MailCorral Documentation

Eric Wilde

Table of Contents

Table of Contents

0. Preface

Eric Wilde, ewilde@bsmdevelopment.com

V2.1.2, 2013 Jul 21

This document shows how to install the sendmail filter, MailCorral, into sendmail as a milter. It describes how to configure sendmail, compile the filter and set it up to run attended. Information about the operation of the filter is also provided.

0.1 Copyright

0.2 Distribution

This document is also available in HTML format at:

MailCorral Documentation

0.3 Contributions

BSM Development is a small company of software developers (perhaps, like yourself) and we would appreciate, if you so choose, a small contribution to pay for this software. We hope you perceive it to be of value and find it useful to you. As a suggestion, we would like to think that it is worth at least US $100.00. However, you may send whatever amount that you feel this product is worth. If you'd like to make a contribution, you can do so directly via the contributions page, of the BSM Development site (www.bsmdevelopment.com/Products/Contribute.html).

Alternately, you can send a check via snail mail to:

BSM Development
44 Whitewood Circle
Norwood, MA 02062

Please make your check payable to "BSM Development", thanks.

0.4 Change Log

Here are the version numbers, dates and a brief description of each program change, in reverse chronological order:

2.1.2

- 2013 Jul 21 - Fix possible "sed: command not found" error in configure script on some systems.
- 2013 Jan 27 - Add the ability to process the SpamAssassin updates tree, to find versioned whitelists.
- 2013 Jan 25 - Fix compile warnings for the free function in configuration processing. Fix reentrancy problem with user configuration files.

2.1.1

- 2012 Jun 8 - Fix compile warnings for string/stdio functions in expression evaluator and other modules. Add image/png MIME type and .png extension to the list of acceptable mail attachements. Fix bogus definition of REMAILHDRBYPASS. Fix longstanding problem with mail from local addresses being erroneously filtered when the CheckExternal flag is not set.

2.1.0

- 2009 Nov 21 - Add .wmv extension to the file types list as an OK file.
- 2009 Nov 15 - Fix problem in with WeakRequire when module not found in MailRelease and SpamNotify.

2.0.1

- 2009 Aug 7 - Fix quote handling by NormalizeName.
- 2009 Aug 3 - Fix problem writing empty message body to spam file when spam messages also contain viruses and the virus scanner deletes the entire message.

2.0.0

- 2009 Feb 4 - Add encoding and decoding of uuencoded entities.
- 2009 Jan 19 - Add support for ClamAV via clamd.
- 2009 Jan 16 - Add support for the spamd 1.4 protocol and the HEADERS command.
- 2009 Jan 8 - Add archiving of messages to disk or to a third party archiver via email forwarding. Add bounceback suppression for bounced messages from non-functioning third party archivers. Use header from name for blacklist and whitelist checks, along with envelope from name. Have name normalization strip "<>" and "()" from addresses.
- 2004 Apr 19 - Add support for evaluating results from user supplied arbitrons, plus internal arbitrons, via logical expressions.
- 2004 Jan 15 - Allow users to programatically supply arbitrons.

1.3.0

- 2007 Aug 13 - Fix problem with multiply defined bool under sendmail 8.14.1.

1.2.0

- 2003 Aug 21 - Fix problem with HTML tag parameters that are a single quote crashing filter (filter goes defunct).
- 2003 Jun 13 - Try "htdocs" as well as "html" for the template file location in ConfigEdit.cgi.
- 2003 Jun 8 - Add exit message if errors found in command line or options.
- 2003 May 29 - Treat "text" as "text/plain". Fix crash when internal spam arbitron is used.
- 2003 May 15 - Allow all_spam_to for non-local users.

- 2003 May 12 - Fix problem with messages containing empty text entities.
- 2003 May 5 - Remove HTML tags from names to release in MailRelease.
- 2003 May 2 - Tweak MailRelease error responses and add a couple more messages. Fix apparent Perl bug in MailRelease whereby push of lc($1) pushes garbage. This affected domain lookup.

