Mercury/32 Help Index

Mercury is a Mail Transport System - a program that deals with the low-level mechanical process of moving electronic mail from place to place. While primarily intended for use with Internet Electronic Mail conforming to the RFC821 and 822 standards, it is nonetheless a general-purpose system capable of a wide range of unattended functions.

Key topics covered in this help file:

An overview of the Mercury System, and Mercury features
Your license to use the Mercury/32 software
What's new in this version of Mercury/32

Configuring the Mercury core module
Configuring mailing lists and Using mailing lists
Configuring the Mercury Mail Server (Maiser)
Template files
Aliases, synonyms and autoresponders
Setting up automatic replies
Automated mail filtering, and Server-wide policies
Content control ("spam" filtering)
Sending messages from within Mercury
Locking the console

MercuryS, SMTP Server module
MercuryC, SMTP Client module
MercuryE, Full SMTP Delivery Client
MercuryP, POP Server module
MercuryD, POP3 Client module
MercuryX, Connection Scheduler module
MercuryH, PH Directory Server
MercuryI, IMAP4rev1 Server module
MercuryB, HTTP (Web) Server module

Support for SSL/TLS secure connections
Support for Novell NetWare Local Area Networks
Daemons (third-party extensions)

Technical support
Obtaining Mercury Manual sets

Mercury - an overview

See also: The history of Mercury

What is Mercury, and why do I need it?

Mercury is a *Mail Server* - that is, a program that provides centralized mail services to a collection of related users' mailboxes on your network or computer. The program consists of a core application, which handles delivery of mail to local mailboxes, and a number of special plugin modules called *protocol modules*, each of which implements a major Internet protocol related to electronic mail. You can "pick and mix" protocol modules to produce a combination that exactly matches your mail system requirements.

The key concept in understanding what Mercury is and why you might need it is contained in the word *centralized*. Mercury is intended to be the hub for all your mail access, and as such it is most useful if you have more than one user needing access to Internet e-mail.

Using Mercury as a mail server can confer the following benefits (among others):

* Increases the security of mailboxes on your file server, because your users do not need rights to access other users' mail.
* Makes management easier by centralizing it - you can administer your mail users and their settings from a single point of access.
* Allows multiple different mail clients to access the same mailbox data
* Mail can be received even if your users' machines are turned off.
* Allows specialized features like automatic replies, automatic forwarding and mailing list distribution without needing the users' machines to be turned on.
* Gives you control over content - Mercury offers extensive filtering and logging, so you can get rid of unwanted "spam" before it gets delivered to your users' mailboxes, or can watch for unauthorized transmissions from inside your organization.
* Acts as an efficient way of sending mail internally even if the recipient is away or has his or her computer turned off.
* Allows efficient use of expensive connection services like ISDN or ADSL - only the computer actually running Mercury need have access to the outside world.
* Frees you from having to have your ISP provide mailboxes for you.

Mercury is an ideal companion for our mail client program, *Pegasus Mail* (available from http://www.pmail.com) - the two programs have special integration code that allows them to work closely together. Having said that, you can use Mercury as a mail server with any applications that support standard Internet mail protocols.

What does all this cost?

Nothing - nada, zippo, rien, nix... Mercury has been provided free of charge as a service since 1993. As such, it is an anachronism, dating from a time when the Internet was different (and in the opinion of the author, much more human than it is today). We *can* sell you technical support and manuals, but these are strictly optional - you do not need to purchase anything, and we have tried to provide you with all the information you need to run the program successfully within this package. If this all sounds too wierd, then perhaps you might like to read the history section to see how it all came about, and why it carries on so wrong-mindedly by the standards of these mercenary times.

How it all works:

Mercury is divided into two principal sections:

The Mercury Core Module This part of Mercury is contained in MERCURY.EXE itself, and is responsible for distinguishing between local mail and foreign mail, for ensuring that messages are either delivered correctly if local, or routed to the proper protocol module if foreign, and for providing the core functionality of the system, such as the mail server, distribution list management, automatic replies and autoforwarding.

Mercury Protocol Modules Protocol modules are DLLs loaded by Mercury that provide services to the core module. Protocol modules are generally tied to a specific delivery protocol and are usually used to ferry mail in and out of the Mercury system. Eleven protocol modules are supplied with the Mercury System:

MERCURYS.DLL	An SMTP Server module for handling incoming SMTP mail
MERCURYC.DLL	An SMTP Client module, for sending outgoing SMTP mail
MERCURYE.DLL	A full SMTP delivery client module for outgoing SMTP mail.
MERCURYP.DLL	A POP3 server, to allow POP3 clients to retrieve mail from the Mercury system.
MERCURYF.DLL	A simple Finger server
MERCURYD.DLL	A POP3 client capable of retrieving mail automatically on behalf of users on the local system.
MERCURYX.DLL	A scheduling module, allowing co-ordinated startup and shutdown of the various Mercury modules. Invaluable for use in dial-up environments.
MERCURYH.DLL	A PH Query Server for Directory Services.
MERCURYW.DLL	A server that allows users to change their passwords.
MERCURYI.DLL	An IMAP4rev1 server allowing users to access their entire mailbox remotely.
MERCURYB.DLL	An HTTP server, allowing web-based services such as mailing list subscription management.

MercuryE and MercuryC both provide the same functionality but in different ways: MercuryE delivers directly to the recipient's mail server, while MercuryC uses a technique called Relaying, by asking another single computer to send all mail on its behalf. MercuryC is ideally suited for use behind firewalls and on dialup links where the shortest connection time is required.

What's new in this version of Mercury/32?

Mercury/32 v4.01b (December 2004)

V4.01b includes fixes for some buffer overflow vulnerabilities in the MercuryI IMAP server. It also closes a number of memory and handle leaks in the IMAP server, and should be generally more robust than earlier versions. As well as reliability fixes, you can now specify the maximum number of delivery threads MercuryE should use, and can perform transaction-level filtering on the MAIL FROM command using MercuryS's filters. Finally, Mercury Policies (external tasks) can now modify the contents of jobs as they pass through the queue.

All sites using Mercury/32 v4.01a are urged to upgrade to v4.01b.

Mercury/32 v4.0 (November 2003)

V4.0, while a long time coming, includes some of the most powerful improvements we have ever made to Mercury, and creates a significant roadmap for future versions. In particular, the MercuryB web server protocol module heralds the arrival of web-based management and services for Mercury, and the inclusion of SSL support improves security dramatically.

* *MercuryB HTTP* server: this new protocol module implements the HTTP protocol for web-based access using a loadable service module approach to provide services. MercuryB is not intended as a general-purpose web server - it is specifically targeted at tasks specific to Mercury, such as mailing list subscription management (included in this release), remote administration and web mail (in development).

* *SSL/TLS support* The MercuryS (SMTP server), MercuryP (POP3 server) and MercuryI (IMAP4 server) protocol modules now have comprehensive, easy-to-enable support for the SSL secure sockets protocol.

* *Web-based mailing list subscription management* Using the new MercuryB HTTP server, mailing list subscribers can now manage their list subscriptions over the web. Subscribing, unsubscribing and managing subscriptions is now as easy as using a web browser and filling in some simple forms.

* *Heavily improved spam filtering* The content control engine in Mercury/32 v4.0 has been heavily enhanced with new tests and a new default rule set that catches vastly more spam. We are also currently developing auto-update mechanisms that will allow spam rule sets to be updated automatically, allowing faster response to new types of spam as it arises. The content control engine now also has more options for adding headers to messages and can generate diagnostic headers indicating which rules contributed to the calculated weight for the message. Mercury's autoreply engine has also been changed so that any message getting a positive content control weight will not receive an autoreply: this prevents the server from sending autoreplies in the majority of cases.

* *Improved HTML and encoding handling in Content Control* Mercury's content control engine has been retrofitted with improvements made in Pegasus Mail v4.12, and now only checks textual components of messages. When doing so, it now correctly applies BASE64 and Quoted-printable decoding before applying its tests, and strips HTML tags from HTML content (leaving only A, IMG and base tags for later testing). Content control now has explicit tests that check for lazy HTML (IMG tags with remote references), IFRAME tags, and

excessive numbers of comments. The regular expression engine has also been extended with a number of powerful new tests.

* *Attachment filtering* Mercury's general-purpose filtering rule engine has been enhanced with the ability to filter attachments within messages. Attachment filters can work on the filename or extension of the attachment, and can delete attachments from messages if required. The filtering engine now also has an action that can add a header of your choosing to any message as it is processed.

* *VERP support in mailing lists* VERP (variable envelope return processing) allows near-total automation of error processing in mailing lists. Mercury/32 now includes three modes for handling mailing list errors, two of which use VERP. For larger lists, VERP can substantially reduce the amount of effort required to keep subscriber lists up-to-date.

* *Subscription passwords* You can now specify that a password be required for subscription to a mailing list - this eliminates casual or unwanted subscriptions to mailing lists and reduces moderator workloads.

* *MercuryI IMAP Server improvements* The IMAP4 server has been heavily overhauled and is now much more robust than in previous versions. Sites using stateless clients (such as webmail packages like Twig or IMP) should notice performance improvements as well.

* *Transaction-level filtering in MercuryS* The MercuryS SMTP server now supports filtering at the transaction level - you can apply expressions to the SMTP EHLO command and to the data of the message as it is actually received. This is an incredibly powerful feature, particularly for dealing with address harvesters and spammers who attempt to relay via your server.

* *Short-term blacklisting* MercuryS now supports the idea of short-term blacklisting, where clients that breach too many compliance conditions or match specific transaction-level filters can be blocked from connecting to the server for a period of 30 minutes. This is a powerful way of dealing with zombie systems (computers that have been taken over by hackers or spammers) and of protecting against denial-of-service attacks.

* *Connection control overhauled* The ability of the Mercury server modules to control which machines can connect to them based on IP address has been totally overhauled. It is now possible to specify arbitrary address ranges to which restrictions can be applied, and each restriction can have specific attributes enabled or disabled (so, for instance, you could create an "Allow" entry that permitted a system to relay mail through MercuryS but which was not exempt from transaction-level filtering).

* *Improved console output* The various Mercury server protocol modules now produce much more detailed information on the console and in the log files to show why particular actions occurred.

* *Progressive backoff in delivery* You can now tell Mercury to use a "progressive backoff" algorithm when calculating retries for mail jobs; this tells Mercury to start with a relatively short retry period, then use gradually longer and longer retry periods the more retries occur. This can dramatically speed the delivery of mail when one-off transient glitches occur, and cuts down queue congestion caused by messages with more significant delivery

problems.

Mercury/32 v3.32 (August 2002)

Mercury/32 v3.32 is a moderate update with many fixes and corrections, and a short list of useful new capabilities. Sites using the MercuryI IMAP server should regard v3.32 as a *mandatory* update - the IMAP server has had a significant number of fixes and performance improvements made to it.

In terms of new capabilities...

* *New Compliance page* in the MercuryS SMTP server allows you to reject certain types of unwanted mail before they even make it into your mail queue. Combined with content control and a small amount of filtering, these options can practically eliminate the occurrence of "spam" in your mailbox.

* *Transcripts:* the MercuryE SMTP client can now generate delivery transcripts, providing reasonable proof that a message has been successfully delivered. To use transcripts, you must first enable them in the MercuryE configuration dialog, then add an *X-Transcript-To: <address>* header to the message you are sending (you can do this easily in Pegasus Mail on the Special page of the message editor). MercuryE will return the transcript to the address you specify.

* *New mail server commands* allow you to add addresses to the MercuryS killfile and to force Mercury to reload its user database after a manual update. These new commands are password-protected, using passwords specified in the *Mail Server* configuration dialog.

Mercury/32 v3.31 (April 2002)

Mercury/32 v3.31 is a major release, with a number of significant new capabilities.

* *IMAP:* MercuryI, the Mercury/32 IMAP server module, is now a standard component of the Mercury/32 release. Long in development, MercuryI allows IMAP4rev1-compatible mail clients such as Pegasus Mail, Mulberry and Outlook to access entire mailbox structures in a managed way. Click here to go to the help section on the IMAP server.

* *Policies:* This powerful new feature allows you to create your own external tasks to examine mail messages. Mercury provides support for things like attachment unpacking - all you do is put together a script, batch file or program that does the processing you need, then tell Mercury how to invoke it. As an example, it takes about 10 minutes to create a virus scanning policy to check all your mail for viruses, even if the virus scanner does not understand Internet message encodings. Policies are found on their own page in the *Mercury Core Module* configuration dialog; click here to view the help section on policies.

* *Content control:* Are you sick to death of spam? So are we. Mercury now has one of the most comprehensive content examination and control features we could put together. Using the new content control option, you can apply tests of aribtrary complexity to incoming mail and take any of a number of actions based on the results. A clever weighting system allows you to aggregate a number of points with varying levels of importance during the evaluation. Click here to view the help section on the content control feature.

* *Blacklist overhaul:* Mercury now allows you to create a practically unlimited

number of blacklist definitions (for services such as the RBL, or ORDB), and the functionality available for testing those blacklists has been thoroughly updated. Click here to view the help section on using blacklist services.

* *NetWare support* A new NDS-mode enabler has been included with this version: the new enabler allows you to specify an object that Mercury should try to look up in the NDS database before starting any mail transaction. This approach, while a little rough and ready, allows Mercury to work reliably in an environment where servers are occasionally unavailable.

* *Programmable autoresponders/autoreplies* Mercury now allows you to set up multiple automatic replies and have the correct one chosen based on factors like the day of the week, the time of day or the current date. Existing automatic replies will, of course, continue to work correctly - the old and the new can co-exist. Click here to see the help section on automatic replies and autoresponders.

* *Numerous bug fixes* Many, many small problems have been addressed, especially in the MercuryE end-to-end SMTP delivery module.

Locking the Mercury Console

In some environments, you may not want Mercury's configuration dialogs to be accessed by unauthorised users while the program is running. You can offer yourself some protection from this kind of tampering by selecting *Lock console* from the *File* menu. Mercury will prompt you to enter a password twice, and when you have done so and pressed *Enter*, will no longer allow any access to its menus or permit itself to be terminated without the password. Once the correct password has been entered, the console is unlocked and operation continues normally. While the console is locked, the program may be minimized and restored normally, allowing the console information to be viewed by anyone.

Note: Mercury cannot prevent itself from being terminated by the Windows Task Manager (accessed by pressing Ctrl+Alt+Del). In environments where this is an issue, you should consider using the Microsoft Policy Management software to disable the Task Manager's "Terminate Application" option.

Creating a startup console password: If you check the control labelled *Save as startup console password,* then Mercury will save your console password in a special encrypted format and will use it automatically at startup. This allows you to start the program locked.

Configuration

Mercury is a large, rich system and offers a wide range of configuration options, accessible via the *Configuration* menu.

Mercury Core Module
Mailing lists
Template files
The Mail Server
Mail filtering rules

You can also configure individual Protocol Modules from the Configuration menu - each Protocol Module will have its own online help you can consult after you have opened its configuration dialog.

The Mercury Core Module Configuration

To configure the basic operation of the Mercury core processing engine, choose *Mercury core module* from the *Configuration* menu.

A tabbed dialog will appear offering several pages you can select. The items on the front page of the dialog, *General*, are described below. The other pages are:

Local domains	Tells Mercury how to identify local addresses
Groups	Tells Mercury about groups on your Network
Files	Tells Mercury where to find or put its files
Reporting	Controls statistics and system messages
Advanced	Advanced Mercury settings
Policy	Server-wide message handling policies

Options on the *General* tab

Internet name for this system Enter here the Internet name for the machine on which Mercury is running. Mercury will use this information when forming certain addresses, such as the postmaster address. The name you enter here should be a fully-qualified domain name; if you are intending to use Mercury to provide mail services outside your immediate organization, the name you provide will need to be accessible in your Domain Name Server (DNS) system.

Mail queue directory, SMTP queue directory These entries control where Mercury should look for and place mail that is to be processed. The mail queue is where mail clients such as Pegasus Mail put messages for Mercury to process; Mercury also places jobs here on occasions, usually when generating autoreplies and mailing list mail. The SMTP queue is the location where the Mercury Core Module should place messages intended to be sent to the outside world by the MercuryC or MercuryE Client module. The mail queue and SMTP queues can be, and normally are the same.

Local mailbox directory path (This entry is ignored if you are using a network support module). Tells Mercury how to locate your users' mailbox directories. The string is a standard pathname containing one of two special placeholders - either *~8* or *~N*. When Mercury uses the string to find the mailbox for a user, it replaces *~8* with the first eight characters of the user's name, or replaces *~N* with the user's whole username. If you are using Pegasus Mail v3.01d or earlier, or any 16-bit Pegasus Mail client as your mail client in conjunction with Mercury, you should not use the *~N* substitution - you should only use the *~8* version. If this 8-character restriction creates problems with usernames for you, you could consider defining synonyms for the names that are longer than 8 characters, or upgrading to a later version of Pegasus Mail.

NOTE: It is currently a restriction of Mercury that the ~8 or ~N placeholder must appear at the end of the path - so, *C:\PMAIL\~8* is legal, but *C:\PMAIL\~8\MAILBOX* is not.

Time zone Enter here the timezone for your site, expressed as a plus or minus difference from GMT. So, if you are in Los Angeles and are currently at GMT - 9 hours, you would enter -0900 in this field. Mercury will accept the so-called "vernacular" time zone format, such as PST and CST, but the use of these formats is no longer recommended on the Internet and we strongly advise you to avoid them, since their use makes it impossible for most mail programs to sort properly by date.

Poll mail queue every x seconds This setting controls how often the core module should check to see if there is mail waiting to be processed in the queue. We recommend that you do not set it below ten seconds for performance reasons.

Username of postmaster Every system capable of receiving Internet mail must have a user called *postmaster*, to whom problem and status reports are sent. The postmaster account is usually an alias to a real user on your system, and this is the expectation within Mercury. Enter in this field the username of the user on the machine where Mercury is running who is to act as your postmaster. While it is permissible to have a non-local address as your postmaster address, we strongly recommend you do not do this, since it can create real problems and mail loops when the remote machine is unreachable. This setting is mandatory - Mercury cannot run properly without it.

For delivery failures return x lines of the message When Mercury cannot deliver a message to a local user for whatever reason, it will invoke a template file you provide for delivery failures. One of the optional replacements that can be used in the delivery failure template file is a special substitution that sends a certain number of lines from the failed message. This configuration option controls how many lines of the message are returned when the special partial return substitution is encountered.

Broadcast notifications for normal mail Mercury has special Network awareness modules that allow it to take advantage of certain specific features of some local area networks. One of the features that some networks (such as Novell NetWare) support is the transmission of a single-line broadcast message that appears on the target user's screen. If this control is checked and you are running Mercury on a network that supports broadcast messages, Mercury will send a short message to users when new mail arrives for them.

Broadcast notifications for receipts (See the preceding section for more detail) This control determines whether Mercury should send broadcast messages advising the arrival of mail messages that confirm reading or delivery.

Send copies of all errors to the postmaster If this control is checked, Mercury will send a copy of all error reports it generates to the local postmaster as well as to the original sender of the message. This allows the postmaster the option of correcting addressing errors and other simple problems.

Change file ownership to recipient As with broadcast notifications, some Network systems support the idea of file ownership, usually to calculate disk space usage. If your network supports this idea and this control is checked, then Mercury will attempt to change the ownership of all the messages it delivers so that the actual recipient owns the file.

Suppress validation of From field when processing mail Mercury usually attempts to validate that the "From" field of all mail it delivers is legal. This can sometimes cause problems if you receive mail from sites that use broken or faulty mail programs; if this is the case, you can suppress the validity check Mercury performs by checking this control.

Hard to quit (exit only on Ctrl+File|Exit) When this option is checked, Mercury will ignore all attempts to quit from it, and will minimize itself to the system tray instead. In order to quit from the program, choose "Exit" from the "File" menu while holding down the Ctrl key. This option is useful when Mercury is run on a server to prevent people from accidentally closing it down.

Configuring local domains

See also: Core module configuration

Domains recognized as local by this server This is probably the single most critical area of configuration in the Mercury system -- if you get this section wrong, you will inevitably get mail loops and other problems. In this section, you must tell Mercury all the Internet names it should regard as "local" -- that is, for which it should attempt direct delivery on the local system rather than forwarding the mail to another machine for processing.

The *host/server* section of each definition is intended to allow Mercury to deliver mail to multiple file servers in supported network environments: if you are running Mercury on a single system or serving Pegasus Mail in either networked or multi-user standalone mode, the host/server entry is ignored. In the NetWare Bindery mode environment, this part is used to tell Mercury that a particular domain represents addresses on a specific file server or tree. In the NetWare NDS mode environment, this part is used to tell Mercury that a particular domain represents addresses within a specific segment of your NDS tree (see below for more detail).

When entering domains into this section, you should usually provide three entries per local Internet domain - a fully-qualified version, a simple version, and a special entry called a domain literal version, which is the IP number of your system enclosed in square brackets. For example, if your system's Internet name was calliope.pmail.gen.nz (192.156.225.76), you might create these domains definitions:

calliope	calliope
calliope	calliope.pmail.gen.nz
calliope	[192.156.225.76]

Domain mailboxes Mercury supports the idea of a domain mailbox, or a mailbox that accepts mail addressed to any user at a given domain. To create a domain mailbox, first create the user account that is to receive all mail addressed to the domain, then place an entry in the *Domains recognized as local by this server section* in the following format:

```
DM=username          domain address
```

username can be any valid reference to a single local user on your system. So, to create a domain mailbox where user *mailserver* receives all mail addressed to any user in the domain *fish.net*, you would create this entry:

```
DM=mailserver        fish.net
```

With this entry in place, mail sent to *[any address]@fish.net* will be delivered into user *mailserver*'s mailbox.

NDS Mode In NetWare NDS mode, the domains section can be used to tie a domain to a specific portion of your tree. So, if you have all mail sent to the domain `myorg.com` to a context in your NDS tree called `sales.us.myorg`, you would use this entry:

```
[Domains]
sales.us.myorg : myorg.com
```

When specifying an NDS domain, you can apply the definition to an entire portion of a tree (including all sub-levels within the NDS tree) by prefixing the context name with the special character **/** - so, in the example above, if you simply wanted to equate your entire NDS tree with the domain `myorg.com`, you would use this entry:

```
[Domains]
/[root] : myorg.com
```

Domain Name Service The process by which systems connected via the Internet find out how to contact each other. Each organization on the Internet advertises the names of its machines via a primary Domain Name Server (usually a unix system).

Mailing Lists - Overview

Mercury has strong support for *mailing lists* -- groups of addresses that can be associated with a single address on your system. When a mail message is sent to the address associated with the list, Mercury will send it on to everyone who has subscribed to the list. Mercury's mailing lists are created and managed using the *Mailing lists* option on the *Configuration* menu.

See also: General list settings
List access settings
List distribution settings
Handling errors in list mail
List membership settings
Using mailing lists - commands and operation

Mailing lists have three key elements: *Membership, Moderators* and *Settings*.

Membership: The membership of a mailing list is the group of people who are subscribed to it at any given time. Mercury's mail server allows people to subscribe and unsubscribe automatically by sending it messages containing subscription commands. List members also have a certain amount of control over the way they receive mail -- they can choose to enable and disable receipt of mail from the list, and if you have enabled digest support for a list, they can choose whether or not they want to receive their mail in digest format.

Moderators: A mailing list can have one or more *moderators*, who are effectively managers for the list. Moderators have full control over the membership and settings of a list, and you can also configure a list so that only moderators may actually send mail to its membership: when you configure a list this way, then the list is said to be *moderated* - that is, only specific people can send mail to it. The intention of a moderated mailing list is that mail must be submitted to the moderator, who will then decided if it should be distributed. Note that a list can have moderators without being a moderated list - that is, a list can have supervisors, but can still distribute mail sent from the general public. A list need not have any moderators if you wish, and it is permissible for a moderator not to be a member of the list.

Settings: Mercury offers a wide range of settings that can be applied to a mailing list, which control the way it behaves when it receives mail for distribution, and the way it responds to control requests, such as subscription messages. You can configure Mercury to manage your mailing lists using its extensive configuration dialogs - follow the links below for more information on each area of configuration:

General list settings
List access settings
List distribution settings
List membership settings

Digests: a digest is a single mail message that contains a group of other mail messages. For busy mailing lists, it is often convenient to allow mail sent to the list to accumulate in a digest and be sent out periodically, instead of sending the messages out immediately.

Mercury supports a special digest format called the MIME digest format. MIME is an Internet standard for the composition and presentation of messages and is widely supported. If your subscribers use a mail program that supports MIME digests, such as Pegasus Mail, then digests generated by Mercury will look like a kind of mail folder to them and they will be able to browse the messages in the digest on an individual basis.

Using mailing lists - commands and methods

See also: Other mail server commands

OK, you've created your mailing list... Now what do you do with it?

In general, the answer to this question depends on whether the list you have created is *moderated* or *unmoderated*. For moderated lists, only users marked as list moderators may send messages to the list: other users, even members, cannot send mail directly to the list. Moderated lists are useful for low-volume announcement lists, or in cases where the subject matter sent to the list needs to be scrutinized before posting.

In the normal case, however, the list will be unmoderated, which means that your users can manage their own subscriptions to it by sending commands to the Mercury Mail Server via e-mail.

Commands affecting list membership or operation should be sent to the mail server address: by default, this will be the reserved address **`maiser`** at your site. Maiser is not a username - it is a kind of alias handled in a special way by Mercury itself. Sending a message to maiser tells Mercury that the message body contains commands that it needs to process, rather than mail that needs to be delivered. Multiple commands can be included in a single message, one line per complete command, and command processing terminates as soon as Mercury encounters a blank line or an `EXIT` command. The user who sends the message will receive a short message back indicating the success or failure of the commands he has issued.

For mailing list management, the following commands are recognized by the mail server:

Commands available to everyone

SUBSCRIBE <list-name> [Full name] (also *SUB <list-name> [Full name]*)	Add the sender's address to the list
UNSUBSCRIBE <list-name> (also *UNSUB <list-name>*, or *SIGNOFF <list-name>)*	Remove the sender's address from the list
ENUMERATE <list-name> (also *REVIEW <list-name>)*	Return the list membership
LIST	Returns the lists available at this host
SET <list-name> DIGEST	Set your list subscription to digest mode
SET <list-name> NODIGEST	Turn off digest mode for a list.
SET <list-name> MAIL	Turn on delivery from a list
SET <list-name> NOMAIL	Turn off delivery from a list
SET <list-name> VACATION X	Temporarily turn off delivery from a list for X days
SET <list-name> REPRO	Receive copies of your own postings to the list
SET <list-name> NOREPRO	No copies of your own postings to the list.
STATUS <list-name>	Get current subscription information for a list

Commands only available to list moderators:

ADD <list-name> <address> [Full name]	Add a user to a list
REMOVE <list-name> <address>	Remove a user from a list
MSET <user> <list> <option> - *option* can be MAIL, NOMAIL, DIGEST, NODIGEST VACATION, REPRO or NOREPRO	Change a user's subscription options
MSTATUS <list-name> <user>	Get a user's subscription status

PASSWORD <password>	Supply the password for moderator commands

For more information on passwords and lists, click here.

Other mail server commands

Other mail server commands

As well as providing automated control of mailing lists, Mercury's mail server also recognizes the following commands; to use these commands, send a message to the Mail Server account (usually called "maiser") with the message body containing the commands you want it to execute, one per line. The Mail Server will process commands until it encounters a blank line or an EXIT command.

HELP Returns the helpfile defined in the MAISER section of MERCURY.INI to the sender.

BOUNCE Returns the message to the sender, headers intact.

LOOKUP <string> Searches the local system for user names matching <string>, which can contain '*' and '?' wildcards. If you wish to disable lookup, make sure that there is no LOOKUPFILE entry in the MAISER section of your MERCURY.INI.

VERIFY <address> Returns a message indicating whether the address specified is valid on the local host.

INDEX Sends the file INDEX.TXT from your "files to send" directory (identical to the command *SEND INDEX.TXT*).

SEND <filename> Sends the named text file from your "files to send" directory. If you do NOT want this facility to be available on your system, make sure that there is no SEND_DIR entry in the MAISER section of MERCURY.INI. The send command is VERY secure - you cannot specify paths of any kind in <filename>, and Mercury will ONLY look in the directory specified in SEND_DIR.

FINGER <address> Returns information about the address supplied. Also returns the contents of the file *PROFILE* if it exists in the specified user's new mail directory (useful for PGP keys and the like).

EXIT Stop processing commands from the mail message.

Configuring Template Files

Template files are files used by Mercury to generate messages automatically. In a template file, you can enter plain text, and also special substitution characters that Mercury will replace with system-specific information. Delivery failure notifications, confirmations of delivery and some of the mail server responses are formatted using template files.

A template file is a plain text file and can be created using any standard editor, for example the Windows NOTEPAD command. It must be formatted as a mail message - in fact, the first four lines of the message will usually look like this:

```
From: postmaster@~N (Mail System Administrator)
To: ~T
Subject: Confirmation of delivery
Date: ~D
```

The ~N, ~T and ~D characters are special substitutions, replaced with the system's domain name, the recipient's mail address and the date respectively. The rest of the message can take any form you wish and you can use any of the special substitutions as often as you need.

Mercury recognizes the following substitutions in template files:

~~	A single tilde character
~D	The date, in proper RFC822 format
~T	The recipient's mail address
~G	The first x lines of the message (*)
~M	The entire original message
~B	The entire original message body (no headers)
~R	The failure text, or results (for mail server searches)
~S	The subject field from the original message
~N	The current system's Internet domain name
~Y	A valid MIME Multipart boundary separator

Using Multipart MIME format in Template files: MIME is the dominant Internet standard for message formatting. One of the more powerful features of MIME is its ability to generate messages with multiple parts: in order to do this, you need to add some special headers to the message, and to separate the parts of the message from each other using a special boundary string. To generate a Multipart MIME message in a template file, add the following two lines to the headers of your template file, exactly as they are shown:

```
MIME-Version: 1.0
Content-type: Multipart/Mixed; boundary=~Y
```

Now, at the start of each part of your message, add the following lines

```
--~Y
Content-type: Text/Plain
```

Making sure that there is a single blank line between these lines and the start of the text of the message part.

For an example of how to generate Multipart MIME messages in Mercury, please see the sample delivery failure template file, FAILURE.MER, supplied with Mercury.

* The number of lines copied from the original message is controlled by the option settable in the *Mercury*

core module configuration dialog.

Configuring the Mail Server

See also: Mailing list commands and Other mail server commands

Mercury's Mail Server provides a number of automated services, including automatic mailing subscription and unsubscription, file transmission, user lookup and search facilities, and remailing files at specific times.

General mail server configuration:

Help file The file Mercury should send when it receives a "help" command, or when it receives a command it does not recognize.

