability to pull messages from MAPI and Unix mail as well as POP servers would be nice.
25. White lists and black lists have been requested. Filter on what? Hate to put a lot of effort into this since they would be easy to fool.
26. Open to max window size on startup.
27. Saving of password on client side, including option for automatic login on startup of client.
28. Ability to locate and view messages in the same thread for reference.
29. Progress bar for archive search would be nice.
30. Complete async operation would be nice, such that GUI events aren't blocked while waiting for a response from server.
31. The code is a bloody mess! It really needs to be re-organized and it suffers from some bad design decisions. Someday it should be cleaned up.
32. Run time changing of fonts would be nice, but not critical.
33. "Automatic" de-MIMEifier would be nice.
34. Automatic detection of duplicate messages (within say, the last 10 or so messages) would be nice.
35. Would be nice if one could designate one's favorite editor or e-mail program to be launched when editing a user's submission or a followup or e-mail response.
36. E-mail alert notifications indicating that there are messages waiting to be reviewed would be nice.
37. Would be nice to search long messages using regular expressions. (Not to be confused with archive searching; this would highlight the text that matches the regular expression.)
38. Column headers on the Review, Limbo, Archive, and Event lists would be very helpful!
39. There needs to be some way for new moderators to test their setups; insertion of test messages when no POP account ready and posting to misc.test for test posts.
40. Usenet cancel capability would be nice. However, since 3rd party cancels would be the ones most useful (e.g. forged approvals) and there may be problems with the way some servers honor cancels, and there being other software that can do this, it was not attempted. In the long run it might prove handy.
41. Posting a follow-up from PyModerator would be nice.

42. There really should be a common Admin e-mail reply-to address for e-mails sent to posters. That way replies could be attached to messages for all moderators to see. Not sure the best way to implement that though.