1.1.3

- 2003 May 1 - Conditionally load Zlib and DB_File, for real this time, in MailRelease and SpamNotify.
- 2003 Apr 26 - Have SpamNotify unzip corralled messages, if necessary (just in case someone zips all the spam in the corral, before we have a chance to send out the notifications). Fix problem with '>' and '<' in the HTML component of MailRelease notifications.

1.1.2

- 2003 Apr 24 - Add an option to keep all messages in the corral, regardless of whether they were altered or not. Add the ability to zip older corralled messages, if desired.
- 2003 Apr 17 - Include the latest SpamNotify in the distro.
- 2003 Apr 14 - Add global ignore options.
- 2003 Apr 13 - Allow source RPM to build on any i386 version of RedHat. Template redhat.mc is missing on RH 8.x. The RPM now picks something else.
- 2003 Apr 11 - Update filterclean to get rid of empty partition directories.
- 2003 Apr 10 - Fix compiler windge about ctime pointer.

1.1.1

- 2003 Apr 8 - Don't reencode second HTML entity if not changed. Allow buffer space for badly formatted Quoted-printable entities.
- 2003 Apr 3 - Use tag delimiters to end mismatched quotes.
- 2003 Apr 2 - Handle mangling of short MIME types properly. Ignore "Content-type: text" in headers.
- 2003 Mar 31 - Fix problem with addresses containing domain names before the '@'.
- 2003 Mar 27 - Check messages from everybody, if paranoid.

1.1.0

- 2003 Mar 22 - Fixed a problem in HTML tag scanning when unmatched quotes are encountered.
- 2003 Mar 19 - Rebuild messages with attachments only to allow virus and spam warning text to be inserted, permit deletion of the attachment, etc.
- 2003 Mar 17 - Handle Apple resource forks better.
- 2003 Mar 16 - Allow attachments containing viruses to be deleted.
- 2003 Mar 12 - Partition spam as well.
- 2003 Mar 11 - Properly handle MIME type messages that have empty bodies.
- 2003 Mar 10 - Fix improper use of HAVE_DBM ifdef.

1.0.18

- 2003 Mar 9 - Implement ommitted cases in non-delivery test. Test for Base64 encoded text/HTML MIME entities in internal, statistical arbitron. Add text/plain and text/html as special cases for "UnknownDisposition MIME". Remove junk after plussed user names.
- 2003 Mar 8 - Allow user to specify timeout for arbitron connection.

- 2003 Mar 7 - Use VerifyUser instead of getpwnam to look up home directories, thereby avoiding a crash due to a threads bug.

1.0.17

- 2003 Mar 4 - Lowercase external addresses used to create corral files.
- 2003 Mar 3 - Fix up the "mailto" link in the remail message so that it is really a link that works.
- 2003 Mar 1 - Allow IP addresses to be supplied in the local domain list. Accept full reports from SpamAssassin versions greater than or equal to 2.50. Make user name check optional in MailRelease. Allow MailRelease to release corralled mail without path and partition name.
- 2003 Feb 28 - Add options to include SMTP AUTH and other special classes of users in the local domain (for purposes of internal -> internal/external determination). Remove the background parameter from the HTML body tag.
- 2003 Feb 27 - Add options to filter mail transiting the system. Change the handling of unknown file types so they are disposed of according to the user's preference.
- 2003 Feb 26 - Allow statistical filter to be tuned through config file options.