Lookup results file The name of a template file that the mail server should use to return the results of user searches using the *Lookup* command. If this field is left blank, then the lookup command will be disabled.

Log file The name of a file in which the mail server should record all the commands it processes. If this field is left blank, the mail server will not perform any logging.

"Send" directory The directory in which the mail server should search for files requested using the *Send* command. Files in this directory must be text files, so if you want to make binary files available via the Send command, you will have to uuencode them yourself first. The mail server will only look in the directory you specify here and will not accept filenames containing paths; because of this, the option is an extremely safe way of distributing data to the public via e-mail. If this entry is blank, then the Send command will be disabled.

"Notify" queue directory Mercury's mail server supports two deferred mail commands - *Notify*, which sends a broadcast message to the sender at a given time (if broadcasts are supported on your network), and *Remail*, which sends a mail message at a particular time. For these commands to be available, Mercury requires a directory where it can create status files for each request: enter the path to that directory in this field (the directory must exist already - Mercury will not create it). If this field is blank, the "notify" and "remail" commands will be unavailable.

Disable the mail server "Lookup" command Check this control if you do not want the Lookup command to be available on your system.

Only accept "notify" commands from local users If this command is checked, then the mail server will only accept "notify" requests from users who are local to your system (that is, to whom it could actually deliver a message). If the control is unchecked, then anyone, no matter where they are located, may queue "notify" requests for users on your server.

Editing the mail server template files

The options to edit the mail server help file and to edit the lookup results template let you customise the responses generated by the mail server to certain commands. The help file is a plain text file - template substitutions cannot be used in it.

Configuring support for Groups

See also: Core module configuration

If your underlying Network system supports the notion of "Groups", or collections of users, then you can enable that support using this configuration dialog.

By default, Mercury does not make groups available for mailing purposes - this is partially a security issue and partially a configurability issue. In order to make a group on your system available to receive mail, you must add it here. Making a group available involves providing three pieces of information:

Public name The *public name* of a group is the e-mail address people will use to send mail to the group. You can give a group the same public name as its actual name on your system, but there may often be reasons why you might not want to do this - for instance, you might feel that the group "everyone" on your Novell NetWare server is less suitable than the name "staff", so you would define the group's public name to be "staff". People would then mail everyone on your server by sending a message to staff@server.domain. You will also need to use different public names for groups on different servers that have the same group name.

Group name The actual name of the group on your network. The group's public name may be different from this name.

Host name The server or host on which the group is based. In single-server environments you will not have to enter anything in this field, and the value entered here will vary depending on the underlying network: for instance, under Novell NetWare Bindery Mode, the host name will be the name of the Bindery Server that holds this group.

Example:

Your NetWare server's Internet name is "orange.com", and you have a group on it called SUPPORT, which you want people to be able to mail as "tech-support@orange.com".

In the Public name field enter	tech-support
In the Group name field enter	SUPPORT

Note that when defining groups you do NOT add the domain name.

Configuring aliases, synonyms and autoresponders

An *alias* is a specialised form of e-mail address that stands in for another e-mail address on your system. Aliases are often used to create addresses which do not vary, even though the person receiving the mail may change. For example, say you want to offer your users an e-mail address they can use to obtain help; it is clearly much better to use an address like help@mydomain.com than to give the address of a user on your system, since if that user leaves or is transferred, you can simply point the alias for "help" at the person's replacement and your users are not forced to change their habits.

A *synonym*is a specialised form of alias that effectively replaces a user's real e-mail address with an alternative form. Using synonyms, you can adopt consistent addressing schemes that differ from your users' actual login names. For more information on creating and using synonyms, click here.

Mercury has very powerful aliasing features: you can access them either from this dialog, or by using the commandline import/export tool MALIAS.EXE supplied with Mercury. An alias simply consists of two parts - the alias (or, the address people use to send mail) and the real world address (the address to which Mercury should deliver any messages it receives addressed to the alias). The real world address does not have to be a local address - it is perfectly valid to have an alias for an address on a remote system (this approach is often used to redirect mail to someone while they are absent, or if they leave the organization).

To create an alias, fill in the alias and the real world address, then click the *Add as new* button.

To change either the alias or the real world address of an existing alias, click on it in the list, then make the changes and press the *Change* button.

To remove an alias, click on it in the list then press the *Delete* button.

Exporting aliases: You can save your alias list to a simple text file in the format expected by the MALIAS commandline utility by clicking the *Export* button.

Autoresponders

An autoresponder is an e-mail address that simply returns a predefined message to whomever sends it mail. Autoresponders differ from automatic replies in that once the automatic response has been sent, no further attempt is made to deliver the incoming message.

In order to support autoresponders, Mercury supports two specialised aliases that work differently from other aliases - *FILE:* aliases and *TFILE:* aliases. A *FILE:* alias is an address that will return the contents of a text file to the sender when it receives any message, while a *TFILE:* alias returns a formatted message using a template file.

To create a *FILE:* or *TFILE:* alias, enter the alias as normal, but for the real world address enter either TFILE: or FILE: followed immediately by the path to the file you want to use.

Example:

```
        info   =   FILE:\\myserver\sys\system\mercury\info.txt
or      faqs   =   TFILE:r:\system\mercury\faq.mer
```

Note that it is very important that the file specified in a TFILE: alias is actually a template file: if you do not specify a valid template file, Mercury may crash when it tries to send the reply.

TFILE: and FILE: aliases are completely secure - they are only accepted if they actually appear in your alias file: a user cannot send a message to a TFILE: address to obtain files illegally from your system.

Configuring file locations and Queue settings

See also: Core module configuration
Creating secondary mail queues

This dialog tells Mercury where to find various files that it uses in regular operation. There should usually be little or no need to change these values. Note that when you are using Mercury on a Network, all paths should be entered in *UNC format* - like this: *\\SERVER\VOLUME\PATH*. It is permissible to use DOS paths as well, but you should not use non-standard paths, such as the Novell NetWare path format. In all cases except for the log file entry it is permissible for the file not to exist - Mercury will create it as required.

"List of lists" file The location and name of the file in which Pegasus Mail stores information about the mailing lists available on your system.

Scratch files directory A directory where Mercury can create temporary files. This directory must exist - Mercury will not create it for you.

Alias database file The location and name of the file in which your system aliases are stored.

Synonym database file Synonyms are a specialised form of alias used in conjunction with the Pegasus Mail system to provide alternative addressing formats on your system. If you have created synonyms on your system using the Pegasus Mail PMGRANT utility, you will need to use the CH_SYN.EXE utility (or NSYNONYM.EXE in NetWare NDS mode) supplied with Mercury to build a synonym database for Mercury, and enter the name and location of that file here. Note that the Pegasus Mail NDS-mode configuration utility, NCONFIG.EXE, can also be used to create synonyms for your users - this utility is a standard part of Pegasus Mail for Windows releases starting with v3.12. For more information on synonyms, click here.

Delivery confirmation template The name and location of a template file that Mercury should use when reporting confirmation of delivery.

Delivery failure template The name and location of a template file that Mercury should use when reporting delivery failures.

System log file The name and location of a file into which Mercury should store information about the jobs it processes. If this entry is left empty, Mercury will not perform any logging.
Special naming options for log files: Mercury allows you to use certain special characters in the names you give for log files: these special characters are replaced by Mercury each time it needs to open the log file, allowing you to "build" a filename based on the time and date. This is particularly useful for making log files "roll over" periodically. The special characters you can use in log file names are:

~Y	The year, expressed as 2 digits
~M	The month, expressed as 2 digits
~D	The day of the month (1..31), expressed as 2 digits
~W	The week of the year, starting from 0.

Example: to create a logfile for each week of the year, you might enter a path like this:

```
C:\MERCURY\LOGS\MERC~Y~W.LOG
```

Directory for noticeboards If you have created a noticeboard system within Pegasus Mail, Mercury can delivery mail to it. Enter the top directory in your noticeboard structure here (exactly the same as the NB environment variable you give to Pegasus Mail). Mail can be sent to any noticeboard using the address format *<boardname>%nb@host.domain* - so, for example, if you have a noticeboard called *comp.sys.mail*, you could mail it using the address *comp.sys.mail%nb@host.domain*. *Tip:* you can make the process of posting to noticeboards much easier by creating aliases for them - using an alias allows you to create a simple e-mail address for a noticeboard.

Queue Processing controls

These settings control how frequently Mercury/32 should retry messages that have temporary delivery problems, and the maximum number of times a job should be retried before Mercury should conclude that it cannot be delivered. You cannot set the retry period shorter than one minute, nor can you set fewer than two, or more than 99 retry attempts.

Use progressive backoff algorithm to calculate job retries When you check this control, Mercury will change the way it calculates retries on undeliverable mail. Normally, it simply adds whatever retry period you have defined to the current time, but in progressive mode, it will increase the period between retries the more of them there are. For every ten retries, the time between retries increases by progressively larger steps, to a maximum of seven times the retry period you have defined. Progressive retry mode works well when you use a retry period of fifteen minutes and a maximum of 99 retries (giving a maximum retry period of about four days).

Process the queue in two passes (file locking) If you check this control, Mercury will change the way it processes mail submitted by programs such as Pegasus Mail; instead of taking the submitted job immediately, it will wait until the job has the same non-zero size for two polling cycles in a row before processing it. This can be necessary in systems such as Windows Peer-to-Peer networking where file locking is not properly implemented; it ensures that the client has finished writing the mail message to the queue before Mercury tries to process it. Turning this option on is always safe, but will result in a slightly longer delay in mail being sent out.

Secondary mail queues

When Pegasus Mail and Mercury are used together, Pegasus Mail submits mail to be processed by Mercury as files in a queue directory. Mercury then takes these *submission files* and creates its own special queue format from them.

Usually, Mercury and Pegasus Mail will simply share a queue directory - this is completely safe and is the normal operating mode. In some cases, though, there may be advantages in separating the core Mercury queue from the submission directory used by Pegasus Mail. To do this, click the *Secondary queues* button, and tell Mercury the full path to the directory where it should look for the submission files created by Pegasus Mail. You can create as many secondary queues as you wish - the Mercury core module will poll them at the start of each mail polling cycle.

There are typically two reasons why you might want to create secondary queues:

1: *Improved reliability* If your file server or shared volume is less than completely reliable, then moving the Mercury core queue to a local drive can reduce the likelihood of problems during processing if the file server or shared volume becomes unavailable for some reason.

2: *Novell NetWare NDS mode* When using Novell's NDS-based networking products, licensing is calculated based on the number of connections made to the server, rather than the number of actual users. If your users need to authenticate to a specific server to place mail in the Mercury queue directory, then each such authentication will consume a license entry, even if the user primarily functions on another file server on the network.

    Creating a secondary queue on each server can dramatically reduce the number of licensed connections used on your NDS network, because the user does not have to establish separate authentication just to send mail - he or she can simply use the queue on the server to which they are currently attached.

Important note: when using secondary queues, it is up to you to ensure that the path is valid, and that the workstation where Mercury is running has been authenticated to the file server where the queue is located - Mercury does not attempt authentication by itself. We recommend in the strongest possible terms that you use the Windows UNC format to specify the location of the secondary queue, rather than a drive letter.

Entering user information

When you are using Mercury/32 in an environment for which it has no network support module, the option to create and manage local users becomes available on the Configuration menu. This option allows you to manage the Pegasus Mail user configuration file PMAIL.USR, which is used by both Mercury/32 and Pegasus Mail to define and locate the users who exist on the system.

Important note: Mercury reads the Pegasus Mail system file PMAIL.USR to obtain local user information, and caches it in memory while it runs. If you make changes to PMAIL.USR using Pegasus Mail or via any other process that manipulates the file directly, you will need open the Mercury user administration dialog while holding down the CTRL key, to force it to re-scan the list. Restarting Mercury will also work in this situation. Until Mercury refreshes its cached copy of the user list, it will not be able to deliver to users added via the alternative method. Mercury saves changes made by its own user dialog as soon as you click OK, so we recommend that you do all your user administration from within Mercury.

Username the name the user will give to identify himself to Pegasus Mail. By default, this is also the user's address. In order to remain compatible with the DOS and Win16 versions of Pegasus Mail, usernames are limited to eight characters total length (because they are used as the name of a directory).

Personal name the "familiar name" by which the user is known to the rest of the world. This is the name that appears in the Pegasus Mail *Local user list* function, and which Mercury uses when responding to mail server VERIFY and LOOKUP commands.

POP3 password The password the user needs to enter when retrieving mail via POP3. If a user does not have a POP3 password, or has a blank POP3 password, he or she cannot retrieve mail from Mercury/32 using the POP3 protocol. The password can be from 1 to 48 characters in length.

APOP secret APOP is a specialised POP3 protocol command that provides a reasonably secure form of login without the user having to send his or her password in clear text over the network. APOP is based on the idea of a shared secret - some piece of text known both to the user and to Mercury. If the user has a mail program that supports the APOP command, enter the shared secret in this field. The shared secret should be at least 8 characters long and can be any length up to 48 characters. It can be the same as the user's POP3 password, but need not be (and in fact, we do not normally recommend that it be so).

In networked mode (where Mercury is using a network interface module such as its Novell NetWare modules) the POP3 password and APOP secret will be obtained or defined in a different way - see the documentation provided with the module for more information.

Administrator privileges A user with administrator privileges can create, edit, rename and delete users on the system from within Pegasus Mail.

Copy default messages This control is only available when you are creating a user - it is disabled if you are editing a user's information. If checked, it tells Mercury to copy any file with the extension .DMI in the directory where MERCURY.EXE is installed into the new user's new mail directory. .DMI files (*Default Message Interface*) should be properly-formatted RFC822 mail messages and will appear in the user's new mail folder the first time he or she runs Pegasus Mail. They are normally used to provide information on using the system, or welcome information.

Configuring Pegasus Mail to work with Mercury/32

While Mercury/32 can be used with a variety of mail programs, it is primarily intended for use with Pegasus Mail, and has a special option to configure any copy of Pegasus Mail to work with it. To use this dialog, enter the full path to the directory where the Pegasus Mail .EXE file you want to configure is located. Mercury will create the files necessary to allow Pegasus Mail to submit mail to Mercury for processing.

Mail Submission (queue) directory   Enter here the path your copies of Pegasus Mail should use to find the directory where they should place mail for processing by Mercury. The default value is Mercury's own queue directory, but this may not be suitable for, or accessible by other workstations. As an example: say Mercury's queue directory is C:\QUEUE; in order to make that queue available to other users on the network, you need to set up a share - but you can't share it as C:, because all systems will already have a C: drive. Instead, you set up the share as T:. In this scenario, Mercury needs to access C:\QUEUE, but the Pegasus Mail clients need to submit mail in T:\QUEUE: the directory is the same, but the path specification is different.

If you have copies of Pegasus Mail (for instance, DOS and Windows versions) in different directories, you will need to use this option once for each version you have installed. At present, the Macintosh version of Pegasus Mail cannot interact with Mercury other than in native Novell NetWare mode.

Sending a mail message from within Mercury

Mercury/32 includes an option you can use to compose and send simple mail messages. The facility is rudimentary, and is not intended to replace a full-fledged mail client such as Pegasus Mail. To send a mail message, choose *Send message* from the *File* menu. Mercury will open the message editor dialog with the "From field" entry filled with the postmaster address for your system. Enter the recipient of the message in the *To:* field (you may enter more than one address, provided they are separated by commas), the subject of the message and the message body, then click the *Send* button.

Tip: If you want to use this option to send a mail message to a Mercury mailing list, make sure you change the From field of the message to something other than the postmaster address - Mercury's mailing list manager contains special traps to prevent mail storms, and one of those traps will prevent mail from the postmaster account being distributed to lists.

Urgent   sets the urgent flag in the message. Different mail clients will respond to this flag in different ways - Pegasus Mail, for instance, will display urgent messages in red at the top of the new mail list. This option does *not*, however, make the message travel any faster.

Confirm delivery   sets a flag in the message asking for a confirmation from the recipient's mail system that the message has been delivered to his mailbox. Getting a confirmation of delivery does not mean that the recipient has read the message - only that it has been successfully delivered and is available for him to read. Many, but not all, mail systems on the Internet will provide confirmation of delivery. The confirmation will be sent to whatever address appears in the *From field* of the message editor dialog.

Confirm reading   Sets a flag in the message that asks the recipient's mail program to confirm that the message has actually been read. At the time of writing, only Pegasus Mail supports this flag, and on some systems the user may be able to decline the confirmation (many people regard such confirmations as an invasion of privacy). The confirmation will be sent to whatever address appears in the *From field* of the message editor dialog.

You cannot attach files to messages sent using this facility, and the total size of the message may not exceed 30,000 characters.

Mercury/32 License

Mercury/32 is fully proprietary copyrighted software. David Harris ("The Author"), asserts his right to be recognized as the author and copyright owner under International Treaty and Copyright Law. The reservation of these rights notwithstanding, the Author wishes to provide the Mercury Software freely for the benefit of the broader Internet User Community, and especially for the benefit of users of his Pegasus Mail mail client.

You are granted a non-exclusive license to use the Mercury/32 software on as many computers as you wish subject to the following terms and conditions:

Consideration: You may use the Mercury/32 software without any fee or charge, although you may at your option choose to license manuals as described elsewhere in this guide. You agree that the author of the Mercury/32 system receives consideration from your use of the program in the form of acclaim and valuable reputation, and that this consideration is sufficient to bind you to the terms of this license even if you opt not to purchase manuals.

Prohibited uses: Mercury/32 may not be used as a transport for the dissemination of Bulk Unsolicited Commercial E-mail (commonly referred to as "spam"). For the purposes of this section, *Bulk Unsolicited Commercial E-mail* shall be taken to mean e-mail containing advertisements for products or services, which is sent to lists of more than 50 addresses where the addressees have not explicitly expressed interest in receiving such e-mail.

No transfer of rights: You agree that your use of the Mercury software entails no transfer of intellectual property rights, copyright or ownership from the Author.

No liability: The Mercury software is provided without warranty of performance. While the author has taken every care to ensure that the software works correctly and without fault, you agree to hold the Author blameless for any and all damage or loss caused by its use.

Distribution and supply: You may distribute the Mercury/32 Software to any person or organization by whatever means you wish provided that:

a) You do not charge more than reasonable handling or media production fees for the supply of the Mercury/32 Software itself.
b) You do not represent ownership or title over the Mercury/32 Software.
c) You supply the Mercury/32 Software complete and unmodified.
d) You do not supply the Mercury/32 software as part of, or in conjunction with, any package or promotion explicitly designed for the purpose of disseminating Bulk Unsolicited Commercial E-mail.

Setting up Automatic Replies

Mercury can manage auto replies for Internet mail, much the same way the UNIX "Vacation" program does. This means that when a message is delivered to an account with an active automatic reply, a message containing pre-determined "canned" text is automatically sent back to the person who sent the message.

There are essentially three different types of automatic reply supported by Mercury:

1: Simple autoreplies These are the easiest to set up, but the least flexible: the user simply creates a file in his or her new mail subdirectory called AREPLY.PM, containing the text to be returned to the sender. Mercury will look for this when delivering mail to the account, and will return its contents immediately the message is delivered locally. Pegasus Mail, Mercury's companion mail client, gives the user a convenient user interface for manipulating and enabling simple autoreplies.

2: Programmed autoreplies This type of automatic reply allows you to create a set of rules that Mercury should follow when choosing the text to return: you can base your rules on the day of the week, the month of the year, date ranges, or the time of day. Setting up programmed autoreplies requires a little more work - click here for details on creating the configuration file for programmed autoreplies.

3: Autoresponders An autoresponder differs from either of the other types of automatic reply in that once the automatic response has been sent, the message that triggered it is discarded - no further attempt is made to deliver the message. Autoresponders are implemented in Mercury as a specialized form of alias, and are extremely useful for creating things like FAQ servers and other addresses that simply return a textual message. Click here to go to the help section on *Aliases*, where autoresponders are covered.

To avoid the possibility of mail storms (which can happen when two accounts start auto-replying to each other, or when an autoreply is sent to a list server), Mercury remembers every address from which a message has been successfully delivered for the account in the last 48 hours; if more than one message comes in from the same address in any 48 hour period, Mercury will generate only one auto-reply. The auto-reply memory is stored in a file called AREPLY.KFL in the user's new mail directory.

You can also specify a static kill file for autoreplies on a user by user basis. When generating an autoreply, Mercury checks to see if there is a file called AREPLY.KFS in the user's mail directory. If there is, it is checked for the address before the autoreply is transmitted. AREPLY.KFS differs from the AREPLY.KFL file Mercury generates automatically in that Mercury never changes or deletes it. It is provided to allow a user to suppress certain autoreplies permanently. Addresses should be entered into AREPLY.KFS one per line, in their simplest form.

Static and dynamic autoreply killfiles will work for both simple and programmed autoreplies.

Creating and enabling programmable autoreplies

See also: Setting up automatic replies

When Mercury attempts to deliver a message to a user's mailbox, it checks to see if it can find a file called AREPLY.CFG in that mailbox. If it can, it assumes that a programmable autoreply is active for that mailbox. If AREPLY.CFG does not exist, Mercury will then check for the simple autoreply file AREPLY.PM instead.

AREPLY.CFG is a text file, containing a potentially unlimited number of definitions, each definition describing a set of circumstances under which Mercury should send a specific automatic response when a message is delivered to this mailbox. Mercury reads the definitions until it finds one that matches the conditions pertaining to the delivery; it then actions that definition. For this reason, the order of definitions in this file is very important - you should place the definitions with the narrowest boundaries at the top of the file, and the definitions with the broadest boundaries at the bottom. If Mercury passes through this file without finding any matching condition, no automatic reply will be sent at all (even if a simple AREPLY.PM automatic reply file also exists in the directory).

Each definition in AREPLY.CFG must appear on one line, and must start with a number from 0 to 4, which indicates the type of definition it is. All definitions have a time of day associated with them, and may have one other condition that is used to determine whether or not the definition applies:

0	Always triggers (no other condition required)
1	Triggers on a specific day of the week (0=Sunday .. 6=Saturday)
2	Triggers on a specific day of the week if that day of the week is also the "xth" in the month - this allows you to create a definition, for example, that triggers on the third Thursday of every month.
3	Triggers if we are in a specific month of the year
4	Triggers if the current date falls between a specific range of dates.

The layout of the remainder of the line depends on the type of definition:

Type 0	The remainder of the line is simply the standard fields (see below)
Type 1	The line contains a single integer representing the day of the week where the definition triggers. Sunday is 0, Monday is 1, Tuesday is 2 and so on through to Saturday, which is 6
Type 2	The line contains two integers; the first is the day of the week (as in type 1), while the second is the iteration of that day of the week within the month. So, to specify the third Thursday of the month, your definition would look like "2, 4, 3", where the "2" is the definition type, "4" indicates "Thursday", and "3" indicates that you want the third instance in the month.
Type 3	The line contains a single integer representing the month of the year, where January=1 through to December=12
Type 4	The line contains a pair of specially-formatted integers, each representing a date in the format YYYYMMDD. The first date **must** be earlier than the second date.

Standard fields

All definitions end with three mandatory standard fields and one optional standard field:

* The start time, specified as "HHMM" in a 24-hour clock. So, to make a definition that starts at 11:30 in the morning, enter *1130*.

* The end time, specified as "HHMM" in a 24-hour clock. So, to make a definition that ends at 8:05 in the evening, enter *2005*.

* The name of the file containing the data to send as an automatic reply. You can give either a fully-qualified pathname, or else a simple filename. If you supply a simple filename, Mercury will assume that it is located in the same directory as AREPLY.CFG.
* If the automatic reply is a valid Mercury/32 template file, add the optional value "1" at the end of the line. All the standard rules for template files apply, and you can use two special substitutions as well as the basic set - ~P0, which is replaced with the full e-mail address of the owner of the mailbox to which the message is being delivered, and ~P1, which is replaced by the date in the "Date" field of the original message.

Examples: (all assume a non-template autoreply file called AR.TXT)

A definition that triggers any day between Midnight and 6am:

```
0, 0000, 0600, AR.TXT
```

A definition that triggers every Friday between 6pm and 9pm:

```
1, 5, 1800, 2100, AR.TXT
```

A definition that triggers on the third Monday of each month between 8am and 10am:

```
2, 1, 3, 0800, 1000, AR.TXT
```

A definition that triggers between 8am and 5pm during April:

```
3, 4, 0800, 1700, AR.TXT
```

A definition that triggers all day between May 1 2003 and May 10 2003:

```
4, 20030501, 20030510, 0000, 2400, AR.TXT
```

Manuals

Manuals and technical support for Mercury are available for sale as an option - and we want to stress the word "option" in that sentence - you are under no obligation to purchase anything from us; we are happy simply to be of use.

If, however, you want to purchase manuals and support, you can do so by visiting our web site and going to the ordering page, http://www.pmail.com/mu_manuals.htm; manuals and support for Mercury are only available as part of our full support subscription agreements, which cover both Mercury and Pegasus Mail. The annual price for these agreements depends on the number of users Mercury is servicing for you, according to the following table:

1.5 users	US$125 per annum
6-20 users	US$225 per annum
21-100 users	US$350 per annum
101-500 users	US$595 per annum
501 - 1000 users	US$875 per annum
1001+ users	US$1000 per annum

The web page contains a link to a secure online ordering facility where you can order whichever option suits your needs. We are happy to accept company or institutional purchase orders on normal credit terms (nett 30 days).

Donations and reduced pricing: we have tried to make our pricing as reasonable as possible, but we're more interested in helping than in profiting. We like to donate manuals and support to non-profit organizations that work to preserve the environment, work for the benefit of animals, or are directly involved in improving conditions in the developing world. We're also open to being approached for reduced pricing by anyone who believes they can make a reasonable case for it. If you would like to discuss this further, please contact the author directly, at David.Harris@pmail.gen.nz.

Configuring Statistics and System Messages

See also: Core module configuration

Mercury keeps extensive statistics about its operation, which can be useful for working out loading and peak traffic times, and for tracking down performance bottlenecks. It also supports a System Messages window, which acts as a kind of "console" in which the various co-operating parts of the Mercury system can report events and messages.

Statistics

Save statistics to a file periodically Checking this control tells Mercury to save a "snapshot" of the current statistics tree to a file in the directory you specify at the frequency you specify. *Save in* should be filled out with the name of a directory (***not*** a filename) in which Mercury should create its snapshot files. Snapshot files have a name representing the date and hour they were saved, and the extension .MSR.

E-mail statistics periodically Checking this control tells Mercury that it should e-mail a "snapshot" of the current statistics tree to a specified address (which does not have to be local) at the frequency you specify. *Send to* should be filled out with any single e-mail address.

Automatically open the statistics window at startup If this control is checked, the statistics window will be opened automatically every time you run Mercury.

Collect statistics about mail sent by local users When this control is checked, Mercury will record the size and number of mail messages sent by each local user on your system. This can consume quite a lot of memory, particularly if you have many users, but it's an extremely useful way of seeing who is using your system's resources.

System Messages

System message reporting level This controls the verbosity of the information reported in the System Messages window. Level 0 disables reporting altogether, while level 5 is typically used to report low-level program status and debugging information. There is usually no reason to adjust this control from its default level 3 setting. At level 3, Mercury reports useful information about delivery problems and actions within your system.

Number of messages to store Controls the maximum number of items that will appear in the System Messages window. Once the total number of items in the window exceeds the number you specify, the earliest entries in the list will be successively discarded. The larger you set this number, the more memory will be used by the System Messages window, although the amounts are not enormous even at quite high numbers, like 1000 lines.

Automatically open the System Messages window at startup If this control is checked, the System Messages window will be opened automatically every time you run Mercury.

MercuryS, the Mercury/32 SMTP Server

MercuryS is the Mercury SMTP Server Module. It listens for incoming SMTP connections from the outside world, placing incoming mail in the Mercury spool directory for processing when it receives them.

SMTP (Simple Mail Transfer Protocol) is described in detail in Internet Standards Documents RFC821 and RFC1652, which are available via FTP from nic.ddn.mil in /rfc, as well as from various other places on the Internet.

Configuring MercuryS
Using SSL to offer secure connections

Configuring the Mercury SMTP Server module

See also: Logging and session logging
Connection control and relaying
Spam control
Compliance controls
Using SSL to offer secure connections

Announce myself as In some situations, you may wish to have your SMTP server to tell clients connecting to it that its name is something other than the value in the Mercury *My Name* field. An example of a situation when this might be necessary is when your *My Name* field represents an entire domain for which Mercury is acting, but you want it to identify itself to connecting clients using its real Internet machine name. In the majority of cases this field can and should be left blank.

TCP/IP Timeout the length of time in seconds that MercuryS should wait for data on a connection before assuming that the connection is no longer valid and aborting it.

ESMTP maximum size If non-zero, the maximum size message MercuryS should accept from compliant ESMTP clients. MercuryS will advertise this via the ESMTP SIZE keyword. Not all clients, even ESMTP clients, will honour this setting.

Listen on TCP/IP port By default, MercuryS listens for connections from the outside world on port 25, which is the standard reserved port for the SMTP protocol. In some cases, particularly when you are behind a firewall, you may wish to listen on an alternative port - enter the number of that port in this field. **If you change this field and save the dialog, you will need to exit and restart Mercury/32 before the change will take effect**.

IP Interface to use If your computer supports multiple IP interfaces, you can use this field to tell MercuryS which interface it should select when listening for connections: enter the interface as a dotted IP address in the general form *www.xxx.yyy.zzz*. As an example, your system may have one IP address assigned to a dialup PPP connection, and another, different IP address assigned to a local Ethernet network - you would enter here the interface MercuryS should use. If you leave this field blank, MercuryS will listen on all available interfaces. Unless you are *very* sure of what you are doing, or have been instructed by an ISP or network administrator, you should leave this field blank. If you change the IP interface in this field, you must restart Mercury before the new interface number will be used.

Sender kill file MercuryS allows you to create a file of addresses from which you will refuse to accept mail. The file can restrict individual addresses, or (using wildcard characters) entire domains or groups of users. This feature can be useful for dealing with spam, or with abusive correspondents. When a message is "killed" by the killfile, you don't even receive the data, so it is an excellent way of protecting yourself from denial of service attacks. Be careful, though - once someone is in your MercuryS killfile, they cannot send you mail at all - you will need to work out for yourself whether or not this presents any problems. To edit the contents of your killfile, click the *Edit* button next to the field. **Note:* using a killfile with more than a few dozen entries can impact significantly on the speed of incoming mail processing.

Display session debugging information Check this control if you want the MercuryS console screen to display more verbose information about each connection as it comes in.

Accept 8BITMIME data connections If this control is checked, Mercury will tell connecting clients that it supports the 8BITMIME SMTP extension. What this means is that Mercury will tell connecting systems that it can handle mail messages containing 8-bit data, bypassing the normal 7-bit restriction on Internet Mail data. It is very important to note that Mercury currently cannot convert 8-bit data to 7-bit data when it passes it on to other SMTP systems, as is required by the 8BITMIME specification: in practice, this is unlikely to cause problems in the majority of cases, but you should be aware that enabling this control has the potential to produce undesirable effects in rare instances.

Accept mail for invalid local addresses In regular use, MercuryS will refuse to accept any message that appears to be addressed to a local user who does not exist. This refusal can result in the sender getting unhelpful mail messages from their mail program. If you check this control, Mercury will accept the message and the Mercury core module will later reject it and send it back to the sender, but in a more clearly-explained form. Mercury will also refer a copy to the postmaster, who can then correct any addressing error and pass it on to the proper recipient.

MercuryS: Connection and relaying controls

See also: Configuring MercuryS
Spam control

Connection control

The Connection Control section allows you to place restrictions on the hosts from which MercuryS will accept connections, and to configure certain capabilities, such as relaying, based on the address of the connected host. A connection control entry can apply to a single address, or to a range of addresses. To add an entry to the list, click the *Add restriction* button; if you wish to create a restriction for a single address, enter that address in the "From" (left-hand) address field in normal dotted IP notation. To create a restriction for a range of addresses, enter the lowest address in the range you want to restrict in the "From" field, and the highest address you want to restrict in the "To" field. The addresses are inclusive, so both the addresses you enter are considered part of the range.

If you check the *Refuse connections* radio control, Mercury will not accept incoming connections from this address. Use this to suppress sites that are abusive or have been hijacked by spammers.

If you check the *Connections may relay through this server* control, Mercury will use this as part of the process it applies to determine whether or not a specific connection can relay mail (see below).

If you check the *Connections are exempt from transaction filtering* control, Mercury will not apply any transaction-level filtering expressions you might have created to filter the commands supplied by connected clients; this is particularly useful, or even essential if you have local workstations running clients like Pegasus Mail or Eudora that need relaying facilities via your server.

To edit a connection control entry, highlight it in the list, then click the *Change selection* button.

How Mercury applies connection control entries

The list of connection control entries you create can contain entries that overlap (i.e, entries that refer to addresses also covered by other entries). In the case of overlapping entries, Mercury uses the following method to select the entry it should use for any given address: if there is an entry that refers to the address on its own (not as part of a range), then Mercury will automatically use that entry; otherwise, it looks for the range that most closely encompasses the address and uses that.

Example: You have a *Refuse* entry covering the range from 198.2.5.1 to 198.2.5.128, and an *Allow* entry covering the range from 198.2.5.10 to 198.2.5.20: if a machine with the address 198.2.5.12 connects to Mercury, it will select the *Allow* entry to cover the connection, because the allow entry most tightly encompasses the connecting address (the range covers 11 addresses, where the Refuse entry's range covers 128 addresses).

Controlling relaying

SMTP relaying is the standard method of propagating mail on the Internet: in normal operation, an SMTP host will accept any message destined for any user, even if that user is not a local user on the system: after it has accepted the message, it will *relay* it to the correct host for delivery. Mail agents like Pegasus Mail and Eudora routinely depend on relaying to send mail.

In recent times, relaying has been abused by perpetrators of mass unsolicited commercial e-mail (or "spam"), and many sites wish to control the way relaying is managed. Mercury provides two anti-relaying modes, *normal* and *strict*. Normal mode is turned on by checking the control labelled "*Do not permit SMTP relaying of non-local mail*". Strict mode is turned on by also checking the control labelled "*Use strict* *local relaying restrictions*". The default mode is *Normal relaying control*, although this can be overridden during installation and at any later time..

In either mode, Mercury will ***always*** accept mail addressed to any local address. Similarly, mail to any address for which Mercury holds an alias will also be accepted, even if the alias resolves to a non-local address. Both modes can be overridden by an optional requirement for SMTP authentication (see below for more information).

In normal anti-relaying mode, Mercury will accept mail for delivery if either the recipient or the originator has a local e-mail address. If neither address is local, Mercury will compare the IP address of the connecting host to its connection control list (see above): if it finds an "allow" entry in that list that explicitly includes the connecting machine, then it will accept the mail, otherwise it will be failed with a diagnostic like "553 - We do not relay...".

In strict anti-relaying mode, Mercury follows the normal rules described above, but if the "From" address appears to be local, then Mercury will search the connection control list and will only accept the mail if an "allow" entry appears that explicitly permits the connecting host.

The difference between the two modes is that normal mode requires less setup and maintenance, but is less secure, while strict mode practically guarantees that no unauthorised relaying can occur at the expense of having to manage a list of permitted relay hosts.

When you configure Mercury to operate in strict mode, you must ensure that you add "allow" entries to your connection control list for every machine that is to be permitted to relay mail via this copy of Mercury. Note that this does NOT mean that you have to enter the address of every machine from which you want to accept mail - mail to local recipients is always accepted, regardless of the relaying mode. Strict mode only requires "allow" entries for machines from which Mercury is to accept mail to be delivered to non-local addresses. Please note that this use of the allow/refuse list is an *overload* - that is, it means that the allow/refuse list is being used for two separate purposes, connection control and relaying control. This overloading is possible because relay control only uses *allow* entries, and any host that can relay from your machine is by definition also allowed to connect to it. You should never use *refuse* entries to handle relay control - only *allow* entries.

It is almost always safe to turn on normal anti-relaying mode.

Authenticated SMTP

Mercury supports a recent Internet standard called *Authenticated SMTP*: when this feature is enabled, Mercury will advertise to connecting clients that it can accept SMTP authentication. If a client then authenticates correctly, it will be allowed to relay. Pegasus Mail v3.12 and other widely-used Internet mail clients support authenticated SMTP, and it is an excellent way of allowing your roving users to use your server without opening yourself to relay abuse. Mercury supports two Authentication methods - CRAM-MD5 and LOGIN, although LOGIN is very weak and you should avoid clients that use it.

Authenticated SMTP requires that both the client and server have access to a common password. For that reason, you need to provide Mercury with a list of usernames and the passwords that correspond to them - Mercury typically cannot get this information from the operating system. Enter the name of the file where Mercury should store the user/password combinations, then click the *Edit* button to edit it. Each line contains one username/password pair.

If you check the control marked *Authenticated SMTP connections may relay mail*, then any authenticated connection will be permitted to relay messages even if it would otherwise have been prevented from doing so by either the normal or strict relaying tests (see above).

If you check the control marked *Only Authenticated SMTP connections may relay mail*, then SMTP authentication becomes mandatory - a non-authenticated connection will not be permitted to relay mail even if it would otherwise have been permitted to do so by either the normal or strict relaying tests. Because this option supersedes all other tests, selecting it will disable the normal and strict controls in the dialog.

Spam (Blacklist) Controls

See also: Configuring the Mercury SMTP Server
Creating a blacklist query definition

Spam, or Unsolicited Commercial E-mail, has become a serious nuisance on the Internet over the last few years, and considerable efforts have been made to find ways of preventing or lessening its impact. One of the more successful of these has been the creation of "blacklist databases", such as Paul Vixie's MAPS RBL, and the now-defunct ORBS open relay database. These "blacklist databases" are essentially just Internet name servers that list either machines or Internet domains that are known to be sources of unwanted mail.

When an incoming mail connection is made, a query-enabled server, such as Mercury, can send a simple name server query to these databases to find out if the machine or the sender of the incoming mail message is listed; if the blacklist database reports that it knows the address or domain, then the server can take whatever action has been programmed for it, such as rejecting the mail, or maybe quarantining it somewhere.

On the surface, this blacklist approach sounds like a good way of reducing unwanted mail, and indeed it can be - but you ***absolutely must*** understand two key issues associated with this type of technique:

1: Blacklist databases **will not prevent all spam**. You should not believe that simply by turning these functions on, you will prevent spam from reaching your site. These functions may reduce the amount of spam you receive, but nothing except co-ordinated global legislation can eliminate it. We suggest strongly that you visit the CAUCE (Coalition Against UCE) web pages at http://www.cauce.org for an overview of the positive steps you can take to combat spam.

2: From time to time, using a blacklist definition **will** inconvenience bona fide senders, and **will** result in legitimate mail not being delivered to your site. None of the blacklist services currently in existence is an official body - all are ad-hoc and subject to standard human failings. From time to time, legitimate sites will be affected by these databases, and legitimate users may sometimes find that actions for which they have no responsibility at all have resulted in their mail being blocked.

We recommend in the strongest possible terms that you think long and hard before enabling these controls: they are certainly useful, but they are also dangerous.

Whitelists

The same technique used by blacklist query servers could also be used to create "whitelists" - servers listing machines that are always acceptable. At the time of writing, no public whitelists exist on the Internet that we know of, but if you control your own local domain name server, there is nothing to prevent you from entering addresses in the proper form within that database then creating a Mercury definition to query it. This approach could be useful if you need to correspond with a site that has somehow become blacklisted, without turning off blacklist controls for other sites.

Query formation, and method of operation

The process used to query a blacklist database is simplicity itself. Depending on its configuration, Mercury takes either the physical IP address of the connecting mail client, or the domain name portion of the sender's e-mail address; it then reverses and appends this information to a static domain name you supply and attempts to do a simple domain name resolution on the resulting string. If the domain name can be resolved, then the address is regarded as being blacklisted. The IP address is reversed to conform to the normal standards for Internet name resolution.

Example: The machine *190.47.32.33* connects to Mercury, which has been configured to use the MAPS

RBL blacklist at *blackholes.mail-abuse.org*. Mercury constructs the domain name *33.32.47.190.blackholes.mail-abuse.org* and attempts to resolve it as a domain name.

You can create as many query definitions as you wish: Mercury will action the query definitions in the order they appear in the list in the *Spam control* page, stopping as soon as a definition results in a positive response from the service. If the responding service is marked as a whitelist, Mercury will take no further action against the message, allowing it to be processed normally - otherwise it will apply the action you have defined in the definition. Once a site has responded, no further definitions in the list are checked. It should be clear that the order of the definitions in the list can have an important bearing on the way your mail is processed - you can use the *Up* and *Down* buttons underneath the list of definitions to rearrange the definitions within the list.

Finding servers

Originally, two services existed providing blacklist services - the RBL, and ORBS. Each took a very different approach to dealing with the problem of handling unwanted mail: the RBL blacklisted sites known to have been involved in mail abuse, while the ORBS system punished any sites with what are technically known as *Open Relays* (this means a mail server that would accept mail for non-local addresses and forward it on). The RBL's approach attempted to identify real abusers, while ORBS attempted to identify systems that might be used for abuse, whether or not they actually had any history of it. Both systems had their share of critics, but ORBS in particular was excessively ad-hoc and was eventually shut down by a lawsuit. These days, there are many systems on the Internet providing a variety of blacklist services, some on a subscription basis, others free. Because the list of these services is fluid, there is little point listing them in this help file, but the original RBL appears to be in for the long haul (although it is on a paid subscription basis these days), so it's a good place to start looking: you can find information on the MAPS RBL by visiting http://www.mail-abuse.org.

The Coalition Against Unwanted Commercial E-mail (CAUCE) is the premier anti-spam lobby group, and its web site, http://www.cauce.org, is an essential reference for anyone interested in fighting spam. Its *Other resources* link usually lists several spam blacklist servers or organizations you can contact.

Defining a spam blacklist query definition

See also: Spam (blacklist) controls

To create a spam blacklist definition, you need to know a certain amount of information about the service itself; you will typically find this information on the service's web site, or in the subscription package you get if the service is one for which you pay.

Name for this definition The name Mercury should display in the Spam Control dialog for this entry. Mercury also uses this name to construct X-BLOCKED headers by default when tagging mail messages that are blacklisted. The name you enter can be up to 50 characters in length, and should not contain special or international characters.

Hostname used to form query To query a blacklist, Mercury appends the name it wishes to query to a static domain name which is supplied by the service itself. You should enter the static domain name part of the query in this field. As an example, at the time of writing, the hostname for the MAPS RBL blacklist database is *blackholes.mail-abuse.org*: other services will use different hostnames.

Type of query service Mercury supports two types of service - *blacklists* and *whitelists*. A blacklist lists abusers or sites that should be be regarded as hostile or pernicious, while a whitelist lists sites or machines that should be regarded as acceptable under any circumstances. Mercury scans the list of definitions you create in order, trying each. As soon as a definition "triggers", Mercury uses it; if the definition is a blacklist, the action associated with the definition is applied to the message. If the definition is a whitelist, Mercury takes no further action against the message. At the time of writing, there are no whitelist services publicly available on the Internet, but there is no reason for them not to exist in future.

Query structure Depending on the service, Mercury can query either based on the IP address of the SMPT client connecting to it, or on the domain part of the e-mail address of the person sending the message. If the service you are defining checks IP addresses, select *Address-based* in this dialog. If, however, the service checks domain name portions of e-mail addresses, select *Domain based* in this dialog. You must select the proper type of query for the service - if you select the wrong type, then you will not necessarily see any errors later, but no mail will ever be blocked by the service.

Strictness level of response Some blacklist servers can return a variety of different values, indicating either the reason for the blacklisting of the address, or in some cases, an indication of the severity of the "offense" that resulted in the blacklisting. Mercury supports three separate ways of evaluating the response from the server. Before describing the methods Mercury offers, a small digression is necessary to explain how these blacklist services work. Mercury creates a special domain name based on the address (either IP or domain) of the originator of the message, then attempts to resolve that domain name using a standard name resolution call. If the domain is unknown or cannot be resolved, then no listing is currently held for it. If, however, the attempt to resolve the name is successful, an IP address will be returned to Mercury, indicating that the address is blacklisted. The address returned to Mercury will be of the form 127.0.0.x, where "x" is a value greater than 1. In almost all cases, only the last byte of this address will vary depending on the type of blacklist in operation - so, some servers may simply return "127.0.0.2" if they hold a blacklist entry for the address, while others may return anything from 127.0.0.2 to 127.0.0.10 or even higher to indicate the type of listing held. With this digression in mind, here is how Mercury manages its three strictness modes:

Normal	Mercury only regards the message as blacklisted if the remote name server returns the value 127.0.0.2. Any higher value returned by the server will not result in a blacklist response.
Any	(Called "*Draconian*" in previous versions of Mercury) Mercury will regard the message as blacklisted if the

name server returns any successful response at all. Use this option with care - it can result in a potentially unacceptably high level of otherwise legitimate mail being blocked depending on the blacklist service.

Range	Allows you to specify a range of name server returns within which the address must fall before Mercury should regard the message as blacklisted. Checking this control will enable the *Range Low* and *Range High* edit fields: enter the lowest return Mercury should regard as a blacklist result as an IP address in the *Range Low* field, and the highest address Mercury should regard as a blacklist result in the *Range High* field. The addresses are inclusive, so if you enter 127.0.0.3 in *Range Low* and 127.0.0.4 in *Range High*, a return of either 127.0.0.3 or 127.0.0.4 will result in Mercury regarding the message as blacklisted, but a return of 127.0.0.2 or 127.0.0.5 will not.

Actions to take when a message is blacklisted

When a service returns a value indicating that the message should be blacklisted, Mercury can perform any of three different actions:

Reject the message When this action is selected, Mercury will refuse to accept the message, and will return a brief one-line message to the remote SMTP client explaining why it has done so. It is very important that you make the rejection message clear - ideally, it should contain a reference to a web site that explains to the sender why their mail has been blocked and how to rectify the problem. Most blacklist services will have such a web page you can reference in your rejection text. The primary advantage of rejecting blacklisted mail is that no bandwidth is consumed in receiving it; the disadvantage is that there is no way for a sender blacklisted in error to contact you by e-mail, because his or her messages will always be rejected.

Tag the message with a header When this action is selected, Mercury will accept the message normally, but will add a header to it in transit. If you leave the *Header* field blank, Mercury will add the header "*X-Blocked: <definition_name>*" to the message, otherwise it will add whatever text you enter without modification. If you enter a header, you must include the keyword (for example, "X-Blocked"), the colon character (":") and the parameter text. Tagging a message in this way allows your users to take advantage of the mail filtering capabilities of their mail packages to handle blacklisted mail in the way they feel is best.

Redirect (forward) the message to another address When this option is selected, Mercury will accept the message normally, but will ignore the recipients specified for it, instead sending it to the address you supply. The address you supply can be any valid local user or Internet address. Mercury also adds an "*X-Blocked: <definition_name>*" header to the redirected message.

Disable this definition (do not use it to perform queries) Check this control if you want to prevent MercuryS from using the definition without actually removing it.

Definitions are stored in a file called MS_SPAM.MER in the directory where MERCURY is installed.

Compliance (restricting certain types of message)

See also: Configuring the Mercury SMTP Server

MercuryS allows you to perform a number of checks on messages as they are received, and to reject messages if any of those checks fails. The types of test can be broadly divided into those that examine the way the SMTP protocol is being used, and those that do basic inspection of the message data. While many of the checks that MercuryS can do can also be done in other places within the Mercury system (especially using filtering) the advantage of doing them at the protocol level is that it prevents non-compliant messages from even entering the Mercury processing queue, thus cutting down the time Mercury wastes processing messages you don't want anyway.

Restrictions to apply at the transaction level

These restrictions examine aspects of the way the connecting client is using the SMTP protocol (the "language" that two mail programs use when talking to each other to exchange mail - covered in the Internet standards document RFC2821, available from http://www.ietf.org).

Require clients to use an ESMTP "Size" declaration If you check this control, MercuryS will only accept mail where the connected SMTP client declares the size of the message in advance, as part of the MAIL TO: command. Turning this on allows Mercury to enforce the maximum size requirements you specify without ever actually having to receive the data, but it *may* cause problems for some older clients that do not use the ESMTP size declaration extension. We recommend exercising caution when using this option, although it may be very useful in some environments where a degree of "bulletproofing" is required.

Limit maximum number of failed RCPT commands to... An increasingly common technique used by spammers to "harvest" valid addresses is to connect to an SMTP server and issue a long list of RCPT TO: commands, building the recipient addresses using a dictionary of common usernames. If the server accepts the RCPT TO: command, the harvesting program can note that the address appears to be good, and add it to a spam list. To help minimize the impact of this kind of attack, Mercury allows you to limit the number of failed RCPT TO: commands it will accept in a single session. This is almost always safe, since in the vast majority of legitimate mail scenarios, there should be no failed RCPT TO commands anyway. We recommend setting this to a value no lower than 3.

Limit maximum number of relay attempts to... Relaying occurs when someone asks your copy of Mercury to accept a message addressed to a non-local user and forward it on to that person. Originally, relaying was benign and useful, a good example of the co-operative spirit of the Internet. Unfortunately, just as they have polluted everything else they have touched, spammers have abused relaying to the point where it now has to be massively controlled. Mercury incorporates a wide range of relaying controls that allow you to manage people with legitimate reasons for relaying on your system, and if you use those controls, then this setting allows you to cut off people who attempt unauthorized relaying before they waste too much of your bandwidth or processing power. Because a legitimate message may appear to be a relay attempt (the address may have been mis-typed, for instance) we recommend that you set this value at a level that allows honest mistakes but still penalizes attempts at cynical abuse - 3 is usually a good number, and we counsel caution if you plan on setting a lower value.

Enable short-term blacklisting for compliance failures If you check this control, MercuryS will note the IP addresses of systems that exceed the limits you set for relaying and RCPT command failures (see above) and will prevent them from connecting for a period of 30 minutes. This is intended to make life difficult for spammers and other undesirable elements who may attempt to "harvest" addresses from your system by dictionary attacks. The short-term blacklist is automatically cleared if you restart Mercury. Transaction-level filtering expressions (see below) can also result in a system being blacklisted on a short-term basis, but only if this control is checked.

Enable transaction-level expression filtering When this control is checked, MercuryS will apply a set of regular expression-based rules you provide to each message as it processes it. These expressions can be used to test the HELO/EHLO (connection phase) of the SMTP transaction, and can also test the subject line of messages as they are received. They differ from other types of filtering in Mercury in that they can prevent a message you know you don't want from being received at all - Mercury can either drop the connection, or can simply discard the data without even placing it in the queue, thus reducing bandwidth waste and processing overhead on your system. Transaction filters are especially useful for detecting and suppressing specific types of messages that have readily identifiable features, such as connections from address harvesters, or attempted deliveries from systems infected by Outlook viruses or trojans. Click here for an overview of the syntax used by the transaction filtering expression file.

Restrictions to apply to message content

These restrictions examine the headers of the message as it is passing through the DATA state, and allow you to reject certain types of message that you don't want to receive. If any of these tests fail, Mercury will accept the remainder of the data (because the SMTP protocol does not provide any means for the server to cancel a transaction in progress), but will discard it, so that it never passes through the Mercury mail queue. A suitable error will be returned to the connected SMTP client so that the sender knows why their message was rejected.

Check originator address fields against the killfile Mercury's killfile allows you to specify particular addresses or domains from which you do not want to receive mail at all. Normally, the killfile is only checked against the *envelope address* - the address the remote system offers as the sender of the mail. If you check this control, MercuryS will burrow into the message as it receives it and will compare the killfile against the From, Reply-to and Sender fields in the message as well, ensuring that someone you have blacklisted cannot sneak into your mail server by forging an envelope address. Checking this option will slow down reception of mail slightly, but if you use the killfile feature in Mercury, it is almost certainly worth the slight processing overhead to enable this option.

Refuse messages containing pure HTML data HTML mail can take two forms - alternative formatting, where the message includes both plain text and HTML variants of the data and the user's mail client chooses which one is preferred, or pure HTML, where the only content in the message is HTML data - there is no plain text variant. HTML is the number one source of viruses, trojan horses and other security problems in modern e-mail, and in our experience, practically all mail that contains only pure HTML data is either viral or spam. Turning this flag on tells Mercury to refuse messages that only contain HTML data, although it will still accept messages in the alternative format, because they are at least nominally safe (especially if you are using a mail client such as Pegasus Mail, which is immune to HTML-based attacks).

Refuse non-MIME messages MIME has been the dominant Internet standard for formatting electronic mail since 1992, and there is no longer any justification for mail systems not to use it. Turning this flag on tells Mercury that only mail with valid MIME signatures should be accepted; it is especially useful when combined with pure HTML refusal (see above).

Refuse messages that have no 'subject' field We think it's a matter of basic courtesy to include a subject line in the mail you send. Turning this switch on allows you to enforce that requirement, although an empty subject field will still be accepted, provided at least the header is present.

Refuse messages that have no or empty 'subject' fields This is a more draconian version of the previous setting: if you turn it on, messages will only be accepted if they contain a subject field, and that subject field in turn contains non-blank data.

Refuse messages that have no 'date' field The Internet standards governing e-mail require that all mail must contain a valid date header. In our experience, practically the only mail that does not meet this requirement is spam.

Exceptions Just like everything else in life, all these compliance conditions are subject to cases of "yes, but..." - there are always going to be a few exceptions: for instance, you may have subscribed to a particular newsletter that regrettably only comes out in pure HTML format, or you may have an automated server somewhere that sends you progress reports that don't have a "date" field (we know of several backup programs like this). To get around this, enter a valid local filename in this field, then click the *Edit* button next to the field, and add the sender address that should be exempted from the compliance restrictions. You can use * as a wildcard character anywhere in the address if you want to exempt entire groups or domains - so, for example, entering *@pmail.com would allow mail from any user at the "pmail.com" domain to pass through even if it failed one or more compliance tests. Exceptions only apply to the controls in the *Restrictions to apply to message content* group, not to the transaction-related group.

MercuryC, SMTP Relay Client

See also: The MercuryX Scheduling module and ETRN

MercuryC passes mail from your system to the outside world using SMTP (the Simple Mail Transport Protocol). MercuryC is a *relay client*: instead of delivering the mail directly to its final destination, it asks a nearby system to deliver the mail on its behalf. Relay delivery is particularly well-suited to dialup connections, and for use behind firewalls, since in each case, you are asking a machine which is better suited to the task to deliver the mail.

Configuring MercuryC

MercuryC is the module that the MercuryX Connection Scheduler can use to trigger remote SMTP queues via an SMTP extension called *ETRN*. Click here for more information on using ETRN.

Configuring the MercuryC SMTP Relay Client Module

See also: Logging and session logging
The MercuryX Scheduling module and ETRN

SMTP "Smart" host details:

Smart host Enter here the name or address of the machine which MercuryC will ask to relay mail on its behalf. If you are using a dialup connection to an Internet Service Provider, then you will usually enter the address of one of your service provider's machines here - ask your Service Provider what to use. If you are behind a firewall, ask your network administrator which machine has SMTP access to the outside world through the firewall - that will be the machine to enter here. Any full SMTP server can be used for relaying, although it is common to apply controls to who can and cannot relay through systems.

Connect on TCP/IP port This is the port on the smart host to which MercuryC should connect. The standard port defined for this is 25, but in some cases (most notably if you are behind a firewall) you may have to enter a different port number here. Consult your ISP or Network administrator to find out if you need to alter the setting of this field.

Announce myself as The SMTP protocol has an identification phase, during which the client will tell the server its name. MercuryC will usually tell the server the name defined under *Internet name for this system* in the Mercury Core configuration dialog, but on rare cases you may need it to use a different name. If this is the case, enter that name here, otherwise leave this field blank. This option is inherently quite technical and you should enter a value here only if you are sure of what you are doing.

Credentials for SMTP Authentication:

With the growth of malicious "spam" (unsolicited commercial junk mail) on the Internet, many sites have begun placing restrictions on who may use their "smart" mailers to relay mail. One of the common ways of enforcing this restriction is to require authentication of some kind before accepting relayed mail. Mercury supports two types of authentication - authentication via prior POP3 connection, and authentication via the extended SMTP AUTH command.

With authentication via prior POP3 connection, Mercury does a simple POP3 login to a POP3 server: if your login is successful, then the POP3 server tells the smart SMTP server that it is OK to accept mail from your machine for a certain time (usually ten minutes or so). Mercury can then connect normally and send mail. If your ISP uses this method to enforce authentication, check the control labelled *Authenticate via prior POP3 connection*, fill in the address of the POP3 server and put the proper POP3 username and password into the *Username* and *Password* fields respectively.

Authentication via the extended SMTP AUTH command is handled automatically by Mercury: if you supply a username and password and have not checked the *Authenticate via prior POP3 connection* control, Mercury will attempt to use AUTH instead. Mercury supports the two most commonly-used variations of the AUTH command, LOGIN and CRAM-MD5. You do not have to worry which gets used - Mercury will automatically detect which variations are available and choose accordingly. Your ISP will be able to tell you whether his SMTP server supports SMTP AUTH.

General settings:

Delivery failure template The path to a template file which MercuryC should use to format delivery failures. The Mercury installer will create a default template file for you which should be suitable for most cases.

TCP/IP timeout the length of time in seconds that MercuryC should wait for data on a connection before assuming that the connection is no longer valid and aborting it. If you are connecting to a smart host that runs the Sendmail SMTP server, you may have to set this value to a quite large number (for instance, 600), because Sendmail insists on validating each address as MercuryC supplies it, which can take time.

Poll the queue every xx seconds How often MercuryC should check the queue to see whether there are any messages it should send.

Use extended SMTP features where possible SMTP is a very old specification, but in the last couple of years, a number of attempts have been made to modernize it. These modernization efforts are designed to be backwards-compatible - so, a client that uses them should still be able to communicate with older servers that do not understand the new features. In rare cases, you may encounter an old server that cannot cope with the new features; when this happens, you may have to uncheck this control. In general, this control should always be left checked unless you find you have specific reasons for unchecking it.

MercuryP, POP3 Server

MercuryP is a POP3 server - it allows users to retrieve their new mail without having physical access to the disk drive where it is stored .MercuryP presents a list of new mail messages to the POP3 client, which then decides which ones it wishes to download to the remote system.

If you are using Pegasus Mail as your electronic mail program, you will not usually use POP3 to retrieve your mail, because Pegasus Mail can read the mail directly from the directory on the file server where it is stored. There is nothing to preclude you from using MercuryP with Pegasus Mail, however, and if you are using other POP3 mail clients, they will interact with MercuryP.

Configuring MercuryP
Using SSL to offer secure connections

Configuring the MercuryP POP3 Server module

See also: Logging and session logging
Using SSL to offer secure connections

IP Interface to use If your computer supports multiple IP interfaces, you can use this field to tell MercuryP which interface it should select when listening for connections: enter the interface as a dotted IP address in the general form *www.xxx.yyy.zzz*. As an example, your system may have one IP address assigned to a dialup PPP connection, and another, different IP address assigned to a local Ethernet network - you would enter here the interface MercuryP should use. If you leave this field blank, MercuryP will listen on all available interfaces. Unless you are *very* sure of what you are doing, or have been instructed by an ISP or network administrator, you should leave this field blank. If you change the IP interface in this field, you must restart Mercury before the new interface number will be used.

Listen on TCP/IP port The TCP/IP port is the socket into which the POP3 client "plugs" to access MercuryP's services. The standard port for POP3 services is 110, but in rare cases (particularly if you use a proxy server) you may need to change this. Consult your ISP or network administrator to find out if you need to use an alternative POP3 port. If you are unsure, leave it set to 110.

Global profile settings

These settings control how Mercury should handle some aspects of connections from POP3 clients. These controls apply global settings - settings which will apply to all your users. You can also create user-by-user POP3 profiles to deal with the case where a specific user's POP3 client needs adjustment (see below for more information). In general, the default settings in this dialog will work with the majority of POP3 clients.

Mark retrieved mail as "read" When this control is checked, MercuryP will change the status of all messages downloaded by POP3 clients to "read". This can be used in conjunction with the "offer only unread mail" option to allow users to leave copies of all their mail on the server, but be offered only mail they have not seen.

Offer only unread mail If this control is checked, Mercury will list only messages whose status is "unread" - mail marked as read will be invisible to POP3 clients. This option is very convenient, but is also not a part of the POP3 standard.

Manufacture status headers Some mail programs (most notably Qualcomm's Eudora) rely on finding a non-standard mail header called "Status" in the headers of the message to determine whether or not a message has been read. The "status" header is generated by some unix mail servers, but is not a standard part of the Internet mail specification. MercuryP does not use status headers, but if this control is checked, it will create them "on-the-fly" for the benefit of POP3 clients that depend on them. There is usually no reason to uncheck this control, except that it adds a very small extra data overhead to POP3 downloads. If you are using Eudora as your mail client, this control should always be checked.

Ignore POP3 delete commands If you want to ensure that no mail is ever deleted from mailboxes, check this control. This tells MercuryP to disregard requests from clients to delete messages on the server. If this control is checked and Offer only unread mail is unchecked, then the user will be presented with the same mail messages every time he connects. Use this option with care.