1.0.16

- 2003 Feb 19 - Remove certain HTML escape sequences.
- 2003 Feb 17 - Fix base 64 re-encoding problem for templates with no line feeds (was causing occasional crash on badly-formed encoded messages). Fix problem with HTML tag quoting algorithm.
- 2003 Feb 16 - Allow enough buffer space for badly formatted Base 64 encoded messages to be modified.
- 2003 Feb 13 - Create a man page for sendmailfilter.
- 2003 Feb 12 - Split message and deliver altered/unaltered copies to those who request virus filtering and those who don't.
- 2003 Feb 9 - Allow filtered viruses to be partitioned in the corral. Delete recipients from the envelope, if they request non-delivery of spam.
- 2003 Feb 7 - Cache /etc/passwd entries to avoid crash due to threads bug.
- 2003 Feb 6 - Add statistical spam fast path checks.
- 2003 Feb 4 - Remove HTML comments embedded in the middle of words.
- 2003 Jan 30 - Use proxy userid for DBM lookup if no options found under regular lookup.
- 2003 Jan 20 - Include sample sendmail.cf file in distro.
- 2003 Jan 19 - Check for extra options on command line. Make spam arbitron error reportage more comprehensive.
- 2003 Jan 16 - Save PID and initial command string in PID file at startup.

1.0.15

- 2003 Jan 12 - Allow virus/spam tags to be defined in smfopts.h. Add options for ignoring spam and/or virus filtering.
- 2003 Jan 11 - Don't corral spam for nonexistant users.
- 2003 Jan 10 - Use the virtual user table to determine local users. Only process spam if the virtual users are local users.
- 2003 Jan 7 - Convert the spam arbitron IP address ahead of time to avoid a crash in gethostbyname on BSD.
- 2002 Dec 24 - Bail out if invalid options are detected on the command line or in the global config file.

1.0.14

- 2002 Dec 15 - Fix crash introduced by language support. Fixed loop caused by messages with bogus initial MIME separators.
- 2002 Dec 13 - Disable possible exploit through the "name" parameter of the "Content-Type" header. Fixed bug that caused a memory overrun when mangling certain file names.

1.0.13

- 2002 Dec 7 - Use flock simulator on systems with no flock.
- 2002 Dec 6 - Fixed problem with overly-greedy regular expression not updating repeating fields greater than nine in ConfigEdit.cgi.
- 2002 Nov 26 - Sum user options if requested to do so.
- 2002 Nov 24 - Accept the ".csv" attachment type.
- 2002 Nov 23 - Use the "X-Accept-Language" tag to set the language used for message inserts.
- 2002 Nov 22 - Handle blanks, etc. in to/from names.
- 2002 Nov 18 - Find attachments in malformed MIME entities.
- 2002 Nov 14 - Improved message insertion for encoded mail.
- 2002 Nov 13 - Added support for HTML tag filtering by parameter. Allow innocuous parameters for the <body> and <meta> tags.
- 2002 Nov 12 - Changes to support Solaris properly during configuration. Fixed problem with percent signs in addresses causing syslog to crash the filter.
- 2002 Nov 8 - Fixed possible exploit through attachments defined directly in message headers.
- 2002 Nov 6 - Allow the <html> tag to have parameters (for Outlook's sake).
- 2002 Nov 5 - Strip pesky <Ctrl-M> from user names in MailRelease. Handle long file names wrapped by mail readers. Clear BASH_ENV in case running suidperl.
- 2002 Oct 21 - Fixed problem with comments being replaced in poorly formed messages.

1.0.12

- 2002 Oct 18 - Fix local AutoRemail option having no effect.
- 2002 Oct 18 - Include all SpamAssassin response headers in reports, not just X-Spam-Status and X-Spam-Checker-Version.
- 2002 Oct 16 - Add missing move of X-Tag options to local.
- 2002 Oct 13 - Use fully qualified names for config database lookup if "-q" or "QualifyNames" is used. Add the Language, LanguageDirectory and "-L" options.

1.0.11

- 2002 Oct 9 - Accept ISO encodings in MIME file name parameters.
- 2002 Oct 7 - Added generic config file editor, supporting flat files and DBM files for storing user parameters.
- 2002 Oct 5 - Allow config lookup via DBM database.

1.0.10

- 2002 Sep 27 - Remove leading space from domain names, since people appear not to be able to read the documentation. Fix problem with fputs returning different return code under BSD. Updated list of HTML tags to allow most innocuous ones. Download is now available in RPM form for RedHat Linux.
- 2002 Sep 26 - Fix errno being reported incorrectly as zero.
- 2002 Sep 24 - Change the corral to /var/spool/MailCorral. Create the corral directory if it doesn't exist.