POP3 deletions survive resets The POP3 standard states that when a POP3 connection terminates abnormally (for instance, because of nework problems), then the mailbox must be restored to the state it was in when the connection was established; all deletions must be undone and the status of all messages is returned to what it was at the time of connection. Checking this control tells Mercury that it should not delete messages that the POP3 client has marked for deletion even if the connection is closed abnormally. This option should be used with considerable care, but may be useful in situations where network stability is often unreliable.

Local profile settings

Local profile settings can be made on a per-user basis by creating a file called POP3.PRO in the user's new mail directory. POP3.PRO can contain any of the following statements:

```
Mark read
Show read
Show status
No delete
Delete is final
```

Each statement can be set to Y or N to enable or disable that setting. For example, to create a POP3 profile for a user that marks all downloaded mail as read and where deletions survive resets, you would add the following two lines to POP3.PRO:

```
mark read : Y
delete is final : Y
```

Statements missing from the file will use the default value determined by the Global profile setting controls (see above). Statements in POP3.PRO are not case sensitive.

Connection control

The Connection Control section allows you to place restrictions on the hosts from which MercuryP will accept connections, and to configure certain capabilities based on the address of the connected host. A connection control entry can apply to a single address, or to a range of addresses. To add an entry to the list, click the *Add restriction* button; if you wish to create a restriction for a single address, enter that address in the "From" (left-hand) address field in normal dotted IP notation. To create a restriction for a range of addresses, enter the lowest address in the range you want to restrict in the "From" field, and the highest address you want to restrict in the "To" field. The addresses are inclusive, so both the addresses you enter are considered part of the range.

If you check the *Refuse connections* radio control, Mercury will not accept incoming connections from this address. Use this to suppress sites that are abusive or have been hijacked by spammers.

To edit a connection control entry, highlight it in the list, then click the *Change selection* button.

How Mercury applies connection control entries

The list of connection control entries you create can contain entries that overlap (i.e, entries that refer to addresses also covered by other entries). In the case of overlapping entries, Mercury uses the following method to select the entry it should use for any given address: if there is an entry that refers to the address on its own (not as part of a range), then Mercury will automatically use that entry; otherwise, it looks for the range that most closely encompasses the address and uses that.

Example: You have a *Refuse* entry covering the range from 198.2.5.1 to 198.2.5.128, and an *Allow* entry covering the range from 198.2.5.10 to 198.2.5.20: if a machine with the address 198.2.5.12 connects to Mercury, it will select the *Allow* entry to cover the connection, because the allow entry most tightly encompasses the connecting address (the range covers 11 addresses, where the Refuse entry's range covers 128 addresses).

MercuryD, Distributing POP3 Client

MercuryD is a POP3 Client Module designed to retrieve mail from as many remote hosts as you wish and to distribute that mail to users on your local system or network.

MercuryD can retrieve mail from a remote account and deliver it all to a single user, or, if the remote account is a so-called *Domain Mailbox*, where all mail addressed to any user at a specific domain is placed in a single mailbox, then MercuryD can distribute the mail from that mailbox to the appropriate local addressees by interrogating the address fields of each message.

You will typically use MercuryD instead of the MercuryS SMTP Server module, in situations where you want intermittent dialup access to the Internet (say, once every hour or so). The two are not incompatible, however, and there may be occasions where you might want to load both modules.

Configuring MercuryD

Configuring the MercuryD Distributing POP3 Client module

See also: Logging and session logging

Work directory Enter here a path to a directory where MercuryD can create temporary files during the download process. The directory should be on a volume with plenty of free space (at least 15MB is recommended).

Check every x seconds This setting controls the frequency with which MercuryD should go through the list of accounts checking them for new mail. For example, if you want MercuryD to check for new mail once per hour, you will enter 3600 in this field.

TCP/IP Timeout the length of time in seconds that MercuryS should wait for data on a connection before assuming that the connection is no longer valid and aborting it.

POP3 account information

This section contains the login information for each account MercuryD is to check for new mail. Each entry consists of a host, a username, a password, and the name of the local user who should receive the mail from the account.

Host The name or IP address of the machine to which MercuryD should connect via the POP3 protocol when checking the account for new mail.

Username The login name MercuryD should use when connecting to the POP3 server.

Password The password matching the username for the POP3 account

Local user If you enter the name of a local user on your system (one to which Mercury can delivery directly) then all the mail downloaded from the remote account will be sent to that local user, irrespective of the address fields in the message. If you leave this field blank, MercuryD will examine the *To*, *CC* and *BCC* fields of each message looking for addresses it recognizes as local. When it finds a local address, it will send a copy of the message to that local user. This facility allows you to have a single mailbox (called a *Domain Mailbox* by most Internet Service Providers) into which all mail for any users at a specified domain is placed; MercuryD can then retrieve the mail from that mailbox and route it to the appropriate local users for you.

Default user When distributing mail from a domain mailbox, MercuryD may encounter messages for whom it can find no local recipient; this will commonly happen if one of your users subscribes to a mailing list, since mailing lists usually do not indicate the actual recipient anywhere in the message headers. In cases such as this, MercuryD can be told to deliver the message to a specific, or *default* user. If you leave this field blank, MercuryD will discard any messages for which it can find no local delivery addresses. This field is only meaningful when you have told MercuryD to distribute mail (by leaving the Local user field blank).

To add an account to the MercuryD service list, simply click the *Add* button then fill in the six account controls and click *OK*. To edit an entry, click once on its entry in the list and click the *Change* button.

Checking for special headers in messages

By default, MercuryD goes through the standard headers in incoming mail looking for local addresses: the fields it examines are: *"To", "Cc", "BCC"* and *"Received"*. MercuryD also records the Message-ID of every message it processes and usually will not attempt to deliver the same message twice.

Unfortunately, not all ISPs use POP3 mailbox schemes that will work with this approach: some use a non-standard header to record the address of the person for whom the message was actually intended - for example, "X-Deliver-To" is one that is seen from time to time. If your ISP uses a non-standard header to record the *delivery envelope address*, you can tell MercuryD about it using the *Headers* control: type in the name of the header Mercury should examine for local addresses (so, from our example above, you would type in *X-Deliver-To*). The field is not case-sensitive (so, *X-Deliver-To* and *X-DELIVER-TO* are treated as identical) and you can add the colon separator at the end of the name or not as you wish. If your ISP uses more than one special header to identify the local addressee, you can enter multiple header names in this field, separated by semi-colon characters (";"). You must not type any spaces in this field.

If you check the control labelled *Check only in these headers* then MercuryD will no longer examine the standard To, Cc, Bcc and Received headers for local addresses and will not discard duplicate messages. Use this control only if you are sure that your ISP *always* adds the header to your mail.

Your ISP will usually be able to tell you if they use a special header to identify the envelope address in your messages.

MercuryX Connection Scheduler

MercuryX can be used to schedule the operation of other protocol modules in the Mercury/32 system; it can also be told to execute certain commands before and after it activates the protocol modules under its command.

MercuryX is primarily intended for users operating in a dialup environment. Its operation is specially tailored to provide maximum flexibility with the shortest practical connection times.

Configuring MercuryX
Dialling considerations

Configuring the Mercury Connection Scheduler module

See also: Dialling considerations

Commands issued before and after connecting

MercuryX can be told to execute programs before it starts the Mercury Protocol modules and after it has shut them down during each rota period. You can enter any command in these fields, and the command can have a commandline. For maximum reliability, we recommend that you include a full path to the executable file in the commandline.

Run this command before starting If this field is not blank, MercuryX will attempt to run the command you specify before activating the Mercury protocol modules. Possible uses for this include invoking dialers, or loading network modules.

Wait until this process terminates before starting Mercury service processes If this control is checked, MercuryX will attempt to wait until the process in the *Run this command before starting* option has terminated before proceeding to activate the Mercury protocol modules. This option will work reliably with all Win32 applications, most Win16 applications, and some DOS applications. If this control is checked, MercuryX will not *wait X seconds* before starting the protocol modules - it will start them as soon as the process terminates.

Run this command after stopping If this field is not blank, Mercury will run this command after it has shut down all the protocol modules at the end of a rota cycle and waited the X seconds delay, if that is defined.

Before and after connections wait X seconds before running command If you enter a number in this field, MercuryX will wait that many seconds after invoking the startup command at the start of a rota cycle, and before invoking the shutdown command at the end of a rota cycle. If the *wait until this process terminates* control is checked, MercuryX will ignore this delay.

Use Win98/IE4 dialling functions Under Windows 98 and on systems where Internet Explorer v4.0 has been installed, Mercury can take advantage of dialling functions built-in to the operating system. If your system matches those described, you can tell Mercury to dial or hang up using these new functions by checking either of these controls. It is possible to "mix and match" options - so, you can use a command to dial, but the Win98 function to hang up if you have a need to do this.

Issue SMTP ETRN commands (RFC1985) to start remote queues Internet Standards Document RFC1985 defines a special command called *ETRN*, which can be issued by a dialup client to indicate that it is online and ready to receive mail. If your Internet Service Provider has a mail server that supports this command, then you can tell Mercury to issue it when it comes online - this is a useful way of scheduling when you will and will not receive mail from the Internet, but it requires the co-operation of your ISP. In order to use this option, you must also have either the MercuryC SMTP Client or the MercuryE SMTP Client Protocol Module installed in your copy of Mercury. For more information on ETRN and whether or not you can use it, please contact your Internet Service Provider (if they don't know what you mean when you mention "ETRN" or "RFC1985") then they probably don't support this option).

When using ETRN commands to start remote queues, you need to create a file called ETRN.DAT in the directory where Mercury is installed. Clicking the *Specify* button in this dialog will create this file if it does not exist, or will edit your existing ETRN.DAT. The format of the file is quite simple and is documented heavily in comments when it is created.

Allow queues to "drain" completely before shutting down connection This setting only affects the client modules, MercuryC (or MercuryE if you have installed that option instead) and MercuryD. When it is checked, once MercuryX reaches the end of a connection cycle, it will wait until the client processes enter an idle state by themselves before proceeding to shut down the connection. This means that you can tell MercuryX to use a one minute cycle once per hour, but all mail in the queue will still be sent even if it takes fifteen or twenty minutes: as soon as all mail in the queue has been processed, MercuryX will close down the connection. If this control is not checked, then MercuryX will ask the client modules to close down after the job they are currently processing is complete: in this case, mail can be left in the queue until the next cycle for processing.

Process control mode This setting determines how MercuryX should handle busy processes when it comes time to terminate a scheduled connection cycle. The setting that you should use depends very much on the way you use Mercury - essentially, it allows you to control how the Mercury client modules (MercuryC, MercuryE and MercuryD) are instructed to go offline, and also tells MercuryX how to handle the Mercury Server modules (MercuryS, MercuryP and MercuryH).

No control When this option is selected, all Mercury modules are instructed to go offline at the end of the connection cycle. Servers will go offline at once, and client modules will complete the job they are currently processing. Jobs that the client modules have not yet processed will remain in the queue until the next scheduled connection cycle.

Clients When this option is selected, MercuryX will wait until all clients indicate that they are idle (i.e, have no further jobs to process) and will then take them offline. The connection will not be terminated until all client modules indicate that they are idle and have been shut down. In this mode, Server modules are **not** instructed to go offline, but will continue listening for connections. This mode is useful if you use the client modules to connect to the outside world and the server modules to handle requests on your local area network. This mode is particularly suited to environments where dial-on-demand routing or ISDN is used.

Clients/servers When this option is selected, MercuryX will wait until all clients indicate that they are idle before shutting them down. It will also wait until all server modules are idle before terminating the connection, but will not actually instruct the server modules to go offline. This mode is intended to handle cases where the server modules may be able to accept connections from both the outside world while the connection is established, and from your local area network at other times.

Scheduling details

MercuryX allows you to create different schedules for each day of the week. Each day can have a *peak time* and an *off peak time* - the assumption is that peak time connections will be more frequent and will last longer than off-peak connections. Defining a scheduling *rota*, or a set of times for a given day, is simplicity itself - simply select the day from the drop-down control, then indicate the peak times in the *Between XXXX and XXXX* fields; once you have done this, indicate how often MercuryX should start the protocol modules and for how long, then do the same in the remaining fields for the off-peak times. Note that the connection cycle includes the activation time - so, if you tell MercuryX to start processing every five minutes for two minutes, it will begin a new connection three minutes after it shuts down the current cycle.

To copy the definition from another day into the current day, click the *Copy from* button and mark the day from which you wish to copy settings.

Dialling considerations

The process of dialling and hanging up intermittent Internet connections is one of the most frustrating and complex issues in the Windows environment.

Properly speaking, dialling and hanging up are functions of the Windows networking component that provides TCP/IP protocol support. This module, called WSOCK32.DLL, is a Microsoft-supplied component that is a built-in part of Windows. Unfortunately, it does not work correctly, and is unlikely ever to do so - Microsoft have shown no inclination to address its quite significant shortcomings. To explain why dialling and hanging up are system functions and not application functions, consider the situation where Mercury/32 is running at the same time as the user on the workstation is accessing the Internet using a web browser. If Mercury hangs up the connection, then the web browser will also be disconnected; similarly, if the user closes down the web browser and it hangs up the connection, Mercury will be cut off in mid-stream. Clearly, the system-level Network module, WSOCK32.DLL (which is used by both Mercury and the browser), is the only component in the system that knows how many tasks are active and hence when it is appropriate to close the connection.

At the time of writing, the Microsoft WSOCK32.DLL supplied with Windows 95, 98 and NT can initiate a dialup connection correctly, but will not correctly hang it up when it is idle.

Microsoft's failure to make WSOCK32.DLL handle dialling and hanging up correctly has meant that application developers have had to come up with their own solutions to the problem. In general, these solutions take two forms: writing calls to the Windows RAS subsystem to force dialling and hanging up, and using functions in a special Microsoft Internet Explorer 4.x module called WININET.DLL to force dialling and hanging up. Mercury/32 supports both these approaches.

1: Making RAS calls Under Windows NT, you can use the MercuryX scheduler module's command options to use the Windows NT *RASDIAL* utility to establish and disestablish connections. Alternatively, you can use a free version of RASDIAL written by Claudio Fahey, called RASDIAL95. This utility, which works under Windows 95, 98 and NT, is included with Mercury in the EXTRAS subdirectory of the directory where you installed Mercury/32. The utility is easy to use and has a comprehensive readme file describing its operation. We wish to offer our appreciation and thanks to Claudio for allowing us to include RASDIAL95 with Mercury/32.

2: Using WININET calls If you have installed Internet Explorer 4.0 on your system, or you are using Windows 98, then MercuryX can take advantage of special functions provided on these systems to establish and disestablish Internet connections. To enable this option, check one or both of the controls associated with it in the MercuryX configuration dialog.

MercuryH PH Directory Server

MercuryH is a directory services module - it accepts requests for addressbook lookups from other systems and returns lists of possible matches. The protocol used by MercuryH is called the *PH protocol*, also sometimes known as the *QI/CSO protocol*: it is widely-used on the Internet by mail clients such as Pegasus Mail and Qualcomm's Eudora. At the time this help file was written, an Internet standard was in the late stages of being formalised for the PH protocol - MercuryH implements that standard.

In order to fulfil queries, MercuryH uses a Pegasus Mail addressbook file, mapping the fields in the addressbook file onto the standard fields defined by the PH protocol. Any Pegasus Mail addressbook can be used by MercuryH, and you can use the import/export tools provided with Pegasus Mail to create and maintain addressbooks for MercuryH.

Configuring MercuryH

Configuring the Mercury PH Query Server

See also: Logging and session logging

Addressbook file: enter here the path to the Pegasus Mail addressbook file MercuryH should use when resolving queries. Pegasus Mail addressbooks consists of two files with the same name, one with the extension .PMR, the other with the extension .PM!. MercuryH only needs access to the .PMR file - enter the path to this file in this field.

MOTD file: The PH protocol allows you to define an arbitrary text message (referred to as a *Message Of The Day*, or *MOTD* file) that is sent in response to a PH *status* command. Enter the name of the text file MercuryH should send when it receives a status command here. The file should be plain text, with lines 60-70 characters in maximum length. This field is optional -you do not have to provide a MOTD file. You can perform simple text editing on your MOTD file by clicking the *Edit* button after you have entered the path.

Admin address: The PH *status* and *siteinfo* commands can advertise the address of an administrator to whom requests for support should be sent. If you wish to have a PH server administrator, enter his or her full e-mail address in this field. As with the *MOTD file* field, this field is optional.

TCP/IP Timeout the length of time in seconds that MercuryH should wait for data on a connection before assuming that the connection is no longer valid and aborting it.

IP Interface to use If your computer supports multiple IP interfaces, you can use this field to tell MercuryH which interface it should select when listening for connections: enter the interface as a dotted IP address in the general form *www.xxx.yyy.zzz*. As an example, your system may have one IP address assigned to a dialup PPP connection, and another, different IP address assigned to a local Ethernet network - you would enter here the interface MercuryH should use. If you leave this field blank, MercuryH will listen on all available interfaces. Unless you are *very* sure of what you are doing, or have been instructed by an ISP or network administrator, you should leave this field blank. If you change the IP interface in this field, you must restart Mercury before the new interface number will be used.

Listen on TCP/IP port By default, MercuryH listens for connections from the outside world on port 105, which is the standard reserved port for the PH Query protocol. In some cases, particularly when you are behind a firewall, you may wish to listen on an alternative port - enter the number of that port in this field. **If you change this field and save the dialog, you will need to exit and restart Mercury/32 before the change will take effect**.

Connection control

The *Connection Control* section allows you to place restrictions on the hosts from which MercuryH will accept connections. To add an entry to the list, type its IP address in the *IP Address* field and select either *Allow* or *Refuse*, then click *Add*. The digit 0 acts as a wildcard in a connection control entry, so adding an entry refusing access to 165.25.9.0 will cause MercuryH to refuse connections from any machine whose address's first three octets are 165.25.9. Note that there is an implicit rule “Allow 0.0.0.0” at the end of this list, so if an address “drops through” the list, it will be automatically accepted.

To edit a connection control entry, highlight it in the list, then click the *Change* button.

Support for Local Area Networks

Mercury supports Local Area Network architectures such as Novell NetWare via a system of plugin modules that give it access to the services offered by the Network. Using a Network plugin allows Mercury to take advantage of preconfigured mailboxes, built-in user databases (such as Novell NDS, or the NetWare Bindery) and generally reduces the administration load associated with maintaining mail users.

Mercury currently has separate plugin modules for Novell NetWare 3.x servers, using the Novell NetWare Bindery as a user database, and for Novell NetWare 4.x servers using Novell's NDS user database.

> Novell NetWare 3.x (Bindery mode) support
> Novell NetWare 4.x and 5.x (NDS mode) support

A network plugin will be supplied as a Windows DLL with the name "MN_xxx.DLL", where "xxx" is a three-character code describing the network supported by the module. For example, the DLL providing support for Novell NetWare 3.x is called MN_NW3.DLL. Exactly one Network support plugin can be installed at any time - Mercury/32 will use the first plugin it finds in the same directory as MERCURY.EXE.

When a Network plugin is installed, it may make configuration options available via the *Network support* option on the Mercury *Configuration* menu.

Novell NetWare 3.x (Bindery mode) support

Using this plugin (MN_NW3.DLL), Mercury can act as a mail server for one or many Novell Netware 3.11 or later file servers, using the file server's Bindery as a user database. In NetWare mode, Mercury is designed to interoperate closely with Pegasus Mail, a mail client by the same developer. It will deliver mail to the new mail location expected by Pegasus Mail, and Pegasus Mail can be configured to send mail to the Internet by placing it directly into Mercury's mail queue.

Under NetWare 3.x, all users are automatically given a mailbox directory in the file server's SYS:MAIL directory. Mercury is able to find and use this mailbox for delivering mail. When you create a user using the NetWare SYSCON command or by any similar means, the user's mailbox is automatically created and can be used immediately without any further configuration.

In NetWare mode, Mercury can use certain feature extensions, such as mail forwarding, which are manipulated using the Pegasus Mail PMGRANT program - see the Pegasus Mail documentation for more information on setting and using these features.

By default, in NetWare mode each user's address will be their NetWare login name @ the Internet name of the system. You can provide users with custom addresses that differ from this scheme by creating synonyms for them.

Multiple servers

Mercury can act as a mail transport for multiple Novell NetWare 3.x servers. Each file server must have an unambiguous domain name - so, you cannot have two servers both using the domain name "myhost.mydomain.com"; each server must have its own Internet name, unless you are willing to create aliases for every user who is to receive mail. When serving multiple servers, Mercury needs access to a privileged account on each server. You tell Mercury about the account and password it should use for each server using the *Network Support* option on the Mercury *Configuration* menu. The information is stored in an encrypted file called NETWARE3.DAT in the same directory as MERCURY.EXE.

If you wish to create a "group domain" (i.e, a situation where two separate file servers have the same Internet domain name) then you must create aliases for every user who is to receive mail. The alias should contain the user's e-mail address as the alias, and the real address must in the form SERVER/USER (the same as you would use to login to the server as that user). You must still have an Mercury access entry defined for every server using the *Network support* configuration option.

Bindery mode on NetWare 4.x and 5.x servers

Novell NetWare 4.x and 5.x file servers have support for a subset of NetWare 3.x's bindery, and the Mercury Bindery mode plugin can be used with those servers (although we would normally recommend that you consider using the NDS mode module). You should be aware, though, that NetWare 4.x and 5.x do not create the user's Bindery mode mailbox until the user logs into the server in bindery mode, using the command login /b, or by specifying a bindery login in the NetWare Windows client. It is important that your users login in Bindery mode before using the mail system if you run in Bindery mode on a NetWare 4.x or 5.x server, or otherwise mail delivery may fail.

In general, we do not recommend running in Bindery mode on NDS-based servers because of some long-standing weaknesses in the way Novell have implemented Bindery emulation: in particular, if you need to run the Novell DSREPAIR utility to rebuild or repair the NDS database, there is an appreciable chance that all your users' bindery IDs will be unilaterally reassigned, which will effectively result in all their mail being "lost" - in fact, the mail will still be there, but it will be stored in a directory different from the one NDS will report to Pegasus Mail as the user's mail directory. Recovering from this problem involves identifying the old mail directories and copying their contents into the appropriate new mail directory - a tedious and time-consuming process better avoided if possible.

Quite seriously - keep away from Bindery mode on NDS servers - run in NDS mode instead.

Novell NetWare 4.x/5.x/6.x (NDS mode) support

With the introduction of NDS under NetWare 4.x, Novell discarded the simplicity of NetWare 3.x and replaced it with a much more complex and administration-intensive system. They also removed most of the features of NetWare 3.x that allowed Pegasus Mail and Mercury to operate in an administration-free manner under that system.

Configuring Mercury and Pegasus Mail to work under NetWare NDS is slightly more complex than in Bindery mode. You need to create mailboxes for each user, and administration of issues such as synonyms and user features is also rather more complex, owing to the nature of NDS itself.

Mercury/32 is supplied with a NetWare NDS mode plugin called MN_NW4.DLL, suitable for use with NetWare 4.x, 5.x and 6.x file servers. This plugin is intended to complement the operation of Pegasus Mail in NDS mode, and assumes that you have already installed the Pegasus Mail NDS enabler, which is a standard part of the Pegasus Mail distribution package. This file contains a powerful NDS-mode configuration and maintenance utility called NCONFIG.EXE, which has been included with Mercury for your convenience as well. We recommend that you run NCONFIG and familiarize yourself with its online help before installing Mercury in NDS mode.

In NDS mode, Mercury takes advantage of NDS to provide an extremely flexible solution, at the cost of some extra administration and maintenance.

Address Synonyms

Many sites need to use custom address formats where the user's e-mail address is quite different from his actual user name on the system. For example, you may want your addresses to have the form *Firstname.Lastname@host.domain*. Simple aliasing won't allow you to do this kind of addressing easily - it will handle incoming mail well enough, but in order to get mail going out from your system to use the alternative address forms requires the co-operation of your mail client.

Pegasus Mail and Mercury/32 can combine to support alternative address forms like this (referred to as *synonyms*) using a special database called a synonym database: Mercury needs this database so it can work out the recipient for incoming mail, while Pegasus Mail needs it to work out what address it should write into outgoing messages. Using synonyms differs slightly depending on whether or not you are running in Novell NetWare mode:

** If both Pegasus Mail and Mercury are running in NetWare mode:*   create your synonyms using the Pegasus Mail PMGRANT (or NPMGRANT in NDS mode) commandline program, then create the Mercury synonym database using the CH_SYN.EXE (or NSYNONYM.EXE in NDS mode) commandline program. Copy the synonym database somewhere accessible by Mercury, and make sure the *Synonym database file* entry of the *Files/Directories* preferences page of the *Mercury core module* configuration dialog refers to that file. Pegasus Mail will automatically pick up each user's synonym from the NetWare user database.

Tip: The Pegasus Mail NDS mode configuration utility, NCONFIG.EXE, can greatly simplify the creation and maintenance of synonyms in NDS mode, both for Pegasus Mail and for Mercury.

** If either Pegasus Mail or Mercury is running in non-NetWare mode*, then you will need to follow these steps:

1: Create a *synonym source file*: this is a text file where each line has the form

```
synonym == username
```

So, on the left-hand side you place the user's e-mail address as you want it to appear in outgoing mail, and on the right hand side his or her local user name. Note that each line must begin hard against the left-hand margin.

2: Compile the source file using the FSYNONYM commandline program. FSYNONYM takes your input file and creates a synonym database file.

3: Copy the synonym database somewhere accessible by Mercury, and make sure the *Synonym database file* entry of the *Files/Directories* preferences page of the *Mercury core module* configuration dialog refers to that file.

4: Copy the synonym database file into the same directory as your copy of the Pegasus Mail executable file (either WINPMAIL.EXE or WINPM-32.EXE), making sure it is called SYNONYM.MER.

5: [Only required once] Use the Pegasus Mail option on the Mercury configuration menu to configure your copy of Pegasus Mail. This operation is only required if you have upgraded your copy of Mercury from an earlier version, and only needs to be done the one time: it creates a new version of PMGATE.SYS that instructs Pegasus Mail to use the synonym file if it exists.

Note: in order to use synonyms in non-NetWare mode, you must be using Pegasus Mail for Windows v3.01b or later. Earlier versions will not recognize the synonym database and will use the user's regular e-mail address instead.

Daemons (third-party extensions)

One of Mercury's most powerful and least visible features is its *Daemon interface* (*Daemon* is a term borrowed from the unix world meaning a *resident process*). This interface allows third-party developers to create extensions to Mercury for processing mail. Using the interface, a Daemon can accept mail for any or all addresses on the server, can send and parse complex MIME messages, and can generally perform nearly any imaginable task that you could do with e-mail.

The Daemon interface is extremely rich, but is not particularly difficult to program if you have a little experience writing Windows code. If you are interested in reading more about the Daemon interface, please see the files DAEMON.TXT and DAEMON.H in the RESOURCE subdirectory of the directory where you installed Mercury/32.

Logging and session logging

Most Mercury protocol modules can keep logs of the work they do. Logging comes in two forms, *General Logging*, which is simply a short record of each transaction maintained by the server modules with one transaction on each line in a file, and *Session Logging*, which is a byte-for-byte transcript of the entire exchange between a server and a client.

General logging is supported by MercuryS (the SMTP server), MercuryP (the POP3 server), MercuryC (the SMTP relay client), MercuryE (the full-delivery SMTP client), MercuryI (the IMAP4 server) and MercuryH (the PH Directory Services server). Each of these modules can be given a filename for a general log file: they will create this file as required and will write a single-line record describing each connection they process. Each module's general log file has a regular format that can be parsed by software tools to generate statistics and reporting. To turn general logging off, enter a blank filename. You can change the name of the general log file any time you wish in order to start a new log file. You must not attempt to use the same filename for the logging output of two different modules - this will result in improper behaviour and may result in Mercury crashing.

Special naming options for log files: Mercury allows you to use certain special characters in the names you give for log files: these special characters are replaced by Mercury each time it needs to open the log file, allowing you to "build" a filename based on the time and date. This is particularly useful for making log files "roll over" periodically. The special characters you can use in log file names are:

~Y	The year, expressed as 2 digits
~M	The month, expressed as 2 digits
~D	The day of the month (1..31), expressed as 2 digits
~W	The week of the year, starting from 0.

Example: to create a logfile for each week of the year, you might enter a path like this:

```
C:\MERCURY\LOGS\MERC~Y~W.LOG
```

Session logging is supported by MercuryS, MercuryP, MercuryH, MercuryC, MercuryE, MercuryI, and MercuryD. When session logging is turned on in a module, it will create one file per connection in a directory you specify. The file will contain a complete transcript of the data sent between the module and the remote machine to which it is connected. Session logging files are given numerically ascending filenames in the general form `TCPxxxx.yy`, where `xxxx` is a serial number and `yy` is an extension representing the module that created the log file. The extensions generated by the various modules are as follows:

`TCP*.MS`	Generated by the SMTP server, MercuryS
`TCP*.MC`	Generated by the SMTP client, MercuryC
`TCP*.ME`	Generated by the full delivery SMTP client, MercuryE
`TCP*.MH`	Generated by the PH server, MercuryH
`TCP*.MP`	Generated by the POP3 server, MercuryP
`TCP*.MD`	Generated by the POP3 client, MercuryD
`TCP*.MI`	Generated by the IMAP4 server, MercuryI

You can turn session logging on and off at any time using the *Enable session logging* checkbox in the relevant configuration dialog for the module. The setting of this checkbox is remembered by each module across sessions.

NOTES You should be aware of some important issues with session logging:

1: The *Session logging* parameter in each module's configuration dialog refers to a subdirectory, not to a filename. If the subdirectory does not exist, Mercury will create it as required. You can have each module create its session logs in a different subdirectory, or, if you prefer, you can have them all write their logs to the same directory.

2: The session log file includes all the data sent across the connection. So, the session log file for the arrival of an incoming mail message that is 5MB in size will be at least 5MB. Keep this in mind when you allow for disk space on your system - session log files can be large and can consume disk space at a startling rate.

3: Session logging slows connections down quite substantially. It has quite an impact on the rate at which mail is processed. You should typically regard session logging as a diagnostic tool rather than something that is left on permanently.

4: When session logging is turned on, all data sent across the link is included: in the case of the MercuryD POP3 client and the MercuryP POP3 server, this means that passwords will be stored in the session log files as plain text. This may create a security risk and should be carefully considered.

Internet name for this system

Enter here the Internet domain name for the machine on which Mercury is running. Mercury will use this information when forming certain addresses, such as the postmaster address. The name you enter here should be a fully-qualified domain name; if you are intending to use Mercury to provide mail services outside your immediate organization, the name you provide will need to be accessible in your Domain Name Server (DNS) system.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Mail Queue Directory

Mail queue directory, SMTP queue directory These entries control where Mercury should look for and place mail that is to be processed. The mail queue is where mail clients such as Pegasus Mail put messages for Mercury to process; Mercury also places jobs here on occasions, usually when generating autoreplies and mailing list mail. The SMTP queue is the location where the Mercury Core Module should place messages intended to be sent to the outside world by the MercuryC Client module. The mail queue and SMTP queues can be, and normally are the same.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

SMTP Queue Directory

Mail queue directory, SMTP queue directory   These entries control where Mercury should look for and place mail that is to be processed. The mail queue is where mail clients such as Pegasus Mail put messages for Mercury to process; Mercury also places jobs here on occasions, usually when generating autoreplies and mailing list mail. The SMTP queue is the location where the Mercury Core Module should place messages intended to be sent to the outside world by the MercuryC Client module. The mail queue and SMTP queues can be, and normally are the same.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Local Mailbox Directory Path

This entry is ignored if you are using a network support module: it tells Mercury how to locate your users' mailbox directories. The string is a standard pathname containing one of two special placeholders - either *~8* or *~N*. When Mercury uses the string to find the mailbox for a user, it replaces *~8* with the first eight characters of the user's name, or replaces *~N* with the user's whole username. If you are using Pegasus Mail v3.01d or earlier, or any 16-bit Pegasus Mail client as your mail client in conjunction with Mercury, you should not use the *~N* substitution - you should only use the *~8* version. If this 8-character restriction creates problems with usernames for you, you could consider defining synonyms for the names that are longer than 8 characters, or upgrading to a later version of Pegasus Mail.

NOTE: It is currently a restriction of Mercury that the ~8 or ~N placeholder must appear at the end of the path - so, *C:\PMAIL\~8* is legal, but *C:\PMAIL\~8\MAILBOX* is not.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Time Zone, Auto timezone

Time zone Enter here the timezone for your site, expressed as a plus or minus difference from GMT. So, if you are in Los Angeles and are currently at GMT - 9 hours, you would enter

-0900

in this field. Mercury will accept the so-called "vernacular" time zone format, such as PST and CST, but the use of these formats is no longer recommended on the Internet and we strongly advise you to avoid them, since their use makes it impossible for most mail programs to sort properly by date.

If the *Auto* control is checked, Mercury will work out the necessary timezone correction automatically using information supplied by the Operating System, and anything you enter in the *Timezone* field will be ignored.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Poll for new mail every x seconds

This setting controls how often the core module should check to see if there is mail waiting to be processed in the queue. We recommend that you do not set it below ten seconds for performance reasons.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Username of postmaster

Every system capable of receiving Internet mail must have a user called *postmaster*, to whom problem and status reports are sent. The postmaster account is usually an alias to a real user on your system, and this is the expectation within Mercury. Enter in this field the username of the user on the machine where Mercury is running who is to act as your postmaster. While it is permissible to have a non-local address as your postmaster address, we ***strongly*** recommend that you do not do this, since it can create real problems and mail loops when the remote machine is unreachable. This setting is mandatory - Mercury cannot run properly without it.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

For delivery failures return x lines of the message

When Mercury cannot deliver a message to a local user for whatever reason, it will invoke a template file you provide for delivery failures. One of the optional replacements that can be used in the delivery failure template file is a special substitution that sends a certain number of lines from the failed message. This configuration option controls how many lines of the message are returned when the special partial return substitution is encountered.

See also:

- The Mercury Core Module configuration
- Configuring Template Files

Press <Esc> to return to Mercury.

Broadcast notifications for normal mail

Mercury has special Network awareness modules that allow it to take advantage of certain specific features of some local area networks. One of the features that some networks (such as Novell NetWare) support is the transmission of a single-line broadcast message that appears on the target user's screen. If this control is checked and you are running Mercury on a network that supports broadcast messages, Mercury will send a short message to users when new mail arrives for them.

See also:

- The Mercury Core Module configuration
- Network Support in Mercury

Press <Esc> to return to Mercury.

Broadcast notifications for receipts

Mercury has special Network awareness modules that allow it to take advantage of certain specific features of some local area networks. One of the features that some networks (such as Novell NetWare) support is the transmission of a single-line broadcast message that appears on the target user's screen. If this control is checked and you are running Mercury on a network that supports broadcast messages, Mercury will send a broadcast message advising the arrival of mail messages that confirm reading or delivery.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Send copies of all errors to the postmaster

If this control is checked, Mercury will send a copy of all error reports it generates to the local postmaster as well as to the original sender of the message. This allows the postmaster the option of correcting addressing errors and other simple problems.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Change file ownership to recipient

As with broadcast notifications, some Network systems support the idea of file ownership, usually to calculate disk space usage. If your network supports this idea and this control is checked, then Mercury will attempt to change the ownership of all the messages it delivers so that the actual recipient owns the file.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Suppress validation of From field when processing mail

Mercury usually attempts to validate that the "From" field of all mail it delivers locally is legal. This can sometimes cause problems if you receive mail from sites that use broken or faulty mail programs; if this is the case, you can suppress the validity check Mercury performs by checking this control.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Hard to quit (exit only on Ctrl+File|Exit)

When this option is checked, Mercury will ignore all attempts to quit from it, and will minimize itself to the system tray instead. In order to quit from the program, choose "Exit" from the "File" menu while holding down the Ctrl key. This option is useful when Mercury is run on a server to prevent people from accidentally closing it down.

See also: The Mercury Core Module configuration

Press <Esc> to return to Mercury.

Mail filtering rules

See also: Configuring server-wide mail handling policies

Like its companion product, Pegasus Mail, Mercury allows you to perform automated processing of mail when messages matching particular conditions are met. You might use this, for example, to forward messages, or print them automatically for certain recipients, or to delete unwanted "spam". The Mercury core module is responsible for applying filtering rules, which it does as the first stage in delivering a message.

Using global rules
Creating and using general rules

You can trigger your rules (that is, define the set of conditions which must be true before the rule will be applied to a message) based on a number of criteria. You can select a number of types of rule trigger using the buttons at the left of the rule editor dialog; the possible trigger types include the following:

Matching on text in standard message headers
Matching on regular expressions
Matching based on list membership

There are also buttons that create rules that always trigger, rules that trigger on the size the message, rules that trigger depending on certain predetermined characteristics of the message, rules that are simply comments and have no effect on rule processing, and rules that are labels (see *Advanced formatting*, below, for more information on why you might want to use a label). Finally, you can use the *Not* button to change any rule so that it only triggers if the condition it describes is not met in the message.

A wide range of actions can be triggered by a rule. Rule processing continues until all rules have been applied, or until the message is moved to another user or deleted as the result of a rule.You can define multiple rules with the same trigger conditions to have multiple actions applied to the same message -- Mercury will apply them in the order in which they appear in the list. Make sure that any rules containing "Move" or "Delete" actions are the last you define for a particular trigger text, since rule processing on a message stops after these actions.

Actions that can be triggered by a rule

Advanced filtering - flow control and logical operations

Global rules

Global rules are a single set of rules applied to all your incoming and outgoing mail. You can create and maintain your global rule set using the *Edit global rules* option on the *Filtering rules* submenu of the *Configuration* menu.

Global rules allow you to create custom actions for all your outgoing mail - for instance, if you want to keep an audit trail of all the mail coming into and leaving your site, you could do this with a single global filtering rule.

The more global filtering rules you define, the longer it will take Mercury to process every message, even messages that do not trigger any rules. This usually isn't a problem, since electronic mail seldom needs to be immediate, but it is something to bear in mind when you create your rule sets.

If you find yourself needing to perform complex tasks that require large numbers of rules, you could possibly consider arranging to have a Mercury Daemon developed for you. Daemons are resident processes that can bring the full power of your PC to bear on specialized mail processing requirements: because they are compiled programs, they are inherenly very fast, but they do require some Windows programming.

Back to the main Mail Filtering help screen

Creating and using general rule sets

General rule sets are sets of filtering rules that can be attached to any e-mail address in your system via an alias. Unlike global rules, which are applied to all mail messages passing through the system, general rules are only applied when mail is delivered to the address to which they are attached.

You can create or maintain a general rule set using the options on the *Filtering rules* submenu of the *Configuration* menu. We recommend you use the extension .RUL to identify your rule sets, but this is not a hard-and-fast requirement of the program.

To use a general rule set you have created, you need to create an alias that tells Mercury to invoke the set. To do this, choose *Aliases* from the *Configuration* menu and create a new alias. For the *Alias*, type in whatever address is to be associated with the set, then for the *Real address*, type in the special string `FILTER:` followed by the filename of the filtering rule set.

Example: if you wanted to invoke the alias set `C:\MERCURY\ORDER.RUL` any time a mail message was sent to `orders@sales.com`, then you would create the following alias:

Alias	**`orders@sales.com`**
Real address:	**`filter:c:\mercury\order.rul`**

Using aliases to trigger your filtering rules ensures the security of the system, since only you can create an alias, and thus only you can choose which rule set is associated with that alias.

Filtering and real addresses Filtering is done before user delivery occurs, so if you wish, you can create a filtering alias that is the same as a real e-mail address on your system, and the filtering will occur instead of delivery. If you do this and still want the message delivered, you must make sure you place a *Copy to local user* rule that always triggers in your rule set, because Mercury does not otherwise deliver a message once it has been processed by the filtering rule subsystem.

Back to the main Mail Filtering help screen

Matching on text in predefined message headers

In the majority of cases, you will simply want to detect messages which contain particular addresses, or which have particular text in the subject field. Mercury provides an easy way of triggering rules based on conditions like this.

Click the Headers button at the left of the rule editor dialog, then simply check the fields in which you would like the test to be made and enter the text you would like to match in the "Contains this text" field. You can check as many of the six controls as you wish, although some are probably mutually exclusive (such as "From:" and "Subject:"). You should usually check at least one control, although it is permissible to check none (this is a useful way of disabling a rule without deleting it).

Exact matching: Mercury usually triggers the rule if any of the fields you check contains the trigger text anywhere; if you want Mercury to trigger only on an exact match, check the control labelled *Exact match*. Doing this means that the rule will only trigger if the header and the match are the same length and contain the same characters. The match is always case-insensitive -- this cannot be changed even for an exact match.

Examples:

If "Subject:" is checked and the trigger text is "SUBSCRIBE" then the rule will trigger if the subject is any of the following:

    Subscribe
    I want to subscribe to your list
    Notice to all subscribers

If "Subject:" is checked and *Message field must match this text exactly* is checked, then the rule will trigger only for the first of the examples shown above ("Subscribe")

Using regular expressions as rule triggers

Using regular expressions to trigger rules is more complicated than matching on predefined headers but is considerably more powerful, since it allows you to match on any message header, or even on the contents of the message body.

Clicking the *Expression* button in the rule editor creates a rule that searches using a regular expression. You can restrict the extent of the search using the scope controls - ***NOTE*** you should be careful when using either of the scopes which permit searching the message body, since this can dramatically increase the time it takes Mercury to process the message (sometimes by a factor of as much as 10 times or more). You need only have one action whose scope is the message body in your entire rule set to cause this increase in processing time, although subsequent rules in the message body scope do not introduce appreciable further delay.

Next, enter the regular expression you wish to use in the "Regular expression" field. Mercury recognizes the following metacharacters in expressions:

*	Match any number of any characters
?	Match any single character
+	Match one or more occurrence of the last character
[ ]	Encloses a group of characters to match. Ranges can be specified in the group using '-'.

All metacharacters can be used as many times as necessary. Regular expression searches are always case-insensitive. To search for literal occurrences of any of the characters *, ?, + or [, you must enclose them in group markers (so to search for a literal asterisk, enter [*]). Regular expressions do not cross line boundaries - you can only perform expression matching within individual lines in the message. If the first character of your expression is not '*', then matching must begin at the start of the line - so if you want to use a regular expression to find lines containing text at any position, you must use a leading '*' character (this is the reverse of matching on predefined headers, where the match is always open unless exact matching is specified).

Note: Unlike predefined header text matching, regular expression matches do not match substrings - they match entire lines. Because of this, if you want to detect a sequence of characters occurring anywhere in a line of text, you must surround the text with a pair of asterisks (*). This is the regular expression way of saying that any text can precede and follow the search text. *Example:* if you want to match on any line containing the string *offer* using an expression, you would need to enter the match text **offer**. Similarly, if you wanted to match any line starting with the word *Subject* and containing the word *offer* anywhere else on the line, you would need to enter the match text *Subject*offer**. If you omitted the second asterisk, you would be telling Mercury that the line would have to *end* with the word *offer*.

Examples:

MIME-VERSION:*[12]*
or MIME-Version: +[12]*
Matches "MIME-Version: 1.0"
but not "MIME-Version: 3.0"
(The '+' matches multiple successive spaces)

pmail.gen.nz
Matches any line containing "pmail.gen.nz"

Priority:*urgent
Triggers a rule for urgent mail

Filtering messages based on list membership

With this type of filtering rule, the rule will trigger if the sender's e-mail address can be located in a Mercury distribution list. When you create the rule, simply type in the name of the distribution list Mercury should search, and it will do the rest (note: you should enter the name of the list as it appears in your "list of lists" file, without a domain).

You can tell Mercury to scan a plain text file instead of a Mercury mailing list by entering the special character '@' followed by the full path to the file. This approach can be a useful way of maintaining traffic lists or kill files.

This type of rule has two major applications:

1: Creating "kill" files to catch "spam" (Unsolicited Commercial E-mail) from known addresses. When you receive an unsolicited spam message, you can add the sender's address to a list of known evildoers, then delete all future messages from that address using a single rule of this type.

2: Verifying that a person is a list member: if you offer services that are triggered by filtering rules (for instance, if you return product information or encryption keys in response to automated messages), then you may wish to verify that the person sending the request is actually a member of a list of authorised people before providing the service. You can use a rule of this type to determine whether or not the person is authorised based on their membership of a list.

Advanced option

You can create an entry in your target distribution list that matches any address from a single domain by editing the list manually and adding a line exactly like this:

\MATCH *@domain.com

into the list. The "\" character must appear hard against the left margin of the file.

Example: to suppress all mail from any address within the domain "`bigdeals.com`", you would add the following line to your distribution list:

```
\MATCH *@bigdeals.com
```

Actions that can be triggered by rules

When a rule's trigger text matches text in the message, its action is performed. Actions are selected from the list in the Rule Editor - selecting an action may result in you being prompted for more information, such as selecting a destination folder, or entering a file name; you can always change the parameters for a rule at any time by clicking the *Set* button.

Rules are processed until a rule is triggered whose action results in the message being either deleted or moved to another user. Because of this, you should always take care to ensure that rules containing Move or Delete actions are the last ones you define for a particular trigger text.

You can also force rule processing to stop for a specific message by using the *Exit this rule set* action.

Most of the rule actions are pretty obvious, but the first two, *Copy to another user* and *Move to another user* require a little explanation. These rules make exact duplicates of the message in the mailbox of another local user - they bypass any autoforwarding or autoreply mechanisms in place, and unlike forwarding, do not change the message in any way at all. These actions are primarily intended for sites that need audit trails of the messages they send and receive. Note that you can enter a directory into which the messages should be placed instead of a user by entering the special character '@' followed by the full path to the directory (which must exist). Mercury ensures that no filename collisions occur in the directory where the messages are placed in any event.

The *Logical AND operator* rule action can be used to create groups of rules where all the rules must be triggered before the final action is applied; this process is called *logical operation* - click here for more information on logical operations.

Advanced filtering - flow control and logical operations

This section covers specialised uses of mail filtering and is intended mainly for advanced users. Please ensure you have read all the other sections on mail filtering before attempting to use the information in this topic.

Flow control

Many times, you may find that there are certain groups of rules that you want to apply repeatedly in a rule set, or that you want to have more control over the order in which rules are processed. This concept is called *flow control*, and Mercury provides six rule actions to support it - skip, exit, labels, call/return and goto.

Using flow control

Logical operations

Often, you may only want to apply a rule to a message if all of a number of conditions are matched, or if any one or more of a number of conditions apply. This kind of operation is known as a *logical operation* (it is also known by the technical name *Boolean operation*). Mercury implements logical operations by a combination of rule order and flow control structures. Before reading about logical operations, we strongly suggest you read the section above on flow control.

Creating logical operations

Using flow control in your rule sets

Many times, you may find that there are certain groups of rules that you want to apply repeatedly in a rule set, or that you want to have more control over the order in which rules are processed. This concept is called *flow control*, and Mercury provides six rule actions to support it - skip, exit, labels, call/return and goto.

Skip The simplest flow control operator is the *Skip next rule* action: when a rule triggers and this action is indicated, Mercury will skip over the next rule in the list without testing or applying it. You can use this as a way of handling exceptions to a general rule - for instance, imagine that you want to delete all messages where the subject contains the phrase *free offer*, except when that message comes from the address *support@pmail.gen.nz* - you would add the following two rules to your rule set:

```
If "From" field contains "support@pmail.gen.nz", then skip next rule
If "Subject" field contains "free offer", then delete message
```

Exit When a rule triggers that has the action *exit this rule set*, all rule processing for the current message terminates at once - no more rules are examined or actioned. The primary use of this action is to separate *subroutines*, or groups of rules that you access via *call label* actions, from the main body of your rule set.

Labels A label is simply a name you can add to any line in your rule set. Labels are used by return and goto actions (see below) to transfer processing to a different location in the rule set. Labels can appear anywhere in the rule set - when calling or going to a label, you can go either forwards or backwards. Labels are simply a textual name - you can use any text or letters you wish up to 45 characters in length. Labels are the only passive item in a rule set - on their own, they do absolutely nothing, and any match conditions they contain are ignored.

Calls and returns If you have defined a label in your rule set, you can call it at any time by defining a rule with the *Call label* action. If the rule triggers, processing of the rule set will transfer to the first rule after the label you name and will continue until either there are no more rules (in which case rule processing terminates), or a rule triggers that has the *Return from call* action (in which case processing resumes at the rule following the one which initiated the call).

Gotos A goto is like a call, in that it simply transfers processing to a label anywhere else in the rule set. The difference is that you cannot return from a goto - the transfer of processing is final. Gotos are primarily useful when implementing complex logical operations.

Example
In this example, we will implement a list server where the user can subscribe to lists on your system by sending you a message containing the subject line *subscribe <listname>*: to do this, we look for the word *subscribe* in the subject line, and if we find it, we call a label that handles the list names. Note the use of expressions in this example to isolate complex cases.

```
If data matches expression "Subject: +Subscribe*" then call subscriptions
[… other rules can appear here…]
If data matches "*" then exit from this rule set

Label subscriptions
If data matches expression "*interest-list* then add sender to list "ilist"
If data matches expression "*beta-testers*" then add sender to list "beta"
[… other subscription cases could appear here…]
If data matches "*" then return from call
```

Some things to note about this example: firstly, notice the way the subject line is detected: the regular expression Subject: +Subscribe* detects a line that starts with the letters subject:, followed by one or more spaces (the + operator) then the letters subscribe. Note that the Return from call rule matches on a single * - this is the way you tell a rule to match on any text, and in this case ensures that the return will always be triggered. Finally, notice the use of an *Exit from this rule set* action to ensure that we don't drop into the subscription processing rules other than at the proper times.

Creating logical operations in your rule sets

Often, you may only want to apply a rule to a message if all of a number of conditions are matched, or if any one or more of a number of conditions apply. This kind of operation is known as a *logical operation* (it is also known by the technical name *Boolean operation*). Mercury implements logical operations by a combination of rule order and flow control structures. Before reading about logical operations, we strongly suggest you read the section on flow control.

Tip When using logical operations in rule sets, it is very important to remember that rules are always applied to the message in the order in which they appear in the rule list editor, starting at the top of the list and working through to the bottom.

Applying a rule when any of several conditions is met (logical OR)

The simplest logical operation you can create in a rule set is that where an action is applied if one or more conditions is satisfied (i.e, condition1 OR condition2 OR condition3 and so on). You can create this kind of operation simply by creating multiple rules matching on different conditions, all executing the same action. For example, say you want to highlight a message in Magenta in your folder if the subject contains the word order, or if the subject contains the word invoice or if the To: field contains the address orders@foo.bar.com - you would add the following three rules to your rule set

```
If "Subject" field contains "order" then highlight message in magenta
If "Subject" field contains "invoice" then highlight message in magenta
If "To" field contains "orders@foo.bar.com" then highlight message in magenta
```

Notice that the action is the same in each case. In cases where repeated application of the rule action might not be desirable (for instance, copying messages to another user, in which case that user could get multiple copies of the message), more complex combinations of goto and call statements can be used to achieve the same effect - for example, like this:

```
If "Subject" field contains "order" then goto label "copy message"
If "Subject" field contains "invoice" then goto label "copy message"
If "To" field contains "orders@foo.bar.com" then goto label "copy message"
label "Next label"
[… other rules …]

Label "copy message"
If "Subject" contains "*" then copy message to "orders"
goto label "next label"
```

In this example, any of the conditions will transfer control to the rule that actually copies the message, which in turn immediately transfers control to the first rule after the group, so you will only ever get one copy of the message. Note the use of the single asterisk (*) in the copying rule: a single asterisk matches any text and in this case ensures that the rule will always trigger.

Applying a rule only if all specified conditions are met (logical AND)

Mercury offers several ways ways of applying a rule or rules only if all of a set of conditons apply. Much the easiest is to create a set of rules all of which have the *Logical AND* action; when you do this, Mercury will apply the action defined in the last rule provided all the preceding rules with the *Logical AND* action trigger properly.

There are two other ways to create logical AND operations that may be useful in some cases. The simpler form allows you to match exactly two conditions, using the *Skip next rule* action. To do this, you simply use the *Skip next rule* action on the first rule in the pair if the data does NOT match the first condition, then apply the action you want in the second rule only if the second rule DOES match the second condition. For instance, in the following example, we want to delete the message only if the subject field contains free offer and the from field contains cyberpromo.com.

```
If "Subject" field does not contain "free offer" then skip next rule
If "From" field contains "cyberpromo.com" then delete message
```

The more complex approach to matching multiple conditions depends on using a call statement to transfer to a group of rules where each rule returns if it does not contain the required text. This approach requires more setup, but allows you to match on an unlimited number of conditions. For instance, say we want to play a sound when we get a message from anyone at compuserve.com where the subject line contains the word Transylvania but not the word vampire, and the To: field contains the address foo@bar.com: to achieve this, we would create the following rules in our rule set -

```
If "From" field contains "compuserve.com" then call label next-test
[… other normal rules are here…]

Label next-test
If "Subject" field does not contain "Transylvania" then return from call
If "Subject" field contains "vampire" then return from call
If "To" field does not contain "foo@bar.com" then return from call
If "Subject" field contains "*" then play sound "tada.wav"
If "Subject" field contains "*" then return from call
```

Note the use of a single asterisk (*) in the last two rules to match any text: this ensures that the rule actions will always be applied. In order to get to the rule that plays the sound, all the rules before it must have failed to match.

Regular expressions: a *regular expression* is a description of an expression which permits matching based on patterns rather than on exact correspondence. If you have ever used a DOS command like "DEL *.BAK" then you have already used a regular expression -- what this command is really saying is "Delete all files where the name contains anything and the extension is BAK".

When you compose a regular expression, you use a combination of *literal characters* which must be present (such as the "BAK" in the command above) and special *metacharacters* (also called *wildcard characters*) which can match varying numbers of characters, or characters in specific groups only. As an example, MS-DOS supports two metacharacters - *, which means "match any number of any characters" and ?, which means "match any single character in this position".

MercuryE, Full SMTP Delivery Client Module

See also: Logging and session logging
The MercuryX Scheduling module and ETRN

Unlike the MercuryC SMTP module, which relies on using a single "smart" host to relay all outgoing mail for it, the MercuryE module is capable of delivering mail directly to the recipient's mail server. It does this by using Internet name resolution functions to work out the address of the proper system, then connects to it directly to perform delivery.

MercuryE is very easy to configure: all it needs is the address of a name server it can use to perform domain name lookups, which it will typically obtain from the Windows registry. Apart from that, its operation is largely automatic. It periodically polls the mail queue for outgoing jobs, handling them as required. A single job may require connections to multiple hosts, and individual addresses may have problems within the message - MercuryE handles all these conditions gracefully and predictably.

Name Servers By default, MercuryE will ask Windows for the address of the name server it should use to resolve mail addresses. If you operate in a dialup environment, or if the machine where Mercury runs gets its configuration information via DHCP, then MercuryE may not be able to retrieve this information properly. In situations like this, you should enter the IP addresses of the name servers MercuryE should use in this field, separated by commas.

Maximum simultaneous delivery threads This setting controls the number of messages Mercury will attempt to deliver simultaneously. The larger the number you enter, the more heavily loaded the system will become. In general, we recommend the default setting of 10 for normal use. In systems with heavy mailing list usage, however, there may be considerable value in setting it higher - for instance, to 30 or 40. You should not set this value above 100.

Honour requests for transcript generation MercuryE can be asked to produce a transcript of successful mail deliveries, by adding an *"X-Transcript-To: <address>"* header to your messages. When a transcript is requested, MercuryE will send a message containing a transcript of the message delivery to the address referenced in the X-Transcript-To header. While not an absolute proof of delivery, a transcript can provide significant evidence that your message was actually delivered. If you are using Pegasus Mail, you can use the *Custom headers* control in the *Special* view of the message editor to add an X-Transcript-To header to your mail. Transcript processing introduces a small overhead to message processing, so if you would prefer MercuryE not to provide transcript processing services, make sure the *Honour requests...* control is unchecked. Transcripts are only available (and only meaningful) if you use MercuryE - they are not supported by the MercuryC module.

MercuryC or MercuryE?

The decision about which Mercury SMTP client you should use depends on a couple of factors:

1: Do you use a dialup connection? If you connect to the Internet using a dialup connection, you will normally use the MercuryC module because it minimizes the time you are connected. You can, however, use MercuryE on a dialup connection if doing so fits your needs; when using a dialup connection, you may have to tell MercuryE to use a specific name server, via its configuration dialog, rather than relying on Windows to provide one correctly.

2: Are you behind a firewall? If your local area network is protected by a firewall, you will often need to use the MercuryC module because systems outside your firewall will probably not be able to contact your system directly.

Technical support

If you have a problem with Mercury/32, the following resources are available to you.

1: Formal technical support If you have purchased a Mercury Support Subscription, then you are entitled to obtain formal technical support directly from the developer of the program. Send a detailed description of your question or problem to the address supplied to you in the welcome credentials mail message you received from us at the time your order was processed.

2: Our knowledgebase We operate a searchable online knowledgebase that is regularly updated with information and issues relating to Pegasus Mail and Mercury. Anyone may access our knowledgebase by visiting:

http://kbase.pmail.gen.nz

3: Our web page Copies of our Frequently-Asked-Question lists, and downloads of current utilities and versions of both Mercury and Pegasus Mail are available from our web site,

`http://www.pmail.com`

4: FAQ (Frequently-Asked-Question) lists Anyone may access lists of common questions and answers concerning Mercury/32 by sending any e-mail message to

`faq-merc32@pmail.gen.nz`

The FAQ file will be returned to you automatically by our mail server.

5: User groups Various public mailing lists and user groups exist that offer valuable resources for Mercury users. To retrieve a list of available lists and groups, and instructions on joining them, send any e-mail message to

`faq-usergroups@pmail.gen.nz`

The FAQ file will be returned to you automatically by our mail server.

Advanced configuration

SEE ALSO: CORE MODULE CONFIGURATION

The items in this page control some of the more advanced Mercury features.

Advanced configuration settings

Allow file-based forwarding specification using FORWARD files Autoforwarding is normally a restricted feature in Mercury and Pegasus Mail, controlled in NetWare mode by the system administrator. In non-NetWare modes, and in NetWare mode if the administrator wishes to open the feature up, you can check this control to enable an alternative autoforwarding feature. When this control is enabled, Mercury will look for a file called FORWARD in the user's new mail directory. If it finds a FORWARD file, it will open it and examine it for lines specifying how forwarding should be done. For the format of the FORWARD file, click here. Enabling this feature can create a security issue, since it is possible for another user with write access to a user's new mail directory to create a "fake" FORWARD file and forward the user's mail without his or her knowledge. In environments where a little trust is possible, however, it's a very useful feature.

Suppress automatic replies system-wide If you do not want your system to send automatic replies, even if your users have attempted to enable the feature, check this control. Note that this only suppresses standard user autoreplies - it does not affect messages generated by rules, by the built-in mail server, or automatically generated notifications.

Address auto-recognition settings

The controls in this group allow you to configure Mercury to recognize certain common Internet address formats based on the names of your users. When any of these controls is enabled, it tells Mercury that it should perform some extra comparisons when trying to work out if an address is local, by comparing the address with your local users' names, as they appear in the Mercury "Manage local users" dialog.

In the examples below, we use *myname.com* to represent your Internet mail domain.

Automatically recognize "Firstname.Lastname" forms This is one of the most common Internet addressing formats: if you have a user whose username is peter and whose full name is Peter Smith, then his e-mail address is both *peter@myname.com* and *Peter.Smith@myname.com*.

Automatically recognize Initial.Lastname" forms This is like the previous setting, but it combines your user's Initials and surname. So, given our hypothetical Peter Smith user, with this setting enabled, his address is both *peter@myname.com* and *P.Smith@myname.com*.

Recognize variants using either periods or underscores This setting combines with either of the previous two settings, by allowing either an underscore character or a period to appear in place of spaces in your users' addresses. So, if all three controls in this group were checked, our Peter Smith user could be mailed using any of the following addresses:

peter@myname.com
Peter.Smith@myname.com
P.Smith@myname.com
P_Smith@myname.com
Peter_Smith@myname.com

All of these settings are smart enough to handle multiple names or initials. So, if our Peter Smith was actually Peter O.Smith, then his addresses would be P.O.Smith, Peter.O.Smith or whatever.

It is up to you to ensure that your usernames are sufficiently distinct from each other if you use these settings - Mercury will use the first valid match it can find. So, if you have both Peter Smith and Patricia Smith on your system, and you use the *Initial.Lastname* format, you should make sure you enter a middle initial for at least on of the two so their addresses become distinct.

Allow the use of "+" forms in addresses to carry user-specified data   If this control is checked, then Mercury will support a specialized address variation where the user can append data to his address using a "+" sign, and Mercury will still recognize it as local. An example might serve to explain how this could be useful: user *bob@mydomain.com* has subscribed to the *Useful widgets* mailing list and wants to use the filtering features of his mail program to move all mail from that list into a folder automatically... To do this, he subscribes to the list using the following address: *bob+widgets@mydomain.com*   When the mailing list software sends the message to Mercury, Mercury ignores the *"+widgets"* and correctly identifies the message as being for *"bob"*, delivering it accordingly. Bob's mail program can then filter on the address and when it sees the *"+widgets"*, recognize that the message should be moved into the *Useful widgets* folder.