- 2002 Sep 23 - Use the first local recipient for local options processing.

1.0.9

- 2002 Sep 21 - Force add of "localhost" to domain list, if not already present.
- 2002 Sep 18 - Fix problem with "-r" and "-rm" introduced by the adding of options as keywords.
- 2002 Sep 17 - Removed a possible exploit that relies on HTML tags being split across two pieces of an attachment.
- 2002 Sep 14 - Switch to wholly percentage based spam determination because Spam Assassin returns the wrong flag value when SQL preferences are used.

1.0.8

- 2002 Sep 10 - Added option to turn off spam fast path. Fixed problem with HTML reports, sent to postmaster, not having MIME headers. Fix handling of headers in responses from spamd 1.3 and up.
- 2002 Sep 9 - Allow all local options to be supplied on a per-user basis from a local config file. Allow local options to turn off global options. Add a protocol type for the internal fast path spam arbitron.
- 2002 Sep 8 - Add message options to allow tailoring of text inserted into filtered mail. Allow options to be continued on multiple lines.
- 2002 Sep 7 - Allow all command line options to be specified as keywords in config files. Allow all options to be supplied in a global config file.
- 2002 Sep 6 - Add flockfile/funlockfile around writes to debug file to reduce overhead caused by locking in low-level I/O routines. Fix random crashes when writing to logfile under heavy threading.
- 2002 Sep 4 - Add queue ID to log entries to assist in debugging when multi-tasking.

1.0.7

- 2002 Sep 2 - Added MailRelease to allow filtered messages to be remailed.
- 2002 Sep 1 - Fix crash when processing straight message/rfc822 types (i.e. not MIME). Allow filtered messages to be automatically remailed.

1.0.6

- 2002 Aug 27 - Added the domain names option and domain names file support.
- 2002 Aug 26 - Add support for spamd 1.3 and the INFO command.

1.0.5

- 2002 Aug 22 - Update list of filtered file extensions. Add flag for checking external deliveries too. Add flag for spam reporting always. Add flags for envelope and version info in the headers.
- 2002 Aug 21 - Remove parameters from <body> tag when laundering HTML. All good/bad HTML tags are now configurable. Allow more innocuous HTML tags to pass through unscathed. Only open save and work files if really necessary to process message, instead of on spec.
- 2002 Aug 19 - Added bad file type ".mhtml". Added good file types ".p7c", ".p7m" and ".p7s". Added good MIME types "application/pkcs7-mime" and "application/pkcs7-signature". Fixed problem with named text being marked as an attachment when certain mail readers treat it as part of the message. Fixed a problem with cleanup during abort, crashing if no recipient seen.
- 2002 Aug 17 - Use received header date for spam tag instead of date header (which can be omitted). Format sendmail socket address for debugging.
- 2002 Aug 15 - Fixed problem with unterminated HTML tags not being laundered properly.

1.0.4

- 2002 Aug 14 - Eliminated erroneous spam check when plain old mail messages are remailed. Added fast path spam lookup for white and black lists.
- 2002 Aug 12 - Fix missing "[SPAM]" tag, if message delivery is bypassed for reasons other than spam. Fix problems with malformed MIME entities that have no trailing separator.
- 2002 Aug 10 - Added support for Spam Assassin option files.
- 2002 Aug 9 - Added debug levels.

1.0.3

- 2002 Aug 6 - Added options for spam delivery modes ("-sc", "-sd", "-st") and reporting types ("-s1", "-s2", "-s3"). Added support for spam reporting levels two and three for Spam Assassin. Fixed problem with inserted text when all text/plain and text/html components of a message are encoded base64.
- 2002 Aug 5 - Added <iframe> (the favorite tag of virus professionals everywhere) to the list of HTML tags that are always removed, regardless of the "-h" flag setting. Remove <cr><lf>s from message subjects, per bug in Outlook that allows virus launcers in subjects.
- 2002 Aug 4 - Move spam processing to spamfilter.c. Fixed memory leak when spam checking plain old mail messages. Improved spam checking performance for multipart MIME types.