Daily maintenance settings

Once a day, Mercury performs a certain number of routine maintenance tasks (such as handling automatic expirations in mailing lists). These settings allow you to control when that maintenance should occur, and to force Mercury to perform a graceful restart each day.

Time to perform daily maintenance tasks   Just what the title suggests. Enter the time you want Mercury to perform its tasks in 24-hour format - so, 22:00 for 10pm. Mercury will perform its maintenance tasks on the first poll cycle after the specified time.

Exit and restart each day after performing daily maintenance   In rare instances, you may wish to restart Mercury each day (for instance, your network connection may need to be relinquished periodically in order to keep it alive). If you check this control, then Mercury will perform a graceful shutdown after it has completed its daily maintenance tasks. If you are using the Mercury loader program, LOADER.EXE, to run it, the loader will restart Mercury after a three second delay.

Format of FORWARD files

FORWARD files control automatic forwarding when this feature is enabled. The file can contain the following lines:

```
Forward-To : <address>
Deliver-Also : Y|N
```

The *Forward-To* line indicates the address to which the message should be sent. You can enter any single valid local or Internet address in this line. You may have multiple Forward-To lines in the file, in which case the message will be forwarded to the address in each line encountered.

The *Deliver-Also* keyword controls whether or not the message should be delivered locally as well as being forwarded. If set to Y, a copy of the message will be delivered to the user's mailbox even if it is forwarded to another address.

Loading protocol modules

Mercury/32 has plugin modules called Protocol Modules, that provide support for most commonly-used Internet protocols related to electronic mail, but you won't necessarily need to use all of these protocols. So, for example, if you have a permanent Internet connection and a registered domain name, it's unlikely that you'll use the MercuryD distributing POP3 server.

The controls in this dialog allow you to choose which protocol modules Mercury/32 should load at startup. You can select any combination of protocol modules, with one restriction - you can load either the MercuryC relay SMTP client or the MercuryE end-to-end delivery SMTP client, but not both. Selecting one of these two modules will deselect the other automatically.

Any changes you make in this dialog will not take effect until the next time Mercury restarts.

General list settings

See also: Mailing lists, overview

List title Every list must have a title -- a descriptive name that Mercury will use to form the "from" field of messages sent to the list. Try to keep the title short and descriptive and avoid international or accented characters. On rare occasions, you may wish to include address details as part of the title (Mercury usually adds the proper address to the list title automatically): in this case, you should ensure that the address you enter conforms to RFC822 addressing rules and includes a fully-qualified domain address appropriate for the list, then check the control labelled *Is a full address* next to the list title. *NOTE:* This feature is extremely specialised and is not normally required; because it can cause problems with mail delivery, we recommend that you only use it if you are very sure of what you are doing.

Membership file The name of a file where Mercury should store the membership information for the mailing list. Mercury will automatically add the extension .MLF to whatever path you enter here. You *must* enter a filename with a full path in this field.

Archive file Mercury can save copies of every message sent to a list in an archive file. If you want it to do this, enter an archive filename here. The filename must be a legal filename and can include a path if you want to create it in a specific directory. You can use the following special characters in the filename:

~Y	The year, expressed as two digits
~M	The month, expressed as two digits
~D	The day, expressed as two digits
~W	The week of the year, expressed as two digits.

Using these substitution characters allows you to create sequences of archive files matching specific periods of activity.

Allow membership enumeration via the mail server REVIEW command The Mercury mail server has a REVIEW command that can be used to list the members currently subscribed to a mailing list. The REVIEW command will only be processed for any given list if this control is checked in that lists's definition- if it is unchecked, the command will return an error.

Conceal this list from the Mai server's LIST command The Mercury Mail Server, MAISER, has a command (LIST) that returns all the lists serviced by the running copy of Mercury. If you do not want a list to appear in responses to this command, check this control and it will not be included in the summary returned by the mail server.

List owners (moderators) A mailing list can have one or more *moderators*, who are effectively managers for the list. Moderators have full control over the membership and settings of a list, and you can also configure a list so that only moderators may actually send mail to its membership: when you configure a list this way, then the list is said to be *moderated*. The intention of a moderated mailing list is that mail must be submitted to the moderator, who will then decided if it should be distributed. Note that a list can have moderators without being a moderated list - that is, a list can have supervisors, but can still distribute mail sent from the general public. A list need not have any moderators if you wish, and it is permissible for a moderator not to be a member of the list. To create or change the moderators for the list, use the *Add new moderator*, *Change selection* and *Remove selection* buttons next to the moderator list.

List access settings

See also: Mailing lists, overview

Settings controlling subscription to the list:

Welcome files, farewell files You can create simple text files that are automatically sent when someone subscribes or unsubscribes from a mailing list. These files should usually contain instructions for unsubscribing and resubscribing to the list, but can contain anything you feel is appropriate. Enter the filenames for the *Welcome* and *Farewell* files in their respective fields in the mailing list definition dialog. You can edit the files once you have entered a name, by clicking the *Edit* button next to the field. *Standard files:* Mercury comes pre-installed with a set of standard welcome and farewell files that it will use by default - these standard files will cover the majority of common list types and applications. You can instruct Mercury to use its standard set of files by typing the word *Standard* in either or both of these fields. You cannot edit the standard files using the Edit buttons, but if you wish, you can change them manually using a text editor - look for files called STDSUB_?.MER and STDFW_?.MER in the directory where Mercury is installed.

Confirmation file You can configure your list so that subscriptions to it will only be accepted when the subscriber replies to a confirmation message sent out by the mail server on receipt of the original request. Doing this reduces the likelihood of someone subscribing to the list with a mis-typed or invalid e-mail address, since they will never get the confirmation request if the address is unreachable. Mercury has a default confirmation request text (contained in a file called CONFIRMS.MER) but you can also provide your own text by entering a filename here and clicking the *Edit* button next to the field. Your confirmation text should include detailed instructions on how the confirmation should be sent - see the base CONFIRMS.MER file for a recommended text.

Allow public subscription (anyone may subscribe) If this control is checked, then anyone may subscribe to the list by sending a *SUBSCRIBE* command to the Mercury/32 Mail Server. If this control is not checked, then only list moderators may add subscribers, using the mail server's *ADD* command.

Require confirmation from subscribers before activating new subscriptions If this control is checked, new subscribers will be required to reply to a confirmation request before being added to the list. See *Confirmation file*, above, for more details.

Automatically set new subscribers to digest mode if available If this control is checked and digest support is available for the list, new subscribers will automatically be subscribed to the list in digest mode.

Subscriptions expire automatically after x days If you enter a non-zero value in this field, then subscriptions to your mailing list will have a limited life span: "x" days after the subscription is activated, the subscriber will be removed from the list. When a user's subscription to a list expires, Mercury sends a short message advising the user that it has happened. If the list is moderated, Mercury sends the file AUTOEXPM.MER, and if the list is not moderated, it sends the file AUTOEXP.MER. You can modify these files if you wish, but they are used for all lists maintained by Mercury.

Settings controlling submission of mail to the list

Mail can be submitted to the list by This group of controls determines who is permitted to send mail to the list for redistribution. If *Anyone* is checked, then Mercury simply distributes any mail sent to the list without performing any checks. If *Subscribers/Moderators* is checked, then Mercury will only permit mail sent by a current subscriber or list moderator to be distributed to the list. If *Moderators only* is checked, then the list is considered to be *fully-moderated*, and only mail sent by people whose addresses currently appear in the moderator list will be distributed to the list. Note that a list moderator is not required to be a current subscriber to the list.

Size limit If you enter a non-zero value in this field, then only messages smaller than that number of bytes can be distributed to the list - messages larger than the limit will be returned to the sender with an error.

Automatically redirect unauthorised postings to the primary moderator If this control is checked, Mercury will forward unauthorised list postings to the first moderator in the moderator list (the *primary moderator*). Clearly, this setting has no effect if mail can be submitted to the list by anyone. The moderator will receive the message in a special multipart format that preserves the original message intact - using most competent mail clients, it should be easy for the moderator to forward the original message with or without changes to the list to allow it to be distributed.

Require an X-Password field with a valid password for submissions When this control is checked, Mercury will only accept messages for submission to the list if it can find an *X-Password* header field in the message's headers, and that header contains either a valid moderator password or a valid subscriber password (see below). Many mail clients will permit you to add custom headers to messages - in Pegasus Mail, for instance, you would add the *X-Password* field in the *Special* view of the Message Editor window.

Subscription passwords Mercury allows you to require that subscribers provide a password in order to subscribe to a list. Password-protected subscription of this type is very useful if you want to prevent people from casually subscribing to a list but do not want to force a moderator to become involved with every subscription. You will typically provide the subscription password by some external means, such as via a reference on a web page, or by e-mail.

Moderator passwords A password or passwords can be associated with a mailing list. When this is done, commands that can only be issued by moderators will need the password before they can be processed. The password is supplied by issuing a PASSWORD command in the message to the mail server at some point in the message prior to the command that needs it. So, if you have set the password fubar on the list called vobis on your server and a moderator wants to add a user to that list, he or she will need to send something like this:

```
password fubar
add vobis user@host.domain
```

Subscriber passwords Subscriber passwords are like Moderator passwords, but are only used in association with X-Password header lines to authorize messages for delivery to the mailing list.

Single passwords vs password files For subscription, moderator and subscriber passwords, you can provide either a single password, or a file of passwords. If you provide a file of passwords, then any password in the file can be used to gain access to the feature it controls. This latter approach allows you to give each moderator or subscriber his own password, and revoke it without affecting other users in the event that he or she ceases to need access.

List distribution settings

See also: Mailing lists, overview

Headers and URLs If the *Generate helper URL headers* option is turned on, Mercury will add specially-formatted headers to messages distributed to the list which will permit compliant mail programs like Pegasus Mail to perform automatic subscription management for the user. If you have a web page that describes the operation of the mailing list, enter its URL in the *Help URL* field. Using Helper headers and URLs can result in a considerable improvement in the usability and friendliness of your lists. For more information on the format and function of Helper URLs, please see Internet Standards Document RFC2369.

Web-based (using MercuryB) If you have installed and enabled Mercury's HTTP server, MercuryB, then you can instruct Mercury to generate helper URLs that refer to the "mlss" (Mailing List Subscriber Services) service run by MercuryB. When this control is checked, Mercury will ask MercuryB for the proper URL and port for access to the mlss service module, and will use that in the helper URL headers instead of maiser commands. Most users find it much easier to manage their settings using a web page than to send commands to a mail server, so this option is recommended unless you have specific reasons not to use it. If you enable this command but MercuryB is not loaded, Mercury will not generate those helper URL headers that depend on it.

List signatures A list signature is a small text file that is automatically appended to the end of every message distributed to the list membership. In digest mode, the list signature is appended once as a separate message at the end of the digest. The first line of the list signature must contain the text to be placed in the "Subject" field of the digest part; the remainder of the signature can be whatever text you wish to include. The "subject" line is ignored for non-digest subscribers. List signatures are usually used to include information on unsubscribing from the list, or on contacting the list moderator. They are optional - if you do not want to define a list signature, leave this field blank. ***Remember:*** the first line of the message is the digest subject line - you must leave a blank line there if your list does not support digests.

Force replies to go to the list (using the reply-to header) If you check this control, Mercury will place the mailing list's address in the reply-to field of all messages distributed to the list. This will cause any competent mail client to send replies to the list instead of to the person who originally sent the message.

Modify subject lines using this string Any text you enter in this field will be inserted at the start of the subject field (or at the end if the *Suffix* control is checked) of all mail distributed to the list. This feature is primarily intended for the comfort of MajorDomo users, and for the benefit of mail programs with rudimentary filtering capabilities. You can enter any text you want in this field - the MajorDomo convention is usually to enter the list topic in square brackets... In any case, try to keep whatever you enter short. For replies to the list, where the subject line already contains the tag text, Mercury will strip it out and re-insert it after any occurrences of "Re", "Re:" or "Re[x]:" at the start of the subject field, ensuring consistent placement without affecting clients that can thread-sort by subject.

Enable Pegasus Mail-compatible encryption If the subscribers to your mailing list all use Pegasus Mail, then you can encrypt the messages you send to the list by checking this control. Whatever key you supply will be used to encrypt the messages, and your subscribers must know that password in order to read the messages. Mail generated with this option turned on is only readable using Pegasus Mail.

Explode submissions For large lists, it can be significantly more efficient to send the message out to several chunks of the subscription list instead of simply generating one large message, since doing so allows multiple SMTP processes to handle the mail at the same time. If you enter a value here, Mercury will "explode" messages sent to the list into that number of outgoing jobs. This setting can have a dramatic impact on list delivery if you are using the MercuryE SMTP end-to-end delivery protocol module. You cannot explode a submission into more than 20 jobs.

Digest support Mercury has comprehensive support for mail digests. To enable digest support for a mailing list, enter a simple filename in the *Digest filename* field. The filename you enter may not have a path component - it is always stored in the Mercury scratch directory. The filename may not contain substitution characters the way an archive filename does. If no filename is entered in a list definition, then digest support will not be available for that list. You cannot prevent a subscriber from changing in or out of digest mode if digest support is enabled for a list. The *Max size* and *Max waiting period* fields control the trigger conditions that determine when a digest is sent out to digest subscribers. If the *Max size* field is non-zero, then the digest will be distributed as soon as the digest file exceeds the number of bytes you enter. If the *Max waiting period* field is non-zero, then the digest will be sent after that number of hours has elapsed. Note that Mercury only checks digests every fifteen minutes, so the *Max waiting period* setting may not result in a precise delivery time.

Create an index of subject lines and senders When this control is enabled, Mercury will note down the sender and subject of every message in the digest, and will construct an "index" as the first item in the digest. The index contains a précis of the contents of the digest and is handy for people whose mail packages do not support the MIME digest format.

Anonymous mail support It is occasionally desirable to set up mailing lists that provide anonymity for people who send mail to them -- examples of this include suggestion boxes, and lists covering sensitive or dangerous subjects. Mercury lists support three levels of anonymity - none, where no attempt is made to hide the sender's identity; logged, where no indication of the sender's identity appears in mail sent out to the membership, but the sender's address is recorded in the Mercury log file; and total, where the sender's identity is neither shown in mail sent to the list nor in the Mercury log file. ***WARNING:*** In many states and countries, there may be legal issues associated with hosting an anonymous list, particularly if it involves discussion of activities that are illegal or subversive. Before agreeing to host an anonymous mailing list, we strongly recommend that you consult your legal counsel and check what legal obligations you may have regarding disclosure and record-keeping.

List membership settings

See also: Mailing lists, overview

This window displays a list of all users currently subscibed to the mailing list. The Status column contains three letters that indicate a variety of information about the subscriber.

The first character in the status string is one of the following status indicators:

A	Active record. A normal subscriber, receiving mail.
N	NoMail record . The subscriber is not currently receiving mail from the list
V	Vacation record. The subscriber is not receiving mail, but will be automatically re-enabled at a particular date
S	Suspended record. The subscription has been set to NoMail by the Mercury VERP processor because it has generated errors in excess of the level allowed by the list's settings. A suspended record is exactly like a NoMail record and can be re-enabled using the same methods.
X	Excluded record. An "Excluded" record denotes an address that is not permitted to subscribe to the list.
D	Deleted record. The subscriber has been removed from the list, but the Mercury daily cleanup has not purged the deleted record yet.

The second character in the status string is D if the user receives the list in *Digest* mode, or N if the user receives the list normally.

The third character in the status string is R if the user receives copies of his or her own postings to the list (*Repro* mode, the default) or N if the user does not receive such copies.

To edit a subscriber's information, double-click his record, or choose *Change selection*.

To find a subscriber quickly, click the *Find* button: this will open an incremental search window, in which you can search for the user by any part of his name, address or both.

Mercury remembers the date that the subscriber joined the list, the number of submissions made by the subscriber to the list since that time, and the date of the last submission made by the subscriber. You can view and reset these statistics in the editor dialog for the user's membership record.

Configuring server-wide mail handling policies

See also: Core module configuration

Mercury allows you to create "policies" or special tasks that are automatically applied to mail as it passes through the server. Using policies, you could do any of the following things and more:

* Scan incoming and outgoing mail messages for viruses
* Check messages for offensive or unsuitable content
* Keep copies of all mail messages sent to or from any address
* Create special logs indicating mail activity
* Start automated maintenance tasks simply by sending a mail message

In its simplest terms, a policy tells Mercury how to run an external program: the external program is responsible for performing whatever task is required and communicating the results back to Mercury, which then decides how to handle the message based on the program's response. The external program can be an executable program, batch file, a script in a suitable scripting language - just about anything you could run in the "Run..." dialog on the Windows "Start" menu. Policies are similar to filtering rules, but focus on providing the greatest possible control over the way the external program is run. The decision whether particular types of mail processing are better-handled by rules or policies will depend on your preferences and system requirements.

You can define as many policies as you wish, and Mercury will apply them all to each message in the central mail queue before it processes it. If any policy "triggers" (that is, indicates by its return value that the message fits the criteria for which it is designed to test), Mercury will perform the action associated with the policy, and will then delete the message.

Understanding how policies work

A policy consists of two parts - a *command*, which is an external program Mercury executes to determine if the policy has been triggered, and an *action*, which describes the steps Mercury should take when a policy's command indicates that a policy exception has occurred.

The policy mechanism is intended to offer the maximum of ease and flexibility in running external processes - you should be able to run a program, a batch file, a script in a scripting language - anything that could reasonably be run on a Windows system. To support this flexibility, two different ways are provided for running the external process:

Run a program and examine the return code Mercury runs the external program and uses Windows' process control facilities to work out when the program terminates. When the program has finished, Mercury retrieves the program's return value from Windows and if it is greater than zero, it assumes that the message is a policy exception. If you are writing your own programs to implement policies for Mercury, this is usually the simplest and most efficient way of doing it - simply return 0 if the message is OK, or 1 if the message contains a policy exception. You can store any information about why the exception has occurred in the *result file*, for which you can specify an explicit name, or for which Mercury can create a name for you.

Run a program using a sentinel file In this technique, Mercury creates an empty file called a *sentinel file,* then runs the external task. When the sentinel file is deleted, Mercury knows that the external task has completed, so it checks to see if another file, called the *result file*, exists on the system: if the result file has been created, Mercury assumes that the message has caused a policy exception. You can specify the names Mercury should use for the sentinel and result files if you wish: if you do not, Mercury will create them for you. You can use commandline substitutions (see below) to pass the names of the sentinel and result files to your external task. The sentinel file approach is usually the best way of implementing your policy if you want to use batch files, scripting languages, or programs that do not indicate success or failure via their process return codes. For more detail on exactly how Mercury runs a program using sentinel files, click here.

Sentinel and result files

As part of defining your policy's commandline, you can specify the names of either or both of two files: the *sentinel file* is only used if you are using the *Run a program using a sentinel file method*, and the *result file* applies to both types of program execution. If your task looks for a sentinel file with a specific name, or creates its result file with a specific name, enter that name in the relevant field. If you do not enter a name, Mercury will create a temporary filename for you automatically, which can be passed to your task on the commandline using substitutions (see below). The result file is particularly important in all cases, since Mercury expects your task to write some kind of explanatory message it can use as a diagnostic into this file.

Policy command settings

The external command in your policy item can have a number of settings associated with it.

This task requires attachment unpacking support   If you check this control, Mercury will check to see if the message contains multiple parts ("attachments"). If it does, Mercury will extract and decode each part and pass it to your task. This is extremely useful, since it allows programs that do not understand Internet transport encodings such as BASE64 or Quoted-printable to work with the simple binary data that they do understand. If you do not check this control, Mercury will invoke your task once for every message, passing it a file that contains the entire text of the message.

This task only acts on the message headers   If your task only examines the headers of the mail message and is not interested in the body or any attachments, check this control. Mercury uses this control to determine whether it can optimize the policy application process by only extracting the headers of the message. Note that even if you check this control, your task may still be passed the entire text of the message, because Mercury only makes a single file and if any other policy task does not have this flag set, Mercury is forced to put the entire text of the message into that file.

This task should only be applied to mail originating locally   If you check this control, then your task will only be invoked if the message was submitted to Mercury locally - that is, the message was placed in the Mercury queue by a copy of Pegasus Mail or another compatible client. Mail received via MercuryD or MercuryS never qualifies as "local" in this context. This setting is primarily useful if you only want to apply the policy to mail sent to the outside world by your local users.

This task should be applied before any filtering rules   Checking this control will cause the policy task to be executed before Mercury applies any global filtering rules you have defined, whereas if you leave it unchecked, Mercury will execute the policy task after the global filtering rules have been applied to the message. It is up to you to decide what order is most suitable for your site.

This task modifies the raw data of the jobs it examines   In some cases, you may wish to allow the external task to modify the actual data in the mail message it examines (for instance, you may want it to add a header to the message). Checking this control tells Mercury to copy the temporary file it creates containing the message's data back into the queue job when policy processing is complete. It is your responsibility to ensure that the policy does not corrupt or damage the message data in this case, and to ensure that the message data remains in legal RFC2822 format. Checking this control will slightly increase the time it takes to process policies on your system.

Actions Mercury can take when a policy exception occurs

When a policy exception occurs (that is, your policy task indicates to Mercury that it has triggered) there are four actions Mercury can take:

1: Delete the message with no further action

2: Forward the message to another address   enter the address to which the message should be forwarded in the "parameter" field. Mercury will attach the suspect message to a new message addressed to the address you supply, and will include any result text provided by your policy task in the body of the message.

3: Return the message as undeliverable   Mercury will return the message to the person who sent it. You can specify a template file in the parameter field - Mercury will use this template file to construct the body of the delivery failure message: you can use the ~R substitution in the template file to include the result text provided by your policy task in the body of the message.

Save to a file and notify a user   In the parameter field, enter a directory where the file containing the message should be placed, then a comma, then the address of the person who should receive notification of the exception. So, for example, if you wanted to save messages that caused exceptions in `C:\BADMAIL` and to send a notification message to `foo@bar.com`, you would type `C:\BADMAIL,foo@bar.com` in the parameter field. Once the copy has been saved and the notification sent, the message is deleted from the queue.

Commandline substitutions

When you create a Mercury policy, one of the things you must provide is a commandline: this is the command that Mercury asks the Windows operating system to execute to test your policy conditions: it consists of the name of a program, and any optional parameters that program needs to run. You can imbed certain special characters in the commandline you enter in the Mercury policy editor - when Mercury runs your command, it will replace the special characters with the proper values they represent. This process is called *substitution*.

You can use the following special characters in your policy commandlines:

~X	Is replaced by the name of the file containing the data to test (so, if your policy requires attachment unpacking, this will be the name of the file containing the specific attachment it is to examine).
~A	Is replaced by the name of a file containing the entire text of the message; if your policy task modifies the data of the jobs it examines (see above) then this is the only file it may modify.
~R	Is replaced by the name of the result file (see above)
~S	Is replaced by the name of the sentinel file (see above)
~F	Is replaced by the "original" filename for the attachment as stored in the message
~Z	Is replaced by the extension-part only of the "original" filename for the attachment, as stored in the message.
~Y	Is replaced by the current year expressed as two digits.
~M	Is replaced by the current month expressed as two digits
~D	Is replaced by the current day of the month, expressed as two digits
~W	Is replaced by the current week of the year, expressed as two digits

The date substitutions are provided largely to allow you to do simple archiving of mail: they can be used to construct file or directory names as required.

Policy issues

Performance: each policy you define will slow down the processing of mail in the Mercury queue somewhat. Depending on the complexity of the task and the size of the message, this performance reduction can range from negligible to quite significant. If your system is extremely busy (for example, more than 750 messages per hour) you should monitor carefully the impact that policy application has on your server's mail throughput.

Timeouts and crashes: if a policy task crashes or hangs, Mercury will timeout after 90 seconds. In this case, Mercury treats the message as having passed the task's tests, and will allow it to be processed normally. During the time that Mercury is waiting for the timeout, however, no mail will be processed in the central queue - so it is obviously important for you to ensure that your policy tasks are reliable and complete as quickly as possible.

How Mercury runs an external policy task using a sentinel file

When Mercury activates a policy command that specifies the sentinel file method, it follows these steps:

1: It creates an empty *sentinel file* - it uses this file to test whether or not the external program has finished its processing. The last thing the external process should do before it terminates is delete the sentinel file. When Mercury detects that the sentinel file is no longer present, it continues processing the policy.

2: It works out the name of a non-existent *result file*: when the external process has indicated that it has finished (by deleting the sentinel file, see (1) above), Mercury checks to see if a file exists using the name it assigned for the result file. If the result file exists, Mercury considers the policy to have triggered, and proceeds to activate the policy's action and to delete the message. The result file should contain a short textual statement describing why the message has triggered the policy (for example, *"Your message contains the Hiroshi Virus"*): Mercury uses this textual statement when activating the policy's action.

3: It constructs a commandline for the policy. You can control the entire format of the commandline, including whether or not the names of the sentinel and result files are included and where, using a set of special substitution characters (see below). Once it has constructed the commandline, it asks the Windows operating system to execute the commandline, and waits for the sentinel file to be deleted.

4: At this point, the world is your oyster. Once your policy command is running, it can perform whatever tests are necessary on the message data. Mercury can pass your command the entire mail message, or alternatively it can perform MIME parsing to locate each component of the mail message, and can then extract and decode each component before passing it to your command. Using this latter approach, for example, you can safely apply a simple commandline virus scanner to each attachment in the message, even if the virus scanner does not understand things like MIME formatting.

The key things to remember about your policy's task are that it must always delete the sentinel file when it has finished, and it should only create the results file if the message fits the criteria that should result in the policy's action being taken. It must also never modify the contents of the message data that is passed to it. Provided you follow these rules, your policy command can do almost anything.

MercuryI, the IMAP4rev1 server

See first:	Concurrent access with Pegasus Mail
See also:	Logging and session logging
	Troubleshooting
	Using SSL to offer secure connections

The MercuryI server module implements the IMAP4rev1 protocol, described in RFC3501. IMAP4 allows a remote client to access all the folders in a user's mailbox, unlike the POP3 protocol, which simply allows users to access their new mail.

At the protocol level, MercuryI is easily the most complex of the Mercury protocol modules, which makes it ironic that it is one of the easiest to configure and use. On many systems, no specific configuration of MercuryI is required at all.

System requirements

Most Internet protocols are reasonably "light-weight", but the nature of the IMAP protocol makes it much more demanding, especially of memory. We recommend that you calculate the memory requirements for the machine where MercuryI will be running as 300KB per user connected at the same time. So, if you have 10 users connected to MercuryI, it will be using approximately 3MB of virtual memory. This calculation is necessarily very rough - if your users have few folders then it will be substantially less, and if your users have many folders (more than 1000) then it may be substantially more. MercuryI allows simultaneous connections to the same mailbox: when simultaneous connections exist to the same mailbox, only the first will typically incur any memory overhead - the other connections are essentially "free". During normal operation, MercuryI may consume significant amounts of disk space in the Windows temporary directory, so make sure that plenty (at least 100MB) is always available.

Client configuration

At present, MercuryI presents the Pegasus Mail message store, which does not support the idea of folders that can contain both messages and other folders: folders can contain either messages, or other folders, but not both. You may need to configure your IMAP4 client to take account of this fact - for instance, in Pegasus Mail, when you create an IMAP profile for a MercuryI server, you would make sure that the "This server supports folders within folders" control is *not* checked. Future versions of MercuryI and Pegasus Mail will almost certainly allow folders to contain both messages and other folders.

Configuration

TCP/IP timeout The length of time in seconds that MercuryI should wait for data on a connection before assuming that the connection is no longer valid and aborting it.

Idle timeout The length of time Mercury should allow connected clients to remain connected between commands before deciding that they have hung and terminating the connection. MercuryI uses the idle timeout when it is not explicitly expecting a response or a command from the client, whereas the TCP/IP timeout (see above) is used when MercuryI *is* explicitly waiting for a response from the client. The TCP/IP timeout will almost always be signicantly shorter than the idle timeout. You cannot set the idle timeout to a value shorter than 30 minutes, because of the explicit requirements of the IMAP4 protocol definition (see RFC3501).

Listen on TCP/IP port The TCP/IP port is the socket into which the IMAP4 client "plugs" to access MercuryI's services. The standard port for IMAP4 services is 143, but in rare cases (particularly if you use a proxy server) you may need to change this. Consult your ISP or network administrator to find out if you need to use an alternative IMAP4 port. If you are unsure, leave it set to 143.

IP Interface to use   If your computer supports multiple IP interfaces, you can use this field to tell MercuryI which interface it should select when listening for connections: enter the interface as a dotted IP address in the general form *www.xxx.yyy.zzz*. As an example, your system may have one IP address assigned to a dialup PPP connection, and another, different IP address assigned to a local Ethernet network - you would enter here the interface MercuryI should use. If you leave this field blank, MercuryI will listen on all available interfaces. Unless you are *very* sure of what you are doing, or have been instructed by an ISP or network administrator, you should leave this field blank. If you change the IP interface in this field, you must restart Mercury before the new interface number will be used.

Connection control

The Connection Control section allows you to place restrictions on the hosts from which MercuryI will accept connections, and to configure certain capabilities based on the address of the connected host. A connection control entry can apply to a single address, or to a range of addresses. To add an entry to the list, click the *Add restriction* button; if you wish to create a restriction for a single address, enter that address in the "From" (left-hand) address field in normal dotted IP notation. To create a restriction for a range of addresses, enter the lowest address in the range you want to restrict in the "From" field, and the highest address you want to restrict in the "To" field. The addresses are inclusive, so both the addresses you enter are considered part of the range.

If you check the *Refuse connections* radio control, Mercury will not accept incoming connections from this address. Use this to suppress sites that are abusive or have been hijacked by spammers.

To edit a connection control entry, highlight it in the list, then click the *Change selection* button.

How Mercury applies connection control entries

The list of connection control entries you create can contain entries that overlap (i.e, entries that refer to addresses also covered by other entries). In the case of overlapping entries, Mercury uses the following method to select the entry it should use for any given address: if there is an entry that refers to the address on its own (not as part of a range), then Mercury will automatically use that entry; otherwise, it looks for the range that most closely encompasses the address and uses that.

Example: You have a *Refuse* entry covering the range from 198.2.5.1 to 198.2.5.128, and an *Allow* entry covering the range from 198.2.5.10 to 198.2.5.20: if a machine with the address 198.2.5.12 connects to Mercury, it will select the *Allow* entry to cover the connection, because the allow entry most tightly encompasses the connecting address (the range covers 11 addresses, where the Refuse entry's range covers 128 addresses).

SSL Support

MercuryI supports SSL for secure access to mailboxes - please click here for more information on configuring MercuryI to use SSL.

IMAP4 Troubleshooting

IMAP4 is a complex protocol doing complex things, and there is potential for quite a range of problems to occur during its use.

Concurrent use and folder damage: You ***must not*** attempt to access a mailbox via IMAP4 when it is already in use by a copy of Pegasus Mail running locally, or vice-versa. If you access a mailbox when it is already in use by another process, the result will often be folder damage. MercuryI incorporates locking protections that will prevent this situation from arising when Pegasus Mail v4.0 and later are in use, but there is no way of protecting against concurrent use with earlier versions of Pegasus Mail. If folder damage occurs in this way, you can often repair it by making sure that no IMAP access to the mailbox is in progress, running Pegasus Mail, right-clicking the folder and choosing "Reindex folder". Note that it is perfectly valid to access the same mailbox through multiple separate IMAP connections to MercuryI: the problem described here only exists if both MercuryI and a local copy of Pegasus Mail are trying to access the mailbox at the same time.

Locking problems: If Mercury crashes while a mailbox is being accessed via IMAP, the lock file used by MercuryI will usually be left behind. This will prevent people from connecting to the mailbox that was in use until the lock file is removed manually. The file is called `MAILBOXM.LCK`, and can be found in the user's home mailbox directory. Removing this file will allow access to the mailbox again.

Client errors: The complexity of the IMAP protocol provides plenty of scope for differing interpretations between client and server developers. If you find that some operation is not working correctly when you access a mailbox via MercuryI, we will be happy to look into it. Please note down any error message returned by the client program and the name and version of the client program. Next, turn on session logging in MercuryI and capture a session where the problem occurs - this step is very important! We almost certainly cannot help you without a session log. When you have all this information, please e-mail it to:

tech-support@pmail.gen.nz

and your report will be escalated to the appropriate people in the development team.

Using the IMAP server and Pegasus Mail concurrently

See also: Mercury**I**, the IMAP4rev1 server
Troubleshooting

You ***must not*** attempt to access a mailbox via IMAP4 when it is already in use by a copy of Pegasus Mail running locally, or vice-versa. If you access a mailbox when it is already in use by another process, the result will often be folder damage. Mercury**I** incorporates locking protections that will prevent this situation from arising when Pegasus Mail v4.0 and later are in use, but there is no way of protecting against concurrent use with earlier versions of Pegasus Mail. If folder damage occurs in this way, you can often repair it by making sure that no IMAP access to the mailbox is in progress, running Pegasus Mail, right-clicking the folder and choosing "Reindex folder". Note that it is perfectly valid to access the same mailbox through multiple separate IMAP connections to Mercury**I**: the problem described here only exists if both Mercury**I** and a local copy of Pegasus Mail are trying to access the mailbox at the same time.

Detecting and handling unacceptable content

See also: Defining an unacceptable content control set
Creating unacceptable content control rules

Let's face it, the worst thing about e-mail is having to wade through dozens of unwanted messages offering unwanted products and services every day - this kind of unsolicited commercial e-mail (more commonly-known as "UCE", or "spam") has become the bane of our existence in the last couple of years. Unfortunately, while spam is obviously a social problem, not a technical one, legislators around the world have been extraordinarily lax, even incompetent in their attitude towards it. As a result, we're left with no real protection from spam, even though it wastes a huge part of the Internet's dwindling resources each day.

Mercury provides a fully-integrated approach to handling this type of unwanted mail, but takes the notion even further, by allowing you to define for yourself what constitutes "unacceptable content" in both the e-mail sent *to* your site, and the e-mail sent *from* your site.

Using the *Content control* option on the *Configuration* menu, you can create sets of tests that Mercury applies to every message it processes: each set consists of three separate and optional tests -

A blacklist check You can create a blacklist of addresses and sites from which all mail is regarded as unacceptable.

A whitelist check This is like the blacklist, except that all addresses and sites that appear in the list are never treated as unacceptable.

A rule set check For messages that are not caught by the blacklist or whitelist, you can create arbitrarily complex sets of rules to test the content of the message. These rules are like Mercury's general-purpose filtering rules, but are more specific to the particular task of content evaluation, allowing unlimited numbers of "and" operations to link conditions together. Also, unlike general purpose rules, content testing rules are given a "weight": when all the rules have been processed, the weights of all the rules that were triggered are added together, and the final result is compared against a predetermined value you assign. If the combined weight of the message is greater than or equal to your preset value, the message is regarded as unacceptable. Content rule sets are stored as text files that can be easily modified using any text editor. They have a simple syntax that most system administrators should be able to learn in a very short time.

Each content control set has an *action*, which is applied when a message is deemed to have unacceptable content - this action can be to delete the message, to quarantine it for later examination, to forward it to an alternative address, to add headers to it, to return it to the sender or simply to do nothing.

You can have as many content control sets as you wish - Mercury will apply them in the order they appear in the list in the *Content control* configuration dialog: the first set that results in the message being quarantined, deleted, or otherwise removed from circulation will terminate content control processing for that message

Using the *Content control* dialog

To create a new content control definition, click the "Add" button - click here for detailed information on the various settings associated with a single control set.

To change the values for a single content control definition, select the definition in the list and click the "Edit" button.

To remove a content control definition, select it in the list and click the "Delete" button: Mercury will ask you if you want to delete the list and rule files associated with the definition as well as the definition itself -

if you use the lists or rules in other definitions as well, you should not delete them.

To adjust the position of a content control definition in the list, select it and click the "Move up" or "Move down" button. The position of a definition in the list is important, because Mercury applies them in the order they appear, and stops applying definitions to a message when a definition results in the message being deleted or otherwise removed from circulation.

Storage   Content control definitions are stored in files with the extension .PNC in the directory where Mercury is installed: each .PNC file contains a single definition. The name of the .PNC file is unimportant - Mercury/32 assigns a name at random when you create a definition, but you can rename them if you wish, provided Mercury/32 is not running when you do so.

Info files   Each content control definition can have an Information file associated with it: this is a simple text file containing information about the definition, and is displayed by Mercury in the *Information on the selected set* area of the content control definition editor dialog. The info file should have the same name as the definition file, but with "`.info`" added to it: so, if the file containing the definition is called `BASIC.PNC`, its information file would be called `BASIC.PNC.INFO`. Info files are useful if you plan to distribute your definitions; they cannot be created or edited within Mercury - they must be created manually.

Distributing definitions   If you create a definition that you would like to share with other Mercury users, all you need to do is provide the `.PNC` file containing the definition, the ruleset file, a `.info` file (if you wish) and any preconfigured blacklist or whitelist files the definition needs (these are optional). The recipient simply places these files in their Mercury install directory and restarts the program. Note that when designing a .PNC file for distribution, you should make sure that the whitelist, blacklist and ruleset filenames in the file do not have paths - this tells Mercury to look for them in its install directory.

Defining a content control set

See also: Detecting and handling unacceptable content

A Mercury Content Control set consists of four separate parts, each of which is edited in its own page within the Content Control editor dialog.

The "General" Page

Use this page to change the name that appears next to the definition in the list of definitions, and to define the types of mail to which the set should be applied.

The "Exceptions" Page

Use this page to create *Blacklists*, which identify senders whose mail should always be regarded as unacceptable, and *Whitelists*, which identify senders from whom you always want to accept mail.

The "Message Tests" Page

Use this page to maintain a set of rules that should be applied to mail messages that are not covered by either a whitelist or blacklist. The rules allow you to perform comprehensive tests on the actual content of the message, and can be linked together to create chains of tests. Each rule can have a *weight*, and after all the rules have been applied, Mercury adds up the combined weights of all the rules that matched the message: if the combined weight is greater than a value you specify, the message is marked as acceptable.

The "Actions" Page

On this page, you define what you want to happen to messages when they pass through the content control system. You can add headers to the message (which can later be detected by your filtering rules), and you can also choose other actions such as moving the message to a folder, forwarding it to another address, or deleting it.

The Content Control editor's "General" page

See also: Detecting messages based on their content
Defining a Content Control set
Next: The Exceptions page

The settings on this page allow you to change the name that appears next to the definition in the list of definitions, and to define the types of mail to which the set should be applied.

Name for this content control definition Whatever you enter in this field is the name Mercury will use to identify this definition in the list shown in the Content control dialog. You can use any name you wish up to 50 characters in length.

Apply this definition to mail originating from Allows you to choose what type of message this set should apply to. *Any source* will apply the definition to any mail passing through the queue; *Local addresses only* will apply the definition if the sender of the message has a local address (one with no domain part); and *Non-local addresses only* applies the definition to any mail where the sender's address is not local (i.e, the address does contain a domain part).

The Content Control Editor's "Exceptions" page

See also:	Detecting messages based on their content
	Defining a Content Control set
Next:	The "Message tests" page

Use this page to create *Blacklists*, which identify senders whose mail should always be regarded as unacceptable, and *Whitelists*, which identify senders from whom you always want to accept mail. For Pegasus Mail users, Blacklists and Whitelists are normally just regular Pegasus Mail distribution lists, which means that you can easily manipulate them using filtering rules, and with the right-click options "Add sender to mailing list" and "Remove sender from mailing list" while you are reading a message or browsing the contents of a folder. It is also possible to share system-wide Blacklists and Whitelists by putting them in a shared directory then entering the path to that directory in this window.

Blacklist file Enter in this field the name of a file in which Mercury should check for addresses and domains from which mail should always be regarded as unacceptable. The file need not exist already. Once you have entered the filename, you can edit the file by clicking the *edit* button next to the field. Within the blacklist file, you can use asterisks as wildcard characters to match entire domains or parts of domains: so, if you want to block all users from the domain *spam.com*, you would enter **@spam.com*. Similarly, to block all mail from any user on any machine within the *spam.com* domain group, you would enter **@*.spam.com*.

Whitelist file Enter in this field the name of a file in which Mercury should check for addresses and domains from which mail should never be regarded as unacceptable. The file need not exist already. Once you have entered the filename, you can edit the file by clicking the *edit* button next to the field. You can use the same type of wildcard operations described for the blacklist file within the whitelist.

The Content Control editor's "Message Tests" page

See also: Detecting messages based on their content
Defining a Content Control set
Content control rules - syntax and usage
Next: The "Actions" page

Use this page to maintain a set of rules that should be applied to mail messages that are not covered by either a whitelist or blacklist. The rules allow you to perform comprehensive tests on the actual content of the message, and can be linked together to create chains of tests. Each rule can have a *weight*, and after all the rules have been applied, Mercury adds up the combined weights of all the rules that matched the message: if the combined weight is greater than a value you specify, the message is marked as acceptable.

Content processing rules file Enter in this field the name of a file containing rules written using the Pegasus Mail/Mercury content control filtering language - Mercury will apply these rules to any message that is not trapped by either the black or white lists. You can either edit the rule file by clicking the *Edit* button next to the field, or by using an external text editor. The internal editor has a filesize limitation of 32KB - if your ruleset file is larger than this, you will need to use an external editor to edit it.

Checking syntax When you are editing your rule file, you can check that the rules you have entered are syntactically correct by clicking the *Check Syntax* button in the rule editor: if Mercury encounters any errors in the rule set, it will pop up a dialog describing the error and place the cursor on the line where the error occurred.

Weight at or above which this definition activates Within the rule file, each rule can be assigned a *weight*, which is a numeric value; after all rules have been processed, Mercury adds together the weights assigned to every rule that matched the message - if the total weight is greater than or equal to the value you enter here, the message will be deemed "unacceptable" and will be subject to the action you define on the "Actions" page of the editor.

Check at most this many bytes in each message If you enter a value greater than zero here, Mercury will only scan that many characters in each message when applying your content control rule set. This can reduce the time taken to perform content control, but can also result in less undesirable mail being detected. If you find that Content Control is taking a significant amount of time on your system (for instance, if you receive many large messages), entering a value of around 8000 in here will typically provide a good balance between speed and detection.

Note that when scanning multipart messages, Mercury adds together the size of each text section it scans to calculate the number of bytes it has scanned - it does not simply blindly read through the file.

The Content Control editor's "Actions" page

See also: Detecting messages based on their content
Defining a Content Control set

On this page, you define what you want to happen to messages when they pass through the content control system. You can add headers to the message (which can later be detected by your filtering rules), and you can also choose other actions such as moving the message to a folder, forwarding it to another address, or deleting it.

If a message has a weight greater than the activation weight When a message has a weight higher than the activation weight, either because it appears in the blacklist, or because its calculated weight after rule processing exceeds your predetermined value, the action you select here will be taken on the message. Some actions have a parameter (for instance, when you select "forwarding", the parameter is the address to which the message should be forwarded). The following actions are possible:

Take no further action This action is useful if you want to turn off processing for a while, but still want the mail to be marked as "processed". This option is also useful if you only want standard headers added to messages as they pass through Content Control (see below for more information on adding standard headers) - you will typically select this case when you want to use filtering rules to handle such messages at a later stage.

Add an identifying header If you select this action, Mercury will add an identifying header to the headers of the message but will not otherwise divert or alter the message. Whatever string you enter in the parameter field, Mercury will add as a header in the delivered message, completely unmodified (so, you must include the header keyword, the colon character, and the header body exactly as you want them to appear in the message). You can use this action as well as using the standard "graphical" and diagnostic headers (see below).

Copy the message to another address Selecting this action will make a copy of the message and send the copy to the address you specify in the "parameter" field. The original message will not otherwise be diverted or altered in transit and will be delivered normally.

Forward the message then delete it This action will divert the message to the address you specify in the "parameter" field. When you select *Forward and delete*, this action will cause all content control processing to terminate for the message, because it will be effectively removed from circulation.

Move the message to a directory as a file This action diverts the message to a "quarantine directory". When you click the "Set" button, Mercury will prompt you to select a directory, which can be anywhere on the local machine or on your network. Mercury will move the message into this directory as a file and remove it from the queue so that no further processing or delivery occurs.

Delete the message Just like it says - this action deletes the message, end of story. All content control processing ceases at this point, and the message is gone forever. We suggest you use this action with considerable care.

Header addition and advanced options

As well as taking the action you specify on the message, Mercury can add certain headers to mail to indicate the results of Content Control processing.

Add graphical X-UC-Weight headers for unacceptable mail When this control is checked, Mercury will add a header called `X-UC-Weight` ("UC" stands for *unacceptable content*) to any message that has a weight greater than the activation weight for the set (see the Message tests page for more information on how the activation weight is calculated). The X-UC-Weight is graphical, in that it contains a little graph that indicates how unacceptable the message actually was. The graph consists of one to four hash characters, with the following meaning:

`[####]`	The message has been Blacklisted, or has a weight greater than 9990.
`[### ]`	The message's weight is more than three times greater than the activation weight for the set (in other words, it's a stinker).
`[##  ]`	The message's weight is more than twice but less than three times the activation weight for the set.
`[#   ]`	The message's weight is more than the activation weight for the set, but less than twice the activation weight.

After the graphic, the actual weight of the message is shown in brackets.

Add graphical X-AC-Weight headers for acceptable mail It is possible to assign negative weights to a message during content control processing - in fact, this is what the whitelist does (it assigns a weight of -9999). If a message comes through the content control process with a negative weight, it is regarded as *Acceptable* - something important or desirable. If you have rules in place that have negative weights to promote the value of a message, you can instruct Mercury to add a header called `X-AC-Weight` ("AC" stands for *acceptable content*) to any messages that end up with a negative weight. This can be a very handy way of highlighting messages with important content - you can use filtering rules later in the process to detect the X-AC-Weight header and take appropriate actions. Like the X-UC-Weight header (see above), The X-AC-Weight header is graphical, in that it contains a little graph that indicates how acceptable the message actually was. The graph consists of one to four hash characters, with the same meanings as in the X-UC-Weight graph above, except that the values are negative: so, `[### ]` would mean that the weight of the message is less than (3 * the activation weight * -1). Similarly, `[####]` means that the message has been explicitly whitelisted, or has a value lower than -9990.

Add a diagnostic header showing which rules were matched When this control is checked and a rule generates any non-zero value after Content Control processing, Mercury will insert a header called `X-CC-Diagnostic` into the message: this header contains a summary of the rules that triggered during processing, and is a useful way of working out why a message was given the weight it received. Each rule is written into the header in an abbreviated form, unless it has a Tag defined, in which case the tag is written into the header instead. For each rule written, the weight associated with that rule is shown in brackets as well.

Content control rules - syntax and usage

See also: Detecting and handling unacceptable content
Specialized content control tests

Mercury's content control rule language has been designed to be simple and flexible: it is based on the use of regular expressions, which describe patterns of text within the message.

A rule set consists of a sequence of tests applied sequentially to the message.

The types of test

Body and subject tests: these tests look for content in either the subject field or the body of the mail message. There are two types of test - a *substring test*, using the `CONTAINS` operator, and a *regular expression test*, using the `MATCHES` operator. The substring test simply looks for a group of characters anywhere in the specified location, while a regular expression test looks for more complex patterns of characters. See below for more information on the difference between substring and regular expression tests.

```
IF SUBJECT CONTAINS "string" WEIGHT x
IF SUBJECT MATCHES "regular_expression" WEIGHT x
IF BODY CONTAINS "string" WEIGHT x
IF BODY MATCHES "regular_expression" WEIGHT x
```

If you want to test for a string or a pattern in either the body or the subject field, you can use the CONTENT test instead - this checks in both places automatically:

```
IF CONTENT CONTAINS "string" WEIGHT x
IF CONTENT MATCHES "regular_expression" WEIGHT x
```

Header tests: these tests check specific headers or groups of headers in the mail message. The `SENDER` test looks in the "From", "Sender", "Resent-From" and "Reply-to" fields of the message, while the `RECIPIENT` test looks in the "To", "CC", "BCC" and "Resent-To", fields. The `HEADER` test allows you to check any single header in the message: if the header does not exist, the test does not trigger. Finally, the `EXISTS` test allows you to check whether or not a specific header exists in the message.

```
IF SENDER CONTAINS "string" WEIGHT x
IF SENDER MATCHES "regular_expression" WEIGHT x
IF RECIPIENT CONTAINS "string" WEIGHT x
IF RECIPIENT MATCHES "regular_expression" WEIGHT x
IF HEADER "headername" CONTAINS "string" WEIGHT x
IF HEADER "headername" MATCHES "regular_expression" WEIGHT x
IF EXISTS "headername" WEIGHT x
```

Wordlist tests - HAS and HASALL There are also some more specialized tests you can use to test for groups of words in a message - `HAS` and `HASALL`:

```
IF xx HAS "wordlist" WEIGHT x
IF xx HASALL "wordlist" WEIGHT x
```

(Note that "xx" can be "subject", "sender", "recipient", "header" or "body") Both of these tests accept a list of words separated by commas as their parameter. The `HAS` test will succeed if the message contains any of the words in the list, while the `HASALL` test will succeed if the message contains all the words in the list, in any order.

Example: to detect a message containing "viagra", "prescription" and "erectile"

```
        IF BODY HASALL "Viagra, prescription, erectile" weight 50
```

Specialized tests Mercury has a number of specialized tests that are specifically designed for detecting spam (unsolicited commercial e-mail); these tests examine special characteristics of the message that could not otherwise be easily detected using standard regular expressions. Specialized testing can trigger on things like Lazy HTML content (messages containing links to graphics instead of the graphics themselves), excessive use of HTML comments and other telltale signs of spam. For more information on the specialized tests, click here.

Negating and linking tests (NOT, AND and OR operators): You can negate a test by using `IFNOT` instead of `IF`: similarly, you can link multiple tests together by using `AND`, `ANDNOT`, `OR` or `ORNOT` instead of `IF` in each test following the first.

Substring matching vs Regular expressions: Any test that uses the `CONTAINS` keyword to perform a substring search does a simple string search instead of a regular expression match: this is a little faster and a little easier to understand than the regular-expression based versions of the rules. Note that `CONTAINS` tests are completely literal - no regular expression matching of any kind occurs. `CONTAINS` tests are always case-insensitive - so, the strings "foo" and "FOO" are identical as far as a `CONTAINS` test is concerned.

Detecting obfuscated text: A common trick in spam is to embed unusual characters in words that commonly trigger anti-spam routines - like "vi@gra", or "pen1s"; indeed, this technique is now becoming so pervasive that Mercury includes a special keyword just to handle it. When defining `HAS`, `HASALL` or `CONTAINS` rules, you can add the keyword `OBFUSCATED` (you can abbreviate this to `OB` if you wish) before the `WEIGHT` keyword in the rule - like this:

```
        IF SUBJECT CONTAINS "viagra" OBFUSCATED WEIGHT 51
```

This rule will detect any of the following words in the subject line of a message: "viagra", "v-i-a-g-r-a", "vi@gra", "V 1 -@- G R A" or even "_v$1&@(G*r*A".

If you want to test for a phrase when using the `OBFUSCATED` keyword, you must enter the phrase in the rule without spaces: so, if you wanted to check for any obfuscated version of the phrase "increase the length of", you would have to enter it like this:

```
        IF CONTENT CONTAINS "increasethelengthof" OB WEIGHT 51
```

Note that you cannot use the `OBFUSCATED` keyword on a `MATCHES` test - if you do, Mercury will simply ignore the keyword and match using the expression you provide.

CAUTION You should exercise a certain amount of caution when using obfuscated tests, because there is a slightly increased risk of false positive matching (i.e, having two adjacent words which while harmless on their own, add together to form a trigger word).

Tags Any rule can have a *Tag*, or a name used to describe it: the tag is used by Mercury when you have told it to construct a diagnostic header for messages, and is useful when the test that the rule is performing is either very verbose or very obscure, or when the actual text of the rule may contain offensive material.

Example:

```
        IF BODY HAS "Fuck, Shit" Weight 100 Tag "Offensive language"
```

In this example, when Mercury prepares the "X-CC-Diagnostic" header in the message, it will format it as `Offensive language (100)` instead of `Body Has "Fuck, Shit" (100)`, which may be offensive to some people.

Tags are optional, and can appear instead of or after a `WEIGHT` statement. The name parameter to a `Tag` statement must always appear in double-quote marks, as shown in the example above.

General layout

The rule language itself is not case-sensitive, so the following tests are both semantically valid:

```
If Sender contains "foobar" weight 80
IF SENDER CONTAINS "foobar" WEIGHT 80
```

Furthermore, whitespace is ignored, so you can layout your tests in whatever way you feel is clearest: as an example, the following is a completely syntactically valid test:

```
If
      sender
            contains
            "Foobar"
      Weight 80
```

The only restriction is that neither a string nor a keyword can cross a line boundary; so, the following test is invalid:

```
If sender con
      tains foobar Weight 80
```

Examples:

1: To detect a message where the sender's address contains "spam.com"

```
IF SENDER CONTAINS "spam.com" WEIGHT 50
```

2: To detect a message where the sender's address contains "spam.com" and the body of the mesage contains the word "viagra"

```
IF SENDER CONTAINS "spam.com"
AND BODY CONTAINS "viagra" WEIGHT 50 tag "Viagra ad"
```

3: To detect a message where the sender's address contains "spam.com" and either the subject field or the message body contains the word "viagra", allowing for possible obfuscation of the text:

```
IF SENDER CONTAINS "spam.com"
AND CONTENT CONTAINS "viagra" OBFUSCATED WEIGHT 50
```

4: To detect a message where the sender's address contains "spam.com", the message has no "Date" header, and the Subject or the Body contains "viagra"

```
IF SENDER CONTAINS "spam.com"
ANDNOT EXISTS "Date"
AND CONTENT CONTAINS "viagra" WEIGHT 50
```

Making the most of regular expressions

The `CONTAINS` test does a simple string search, looking for the exact text you provide anywhere in the message. Often, however, you may want to look for patterns of text rather than exact strings: you can do this by using a `MATCHES` test instead of a `CONTAINS` test, because `MATCHES` tests use a special pattern-matching mechanism called a *regular expression* to describe the general form of text you want to find.

Using regular expressions, you can detect extremely complex patterns of text within the messages you filter. Mercury's regular expression uses what is known as a *metasyntax* to describe the pattern you want to match: in the metasyntax, certain characters have special meanings that Mercury applies to the text it is testing. The following special characters are recognized in your expressions:

`*`	Match any number of any characters
`?`	Match any single character
`+`	Match one or more occurrence of the last character
`[ ]`	Encloses a group of characters to match. Ranges can be specified in the group using '-'.
`/w`	Match zero or more whitespace characters
`/W`	Match one or more whitespace characters
`/c`	Toggle case sensitivity (case-insensitive by default)
`/s`	Toggle whitespace stripping - ignore all whitespace in the source
`/b`	Match a start-of-word boundary (including start-of-line)
`/B`	Match an end-of-word boundary (including end-of-line)
/x	Toggle "ignore-non-alpha" mode - ignore non alphanumeric characters
/X	Toggle "ignore-spam" mode - ignore non-alphanumeric except @ and |

You can use any number of metacharacters in an expression - so, for example, to detect all users at any system within the domain "spam.com", you could use the regular expression

```
*@*.spam.com
```

The set operation is especially powerful, particularly when combined with the repeat occurrence operator: so, to detect a message where the subject line ends in a group of three or more digits (a common indicator of a spam message) you would use this expression:

```
Subject:*[0-9][0-9][0-9]+
```

In this expression, we use the "*" operator to match the general text within the subject line, then we use the set "`[0-9]`" three times to force a minimum of three digits, and a "+" operator to detect any extra digits following the third one. Because there is no "*" at the end of the expression, the digits must therefore be the last characters on the line - if there is any text following them, the expression will fail.

Normally, Mercury compares all text on a case-insensitive basis - that means that it will regard "hello" and "HELLO" as being the same. In some cases, though, the case of the text you're matching can be important, so the "/c" operator allows you to toggle Mercury between case insensitive and case-sensitive comparisons. So, to detect the string "FREE!" anywhere within the subject line of a message, you would use this expression:

```
Subject:/c*FREE!*
```

In this expression, the expression will only succeed if the word "free" appears in uppercase characters.

Important note: matching anywhere within the target text Mercury's regular expression parser is designed to start at the beginning of the text it is evaluating and to stop matching at the end. As a result, if you want to find a regular expression anywhere within the text you are examining, you need to start and end the expression with an asterisk operator (*). To illustrate why this is necessary, consider the following three regular expressions:

```
Wearing a fedora hat
Wearing a fedora hat*
*Wearing a fedora hat.
```

The first will only match if the target text consists only of the string "`Wearing a fedora hat`": if there is text before or after the string, the match will fail. The second will match only if the text *starts* with the string

"`Wearing a fedora hat`". If there is any text before the string, the match will fail, but the "*" at the end ensures that any text following the string will not prevent a match. The last example will match only if the text *ends* with "`Wearing a fedora hat`" - again, the "*" at the start of the expression will match anything prior to the string.

If you want to find the expression anywhere it occurs in the target text, you need to enter it as

```
        *wearing a fedora hat*
```

If you forget to add the leading and trailing * operators, the rule will typically not work, and this error can be quite difficult to spot when you're simply reading the source file.

Specialized Content Control Tests

See also: Detecting messages based on their content
Content control rules - syntax and usage

Mercury has a number of specialized tests that are specifically designed for detecting spam (unsolicited commercial e-mail); these tests examine special characteristics of the message that could not otherwise be easily detected using standard regular expressions.

Specialized tests are entered like any other rule in the rule set, and have the following general form:

```
IF TEST "Testname-and-parameters" WEIGHT x
```

The name of the test and any parameters it requires are entered as a single string after the keyword `TEST`: doing things this way allows tests to be added in future without breaking existing copies of Mercury/32. Tests are case-insensitive unless specifically noted below.

The following tests are available at present:

LazyHTML This test will trigger if the message is an HTML message that contains an IMG link to a remote graphic - apart from being extraordinarily rude and annoying, this type of link is a very reliable indicator of spam. Two parameters are available for this test - `Tolerant` and `Strict`; the `Tolerant` parameter tells Mercury that a message may contain one (and no more than one) Lazy HTML graphic link without triggering, while the `Strict` parameter tells Mercury that any Lazy HTML is to cause a trigger.

Example: `If Test "LazyHTML Tolerant" weight 51`

HasIFrame This test will trigger if the message contains an HTML `IFrame` tag - this is an almost 100% certain indication of a virus-generated message containing viral payload designed to take advantage of an infamous activation bug in Microsoft Outlook. There is no imaginable justification for a valid e-mail message to contain an `IFrame` tag. This test takes no parameters.

Example: `If Test "HasIFrame" weight 51`

HTMLComments This test allows you to trigger if a message has more than a certain number of HTML comments. Spam often uses HTML comments to break up keywords that would otherwise be detected as "naughty": because Mercury strips HTML tags before applying content control testing, this type of trick won't work with it, but the presence of all those comments is a dead giveaway that the message is spam. The parameter to this test is the number of comments above which Mercury should trigger the test.

Example: `If Test "HTMLComments 20" weight 51`

Garbage This test simply counts the number of characters in the message that are not standard ASCII characters: if the percentage of non-ASCII characters is higher than the value you specify, the test will trigger. This test is an almost infallible way of detecting Russian and Asian spam, but you will need to be careful if you receive legitimate mail from these regions (we recommend whitelisting senders who might need to send you messages like this). The parameter to this test is a percentage value of the whole message that must be non-ASCII before a trigger occurs.

Example: `If Test "Garbage 25" weight 51`

Other arbitrary tests may be added in future versions of Mercury/32.

The History of Mercury (and Pegasus Mail)

In 1989, I wrote an e-mail application in my spare time, because I needed it at my work. The people at my work seemed to like it, so I released it on the Internet because I thought other people might benefit from it too: by 1993 it was one of the most widely-used e-mail applications in the world - and I'm still as surprised about that today as I was back then. That program is called Pegasus Mail, and it is now the oldest PC-based mail system still in wide-spread use. By 1993, though, it was clear that the Internet was going to dominate the future of computing, so I sat down and started writing the Internet mail server package that I felt Pegasus Mail needed as a complement. By August 1993, Mercury was an established product, and was handling millions of mail messages every day for people all around the world.

My name is David Harris -- I'm the person who develops both Pegasus Mail and Mercury. There is no anonymous corporation behind the program, and the same pair of hands that wrote the first versions way back then is still writing them in 2004. People regularly ask me why I originally wrote these programs, and why I still make them available for free: Pegasus Mail and Mercury are closely - or even inextricably - intertwined in history and pedigree, so it's probably easiest if I describe how Pegasus Mail came about - because in doing that, I'll also expose the motivations that underly Mercury and explain why it's so important to me that it be available for free.

In 1989, the University where I worked (in Dunedin, New Zealand) installed its first Novell NetWare network. It wasn't until after we installed it that we found that it didn't include an e-mail system, but we'd already used up our budget and the commercial mail packages that were available were very expensive. To fill the gap, I wrote a simple e-mail program in my own time and made it available on the network: I was quite surprised to find that people liked it.

Early in 1990, after tidying it up a little, I made it available on the Internet at a friend's FTP site in Hawaii, expecting that four or five other sites might find a use for it... In the first week of availability, it was downloaded more than 100 times, which also surprised me. I found that I was receiving mail from people thanking me for giving them something they couldn't have afforded any other way -- communication. I grew to understand that communication had to be regarded as a right, not as a privilege: it seemed to me in 1989, as it still seems to me now, that *freedom of speech is useless if nobody can hear you*. Giving away Pegasus Mail seemed to be a means by which I could try to make communication more accessible to a much wider range of people who needed it - it seemed like a small act that might help in some minor way to level the playing field between the wealthy corporations and the rest of us.

From that time, I began a curious double existence, working at the University by day, and working on Pegasus Mail at night, refining and tuning it to add the things people were asking for. With each release of the program, usage grew, until by 1993, the demands it was placing on my time were so great that I had to make a choice between my safe University job and going out full-time to support Pegasus Mail and (by that stage) Mercury. Leaving the University gave me what I needed most -- time -- but took away what I needed to survive -- my salary.

This put me in an awkward situation: the ideals that had motivated me to make the program available in the first place were still just as valid as ever, but I also had to eat. I hit upon the idea of making the manuals available for sale as an option to support the development of the program. This allowed the software to remain free, and the addition of extensive online help ensured that the program remained useful even without the manuals: so, the larder was stocked without compromising the ideals. To this day, my only source of income remains the sale of manuals for Pegasus Mail and Mercury.

Since 1990, the world has changed: the Internet has become more or less a commodity, and people's expectations of software have altered enormously. I've worked hard to try to keep up with the expectations of my user base and to keep offering programs that fit all their needs. I enjoy making Pegasus Mail and Mercury available on these terms, knowing that they both help people: your support is a key component of making this all happen, whether it's by purchasing manuals, or by showing the programs to people who might benefit from using them, or simply by enjoying the fruits of my labour. With your support and backing, I look forward to being able to offer both Mercury and Pegasus Mail in the future for as long as the ideals that originally drove me to write them remain relevant.

Cheers!

-- David Harris --
Author, Pegasus Mail and Mercury
Dunedin, New Zealand, December 2004.
David.Harris@pmail.gen.nz

Mercury/32 Features

Shakespeare said "Nothing will come of nothing", and the common peoples' wisdom usually tells you that "you get what you pay for". On these grounds, since Mercury is free software, you might expect its feature set to be limited in some way... Well, we think Mercury is one of the richest, most powerful mail processing environments you can get at any price, and we believe that you will think so too after you've tried it. Below is a partial list of the features Mercury offers.

Protocols supported

* SMTP (server, relay-based client and full end-to-end delivery client)
* POP3 (server and distributing client)
* IMAP4rev1 (with multiple simultaneous access to the same mailbox)
* PH (server, for directory lookups)
* Finger (server, for directory lookups)
* PopPass (server, for remote password changing)
* HTTP (server, for web-based mailing list management)
* SSL (Secure sockets layer) on SMTP, POP3 and IMAP servers

Core module features

* *Multiple domains* Support for multiple domains on one system. A single Mercury/32 server can service a practically unlimited number of different e-mail domains.
* *Aliases and addresses* Support for unlimited aliasing and alternative address forms (so, a user can have any address, not just his or her username@your.domain).
* *Autoresponders* fully-programmable automatic replies, including options allowing different replies to be sent automatically depending on the day of the week, the month, the time of day or during an arbitrary range of dates. Mercury's autoreply logic has extensive checks and balances to prevent mail loops and mail storms, which are common in other systems.
* *Forwarding* Full support for automatic forwarding of mail, including forwarding to multiple addresses.
* *Filtering* Comprehensive mail filtering allowing an almost unlimited number of tests and actions to be performed on a message. Filters can be global, or tied to specific addresses.
* *Comprehensive content control* set up exacting tests to reduce or eliminate unwanted "spam", or to monitor the content of messages passing through the server.
* *Policies* -apply your own external tests to mail passing through the system. For example, using a policy you can add comprehensive virus scanning to your mail server at no cost by simply linking it to a personal virus scanning tool.
* *Mailing lists* Mercury's support for subscription-based mailing lists is so strong it deserves its own section (see below for more details).
* *Statistics* Mercury keeps detailed statistics of many aspects of system operation, and can mail these to any address at an interval you specify.
* *Templates* All notifications (such as delivery failure messages) can be customized using a simple text editor.
* *Multiple queues* Mercury can service queues on many servers: this is particularly useful if you are using Pegasus Mail, since it allows a single Mercury/32 system to service an almost unlimited number of Novell NetWare or Windows NT servers.
* *Mail server* Mercury has a built-in command-driven mail server that can be used to perform list management, alter user settings and more simply by sending it an e-mail message.
* *Easy user interface* Mercury has an attractive, easy-to-use interface that will be familiar to any Windows user - no need to fiddle around with obscure text files to do configuration.
* *Domain mailboxes* Mercury makes it easy to create a single mailbox that receives all mail addressed to a particular domain.
* *NetWare support* Mercury has special support for Novell NetWare local area networks, and can integrate with either NetWare NDS/eDirectory-based systems (such as NetWare 5 and 6), or older Bindery-based NetWare servers.

Scheduling module features

* Mercury includes an "overseer" module that controls connection to the Internet. If loaded, this module can control when and how often Internet connections should occur, including separately-settable peak and off-peak settings for each day of the week (so, you can have it connect less frequently during the weekend, for example). The scheduling module integrates with other protocol modules to ensure that they are active when they need to be, and is especially well-suited to environments where the Internet connection is intermittent, such as dialup, ADSL or ISDN systems.

SMTP Server features (incoming mail)

* Full support for RFC821/2821 transmission format, including ESMTP
* Support for Authenticated SMTP using LOGIN, PLAIN and CRAM-MD5 authenticators
* Exceptional range of methods for handling relay prevention
* Killfile allows mail to be refused from certain domains or addresses
* Connection restrictions allow connections to be refused from certain addresses or address ranges.
* Support for "spam blacklist" services such as RBL. You can set up any number of blacklist query definitions and can tag (mark with a header), redirect or reject messages that are blocked by a service. Both address-based and domain-based blacklist services are supported.
* Full session logging capability allows you to record every character that passes between the server and the connected client.
* Listen on multiple ports - Mercury can listen on any port you choose and can be told to listen to two different ports simultaneously if you wish (handy for working behind firewalls or with proxy servers).
* Supports ESMTP size restrictions for controlling the size of incoming mail

SMTP Client features (outgoing mail)

* Choose between either relaying SMTP (where Mercury connects to another computer to send mail) or full end-to-end delivery. Relayed SMTP is especially useful behind firewalls, or when you have an intermittent Internet connection.
* Full session logging, allows you to see every character that passes between the client and the server to which it connects.

POP3 Server features (remote new mail access)

* Connection restrictions allow connections to be selectively permitted or refused based on addresses or address ranges.
* Special profile options allow you to tailor the behaviour of the mailbox, including options to present only unread mail, options to make deletions final, preventing deletions from the mailbox and more.
* Individual POP3 profiles allow customization of the server's behaviour for each mailbox.

Distributing POP3 Client features

* Can be configured to download mail from an unlimited number of remote POP3 mailboxes and to process that mail into any number of local mailboxes (so, you can have four remote mailboxes all downloaded into one local user, or each remote mailbox being downloaded to separate users).
* Supports "domain mailboxes" - can distribute mail properly to local users even if all mail for your domain ends up in a single mailbox. The POP3 client uses extensive heuristics to work out exactly to whom each message in a domain mailbox is actually addressed.
* Extensive customization options allow you to tailor the behaviour of the client to suit your circumstances: for instance, you can specify non-standard headers in messages that contain recipient addresses, and you can specify a "default user" who should receive all mail that cannot be explicitly identified as local.

IMAP4rev1 Server features (remote mailbox access)

* Practically maintenance-free - no specific per-user configuration is necessary: just turn the server on and your users can retrieve mail using any IMAP-compliant mail package.
* Tested with many webmail packages that use IMAP as a back end, including IMP, Twig, SquirrelMail and others.
* Allows multiple separate users to connect to the same mailbox simultaneously - especially handy for helpdesks, support lines and other situations where shared write access to folders is required.
* Connection restrictions allow connections to be selectively permitted or refused based on addresses or address ranges

Subscriber-based Mailing List features

* Simple e-mail based subscription and unsubscription
* Web-based subscription management facilities
* Allows both moderated (managed) and unmoderated mailing lists. For moderated lists, you can choose to allow only moderators to post to the list, or to make posting public, but to have moderators the only ones able to add or remove subscribers. A "primary moderator" can also be designated to receive rejected list postings automatically.
* Extensive error-handling options, including full support for VERP (automated error management).
* Member lists can be made available to subscribers or not, as required. Simple mail-based commands allow enumeration of list membership where it is permitted.
* "Rolling archives" of all mail posted to lists can be created automatically.
* You can control where errors resulting from list mail delivery should be sent
* You can create "welcome" and "farewell" messages that are automatically sent to subscribers on creation and termination of their subscription respectively.
* Confirmed subscription: you can force users to send a special confirmation message before their subscription becomes activated.
* Full support for automatic generation of mail digests, using the MIME digest format. Individual subscribers may choose whether or not to receive list mail in digest form. When digests are sent can be based on either digest size or elapsed time.
* Subscription expiration - subscriptions can be configured to expire after a certain number of days.
* Posting restrictions: lists can be configured to accept mail from anyone, from subscribers and moderators only, or from moderators only.
* Size restrictions: lists can be configured not to distribute mail larger than a size you specify.
* Password-controlled submission: lists can be configured to accept mail for distribution only if a specific password header is present in the message (many mail clients, such as Pegasus Mail, allow you to add custom headers to your messages). Moderators and subscribers can have separate and different sets of passwords..
* Full support for RFC2369 List URL header fields for automating common subscription tasks for subscribers.
* Replies can be forced to go to the list instead of the original sender or not, as required.
* Automatic subject line modification: you can force certain text into the Subject line of messages distributed to lists, usually to assist with mail filtering by subscribers.
* Encryption of mail distributed to lists is possible if your subscribers use Pegasus Mail.
* Anonymous lists - you can create lists that automatically "anonymize" the mail they send - this is especially useful for "suggestion boxes" or for subjects involving sensitive content.
* Rich subscriber settings: subscribers can be "active", "on vacation" (i.e, their subscription automatically turns on again after a set period) "inactive" (i.e, still belong to the list and able to post to it, but not receiving distributions) or "barred" (i.e, prohibited from subscribing in future).
* Subscriber statistics - see at a glance when a subscriber joined the list, when he or she last posted, and how many postings he or she has made to date.

... And much, much more. We haven't even touched on user interface features like consoles, activity monitors, speed search in lists, pause options for each module... There's an enormous amount in there, and the only way to grasp how comprehensive the system really is is to install it. We hope this list has whetted your appetite, though.

Secure connections using SSL/TLS

Mercury/32 has comprehensive support for secure connections using the Internet SSL/TLS protocols. "SSL" (*Secure Sockets Layer)* and "TLS" (*Transport Layer Security*) are standards for transferring data across Internet connections in an encrypted format to ensure security. Pegasus Mail supports both SSL and TLS so from this point on we'll use the term "SSL" to mean both. The way these protocols are implemented means that a client and a server can negotiate an encrypted transaction in a way that prevents the data from being intercepted in transit even if the intruder can see the entire session.

There are three steps to enabling support for SSL in a Mercury/32 protocol module:

Step 1: Enable advertising of the protocol

Step 2: Install a certificate the server can offer to clients connecting via SSL

Step 3: [Optional] Decide whether you will allow users to login without using SSL.

Step 1 is easy: in the configuration dialog for the protocol module, switch to the page called "SSL" and check the control labelled *Enable support for SSL/TLS secure connections*. This tells Mercury that it can advertise the availability of SSL services to clients when they connect to it. If you do not enable this control then the protocol module will neither advertise the availability of SSL services, nor will it accept attempts to establish SSL connections.

Step 2 is also easy in Mercury, but it needs a little explanation. When an SSL connection is established, the server is required to supply a *certificate* to the client: a certificate is a specially formatted piece of data that is intended to prove that the server is in fact who it claims to be: the server is required to provide a certificate even if the proof of identity is not particularly important. You can obtain certificates from a *Certification Authority*, such as Thawte or Verisign, but the process is complicated and expensive: what's more, for SSL connections to mail servers, the proof of identity that a Certification Authority Certificate provides is typically not as important as encrypting the data in transit. For this reason, Mercury also allows you to create a special type of certificate called a *self-signed certificate.*

A self-signed certificate basically says to the client "This is my name, and you can trust me"; now, you don't have to think about this for very long to realize that this isn't very secure, but there's another aspect of certificates that compensates for this: every certificate, even a self-signed one, has a unique attribute represented by a calculation called *a fingerprint* which is essentially impossible to forge. As a result, you can get excellent security even when using a self-signed certificate by comparing the certificate's fingerprint with the fingerprint you obtained the last time you connected to the server: if the fingerprints are different, this indicates that the certificate has changed and that there may be a security issue. The point is that provided you are confident you connected to the right server the first time you ever connected to it via SSL (and hence got a valid fingerprint for the server), you have a basis for detecting changes in the server's certificate fingerprint, and hence can detect potential security breaches on all subsequent connections. This technique is not a good approach for things like e-commerce sites, because you'll mostly only connect to them once or twice, so the risks of certificate falsification are magnified, but it works quite well with mail protocols because you tend to connect to the same small group of servers continuously, hence the change in fingerprint is really the most significant issue. Pegasus Mail, Mercury's companion mail client, supports fingerprint comparison on certificates: other mail clients may also do so.

To create a self-signed certificate in Mercury, type a filename into the Server Certificate "filename" field in the SSL configuration page: this is the name of a file in which Mercury can secure the certificate and its associated security information - any existing file by this name will be overwritten when you create the new certificate.

Important note: If you have already created a self-signed certificate for one Mercury protocol module, you can use that certificate in any other protocol module without having to create it again. So, if you have already created a self-signed certificate for use in the MercuryI IMAP server, you can simply type in its filename for both the MercuryP POP3 server and the MercuryS SMTP server without having to create new ones.

Once you have entered the filename, simply click the *Create...* button in the SSL configuration dialog. Mercury will open a dialog prompting you for the Internet domain name to be associated with the certificate - the default value for this is the server's Internet domain name as it has been entered in the Mercury Core Module configuration dialog: it is very important that you enter the right domain name here, because some clients may refuse to accept the certificate if its associated domain name does not match the domain name they thought they were connecting to. When you have entered the name, simply click "Create" and Mercury will manufacture a suitable self-signed certificate for you and will store it in the filename you supplied. Assuming no error occurs in certificate creation, you can now click the OK button to save the configuration and Mercury can immediately begin accepting SSL connections - that's all there is to it.

Even more important note: The file in which Mercury stores your certificate is not especially secure; it is encrypted in a manner beyond the ability of almost anyone except the most determined and experienced security expert, to crack, but it is conceivable that it *could* be cracked. As a result, we do not recommend the use of Mercury's SSL services in environments where the physical system on which Mercury runs is not located in a secure location.

Step 3 involves deciding whether or not people should still be able to login to the server without first establishing an SSL connection. Since the primary reason for using SSL is to prevent usernames and passwords from being transmitted in a format that could be intercepted in transit, it makes little sense to allow people to login without securing the link first. The MercuryI IMAP server and the MercuryP POP3 server allow you to check a control called *Disable plaintext logins for non-SSL connections*: if this control is checked, these servers will not allow people to login unless they first establish an SSL connection. The conventional wisdom on the Internet is that you should always enable this kind of refusal for unsecured logins, but this may be impractical if you have some users running mail clients that do not support SSL. We recommend strongly that you enable this option if you can do so practically.

Mailing list error handling

See also: Mailing lists, overview

If you've ever managed a large mailing list (200 or more addresses) you'll know that handling errors quickly becomes a significant problem: people change e-mail addresses without unsubscribing from the list, domains change their names, temporary DNS problems result in valid addresses bouncing for a day or two - all this and more results in tens or hundreds of error notifications every time a message is sent to the list.

Unfortunately, the way the Internet's mail protocols work make it hard to come up with automatic ways of handling this kind of problem that aren't also very expensive in terms of the bandwidth they use. In recognition of this, Mercury allows you to choose between three different ways of handling errors on your mailing lists.

1: Conventional error handling   When this method is chosen, a single address is supplied as the end point for errors in list mail. This address is placed in a special e-mail header called the *Return-Path*, which all responsible mail programs should use when returning error notifications. The result of this is that all "mail undeliverable" errors and other errors in sending mail to the list will be redirected to this one address, the idea being that a single person will be assigned the task of handling list errors manually. For smaller lists or lists where the personal touch is important, this approach usually works quite well, but it rapidly becomes unsustainable for larger lists.

2: VERP-based error handling   VERP stands for "Variable Envelope Return-path Processing": when using this method, every recipient in the list gets a separate copy of every message sent to the list, and in that copy of the message, a special version of the Return-path field is created that allows Mercury to work out the individual list and subscriber from any errors that get returned to it. Using this information, Mercury can automatically take certain actions when errors occur, such as setting the subscriber's entry to "NOMAIL" or deleting the subscription. Using VERP allows error handling to be almost entirely automated for a mailing list, but it is very "expensive" in the sense that it generates an individual message for every subscriber when a message is sent to the list. Even given the expense factor, however, VERP is often the only manageable way of handling larger mailing lists. *Note:* when using VERP error handling, any value you enter in the *"Explode submissions into x jobs"* field of the *Distribution* page of the Mailing List editor will be ignored - VERP-based mailing always generates a separate copy of every message for every subscriber on the list.

3: Hybrid error handling   This approach combines both conventional error handling and VERP-based error handling, and is a good compromise for medium-sized mailing lists. In hybrid mode, messages sent to the list are distributed normally and conventional error handling (see above) is used to field errors arising from the distribution. Combined with this, however, you can instruct Mercury that it should periodically send a specialized VERP mailing using technique (2) above: this VERP mailing is called a *probe*, the idea being that the probe will result in errors that will be handled automatically by Mercury. The advantage of Hybrid error handling is that most of the mail sent to the list will go out normally, allowing the usual economies of scale associated with list mail, but the periodic VERP probe will automatically catch and handle the majority of error conditions within a reasonable time frame. Mercury allows you to create your own template file and use that to create the probe message. This means that you can send out a monthly help guide to remind people how to manage their subscriptions and so on, and have that guide double as your VERP probe.

Error settings

Error handling scheme for this list's mail: Choose between *Conventional error handling*, *VERP-based error handling* or *Hybrid error handling* (see above for more details on the differences between these methods). The option you select will enable or disable certain other options in the dialog.

Errors go to (only in Conventional and Hybrid mode): Enter here the e-mail address to which any errors arising during delivery of mail to the list should be referred. In Hybrid mode, errors in normal mail distribution will use this address, but the periodic VERP probe will not.

Errors allowed before error handling occurs (only in VERP mode): Enter here the number of errors Mercury's VERP handler should allow on any individual address before taking action against it. By default, Mercury accumulates errors over the life of a subscription, but you can also tell it that it should zero its error count for a subscription on a weekly or monthly basis. It is up to you to decide how tolerant of errors you wish to be: if you set this field to zero, any errors at all will result in action being taken against the subscriber. When errors are accumulated on a weekly basis, each subscriber's error counter is reset on Sunday; similarly, when monthly accumulation is in force, each subscriber will have his or her error counter reset on the first day of the month.

When a subscription reaches the error limit... (only in VERP and Hybrid modes): Tells Mercury's VERP handler what action it should take against an address that generates too many errors. You can either suspend the subscription (which is the same as setting it to NOMAIL - the subscription is still there and can be reactivated, but will receive no postings from the list until specific action is taken) or delete the subscription. Note that if you choose to delete the subscription, no "Farewell" message will be sent to the subscriber, even if you have one defined for the list. In Hybrid mode, there is an assumed error limit of "1" - so, if your periodic VERP probe message results in an error, the action you define here is taken immediately. *Note:* if a subscription is suspended by this action it will remain suspended even if you tell Mercury to reset its error count periodically: once a subscription is suspended, reinstating it must be done manually, either by the subscriber or by a list moderator.

Mail a summary of VERP changes to moderators... (only in VERP and Hybrid modes): If you check this control, Mercury's VERP processor will keep a log of the changes it makes and will periodically send that log to the list's moderators. All moderators will receive the log, not just the list's primary moderator. Weekly summaries are sent each Sunday, while monthly summaries are sent on the first day of each month.

Enable sending VERP subscription probes (only in VERP and Hybrid modes): Checking this control tells Mercury to generate a periodic message to the list in VERP mode automatically. In Hybrid mode, any errors resulting from this probe message will be processed immediately; in VERP mode, this option is useful for lists that do not generate much traffic. By default, Mercury uses a fairly nondescript template file to generate the probe message, but you can specify your own template file if you wish - doing this allows you to send out a periodic reminder about the list, its subscription and unsubscription processes, etiquette rules or whatever and have that reminder do double duty as a VERP probe. If you wish to use your own template for the VERP probe, type a full path to the template file in the *using this template file* field. Weekly VERP probes are send on Wednesdays, while monthly VERP probes are sent on the fifteenth day of the month.

Transaction-level expression filters - syntax

See also: MercuryS Compliance settings

Each line in a transaction-level expression filtering file defines a test that MercuryS should apply at various stages of the SMTP transaction processing phase of mail delivery. A line describing an expression has the following general format:

```
<Operation>, <"Expression">, <Action>[Action]> ["Response"]
```

`"operation"` can be:

- `'H'` for an expression applied to the client's SMTP `HELO` greeting
- `'S'` for an expression applied to the subject line of the message
- `'R'` for an expression applied to each SMTP `RCPT` command
- `'M'` for an expression applied to the SMTP `MAIL FROM` command.

`"Expression"` is a Mercury regular expression - The expression must be quoted, and is applied to the entire HELO command. The following metacharacters are recognized in the regular expression:

`*`	Match any number of any characters
`?`	Match any single character
`+`	Match one or more occurrence of the last character
`[ ]`	Encloses a group of characters to match. Ranges can be specified in the group using '-'.
`/w`	Match zero or more whitespace characters
`/W`	Match one or more whitespace characters
`/c`	Toggle case sensitivity (case-insensitive by default)
`/s`	Toggle whitespace stripping - ignore all whitespace in the source
`/b`	Match a start-of-word boundary (including start-of-line)
`/B`	Match an end-of-word boundary (inicluding end-of-line)
/x	Toggle "ignore-non-alpha" mode - ignore non alphanumeric characters
/X	Toggle "ignore-spam" mode - ignore non-alphanumeric except @ and |

Note that the expression begins at the start of the source line, so if you want to match an expression anywhere within the line, you need to start and end the expression with closure characters (*): so, for example, if you wanted to search for the term `"pharmacy"` anywhere in the line you are checking, you would need to enter the expression as `"*pharmacy*"`.

`Action` is one or more characters indicating the action MercuryS should take when the expression is matched: the first character in the action can be one of the following:

- `'R'` to refuse the transaction
- `'D'` to drop the connection immediately with no error response
- `'L'` to log a system message
- `'B'` to issue an error response then drop the connection immediately.
- `'X'` to exit immediately from rule processing
- `'F'` to fail the current command without prejudice to future commands

The difference between the `'R'` and `'F'` actions is that the `'R'` action will result in all future commands during the session also being failed, whereas the `'F'` action fails only the particular command to which it applies. The `'F'` action is very useful for temporarily disabling certain mail addresses during mail storms, or for any other action where failing a command at the transaction level is useful.

The second character in the action string is optional and can have one of the following values:

`'S'` to blacklist the host for the next half hour

`"Response"` is an optional response code that MercuryS should return to the client (for the `'R'` operator) or the string to log as the system message (for the `'L'` operator). It must be quoted, and if it is returned as an error response to the client, then it must start with a 3-digit RFC2821 error response code (we recommend 554 for this).

Examples:

To detect and refuse any connection where the client tries to connect using your IP address as its HELO greeting (using 192.156.225.44 as your IP address in this example)

```
H, "*192.156.225.44*", R, "554 Get out of here, worthless scum."
```

To detect and refuse any attempt to deliver a message where the subject line contains the word `"Viagra"`:

```
S, "*viagra*", R, "554 Unacceptable subject line - message refused."
```

Note that in this case, Mercury will accept the entire message but will discard it. This costs you some bandwidth, but guarantees that "real" hosts that try to deliver such messages will return a proper error response to the sender.

To detect any message where the subject line contains the word `"Vicodin"` and drop the connection unceremoniously:

```
S, "*vicodin*", R, "'Vicodin' in subject line - connection dropped."
```

Note that dropping the connection is extremely abrupt and rude, and may result in some better-behaved hosts spending a lot of time retrying the delivery. You should only drop the connection in cases where you know that a virus or zombie system is attempting to send you information: such systems are usually very poorly-written and will be defeated by this technique (besides, who cares whether or not systems like this get upset by your action?).

Connection controls

The *Connection Control* section allows you to place restrictions on the hosts from which Mercury will accept connections, and to configure other module-specific capabilities based on the address of the connected host. A connection control entry can apply to a single address, or to a range of addresses. To add an entry to the list, click the *Add restriction* button; if you wish to create a restriction for a single address, enter that address in the "From" (left-hand) address field in normal dotted IP notation. To create a restriction for a range of addresses, enter the lowest address in the range you want to restrict in the "From" field, and the highest address you want to restrict in the "To" field. The addresses are inclusive, so both the addresses you enter are considered part of the range.

If you check the *Refuse connections* radio control, Mercury will not accept incoming connections from this address. Use this to suppress sites that are abusive or have been hijacked by spammers.

To edit a connection control entry, highlight it in the list, then click the *Change selection* button.

How Mercury applies connection control entries

The list of connection control entries you create can contain entries that overlap (i.e, entries that refer to addresses also covered by other entries). In the case of overlapping entries, Mercury uses the following method to select the entry it should use for any given address: if there is an entry that refers to the address on its own (not as part of a range), then Mercury will automatically use that entry; otherwise, it looks for the range that most closely encompasses the address and uses that.

Example: You have a *Refuse* entry covering the range from 198.2.5.1 to 198.2.5.128, and an *Allow* entry covering the range from 198.2.5.10 to 198.2.5.20: if a machine with the address 198.2.5.12 connects to Mercury, it will select the *Allow* entry to cover the connection, because the allow entry most tightly encompasses the connecting address (the range covers 11 addresses, where the Refuse entry's range covers 128 addresses).

MercuryB, the HTTP server

See also: Configuring MercuryB

MercuryB is a specialized implementation of the HTTP protocol used on the World-wide web. Unlike conventional HTTP servers, which are general-purpose tools used for things like home pages, order forms and information resources, MercuryB is service-oriented - it uses its own loadable module scheme to implement specific services within the Mercury environment, such as mailing list management and remote administration.

The first release of MercuryB includes only one loadable module, called MLSS (Mailing List Subscriber Services). This module allows mailing list subscribers to manage their mailing list subscriptions using a simple web-based interface. Future versions of Mercury will include mailing list services for mailing list moderation, remote server administration features, and ultimately a webmail service.

Configuring the MercuryB HTTP server

See also: Logging and session logging

Domain name for URLs MercuryB uses any value you enter here to form the base domain of URLs in the HTML documents it returns to clients. If you leave this field blank, MercuryB will use the value defined in the Mercury Core Module's "General" configuration page. You will usually enter a value here when a specific domain name that is different from the name usually assumed by Mercury needs to be used to access HTTP services.

TCP/IP timeout The length of time in seconds that MercuryB should wait for data on a connection before assuming that the connection is no longer valid and aborting it.

Default session timeout HTTP is usually a connectionless protocol - that means that a client connects, makes a request, then disconnects. The problem with this is that it is difficult to keep a "state" for the client - to remember the way things were the last time it connected. There are a number of ways of getting around this state-handling problem, and one of them is to allocate *tickets* for a session then allow the client to present that ticket the next time it connects; the ticket is effectively a reference to a structure maintained by MercuryB in which state information is stored. When using a ticketed approach, the internal structure must have a means of being discarded past a certain period of time - imagine, for example, that the client connects, then that machine is shut down without the user performing a "logout" operation to the HTTP server. The server is left carring a state structure for a connection that will never be accessed again. The value you enter here is the maximum period of time MercuryB will keep a session ticket alive before deallocating it. Service modules can allocate their own session timeout values as required - this value is merely the default. We recommend a value between 10 and 30 minutes for this.

Listen on TCP/IP port The TCP/IP port is the socket into which the HTTP client "plugs" to access MercuryB's services. The standard port for HTTP services is 80, but in some cases (particularly if you use a proxy server, or you already have a web server running on the machine) you may need to change this. Consult your ISP or network administrator to find out if you need to use an alternative HTTP port. If you are unsure, leave it set to 80.

IP Interface to use If your computer supports multiple IP interfaces, you can use this field to tell MercuryB which interface it should select when listening for connections: enter the interface as a dotted IP address in the general form *www.xxx.yyy.zzz*. As an example, your system may have one IP address assigned to a dialup PPP connection, and another, different IP address assigned to a local Ethernet network - you would enter here the interface MercuryB should use. If you leave this field blank, MercuryB will listen on all available interfaces. Unless you are *very* sure of what you are doing, or have been instructed by an ISP or network administrator, you should leave this field blank. If you change the IP interface in this field, you must restart Mercury before the new interface number will be used.

Connection control

The Connection Control section allows you to place restrictions on the hosts from which MercuryB will accept connections, and to configure certain capabilities based on the address of the connected host. A connection control entry can apply to a single address, or to a range of addresses. To add an entry to the list, click the *Add restriction* button; if you wish to create a restriction for a single address, enter that address in the "From" (left-hand) address field in normal dotted IP notation. To create a restriction for a range of addresses, enter the lowest address in the range you want to restrict in the "From" field, and the highest address you want to restrict in the "To" field. The addresses are inclusive, so both the addresses you enter are considered part of the range.

If you check the *Refuse connections* radio control, Mercury will not accept incoming connections from this address. Use this to suppress sites that are abusive or have been hijacked by spammers.

To edit a connection control entry, highlight it in the list, then click the *Change selection* button.

How Mercury applies connection control entries

The list of connection control entries you create can contain entries that overlap (i.e, entries that refer to addresses also covered by other entries). In the case of overlapping entries, Mercury uses the following method to select the entry it should use for any given address: if there is an entry that refers to the address on its own (not as part of a range), then Mercury will automatically use that entry; otherwise, it looks for the range that most closely encompasses the address and uses that.

Example: You have a *Refuse* entry covering the range from 198.2.5.1 to 198.2.5.128, and an *Allow* entry covering the range from 198.2.5.10 to 198.2.5.20: if a machine with the address 198.2.5.12 connects to Mercury, it will select the *Allow* entry to cover the connection, because the allow entry most tightly encompasses the connecting address (the range covers 11 addresses, where the Refuse entry's range covers 128 addresses).