1.0.2

- 2002 Aug 3 - Fix problem with corralled spam only being held for the last recipient in a list of recipients, when multiple recipients passed to SMTP for a single messge. Problem occurred when envelope was addressed to more than one user. Spam would be detected, processed and corralled but only one notification was sent to a single recipient (last one on the envelope), not all of the recipients.
- 2002 Aug 2 - Allow multiple domains in local domain list for "-i" option, etc.
- 2002 Aug 1 - Add "-d" (debugging option) for dynamic debugging.

1.0.1

- 2002 Jul 29 - Add the "-q" (qualify recipient names in the spam corral) and "-u" (supply a proxy userid for filter criteria lookup) options.

1.0

- 2002 Jul 12 - Add spam filtering.
- 2002 Jun 20 - Add decode of MIME components for HTML check.
- 2002 Feb 25 - Initial coding.

1. Installation

The steps to installing the MailCorral filter on a Linux or Unix system, running sendmail, are:

1. Check that your system meets the requirements.
2. Rebuild sendmail to use milter, if you haven't already done so.
3. Compile and install the sendmail filter program.
4. Configure sendmail to invoke milter.
5. Hack the sendmail startup script to start the filter.
6. Install the optional message remailer, if you wish to use it (you will need to create a message handler robot and configure sendmail to allow forwarding to work).
7. Install the optional spam notifier, if you wish to use it.
8. Set up a cron job to periodically clean up the filter files, etc.
9. Configure any local options in the system and/or user configuration files.
10. If you are installing the RPM for your system, you can skip most of these steps and go straight to the RPM install.

1.1 Requirements

Before beginning with the installation of MailCorral, you might want to check that your system meets the requirements listed below:

- Requires, at a minimum, sendmail 8.12.
- Runs under the sendmail milter interface, which must be installed and enabled.
- Unless you are installing one of the RPMs, you must be able to compile the source (needs a C compiler such as GCC).
- Enhanced spam detection can work in conjunction with third party spam recognition packages (e.g. SpamAssassin), which must be pre-installed.
- Enhanced virus detection can work in conjunction with third party virus recognition packages (e.g. ClamAV), which must be pre-installed.
- Sending and receiving of encrypted email requires that the GnuPG library be pre-installed.

You should note that MailCorral was developed on a Linux system. If your system is something other than Linux or Unix, you might want to investigate whether the sendmail/milter configuration is similar and whether standard Unix-type software will run on your machine. The sendmail filter does nothing particularly special so it has a good chance of working on other systems, but you never know.

1.1.1 Install the Optional Prerequisites

If you will be using any of the optional arbitrons or will be sending/receiving encrypted email messages, install those optional components noted below, according to their directions, and make sure that they are working (e.g. that spamd and/or clamd are up and running) before proceeding with the steps outlined herein. Once again, the optional components are:

- The third party spam recognition package SpamAssassin) provides the daemon spamd.
- The third party virus recognition package ClamAV) provides the daemon clamd.
- The third party encryption/decryption library GnuPG is used to send and recieve encrypted messages.

1.2 Rebuild Sendmail

Before you can proceed with filtering mail messages, you need a version of sendmail that supports milter (version 8.12.0 and above). You can get sendmail from www.sendmail.org.

If you haven't already done so, compile sendmail with the flag "-DMILTER" set. If it was already compiled without it, add the following line to ./sendmail/devtools/Site/site.config.m4:

APPENDDEF(`conf_sendmail_ENVDEF', `-DMILTER')dnl

If you are starting from scratch, make sure that this line appears in that file before you start.

If necessary, delete the entire sendmail object directory named for your operating system (e.g. ./sendmail/obj.Linux.2.2.16-22.i586/sendmail) and rebuild using "sh Build". Don't forget to install the new sendmail into your system by issuing a "sh Build install" from the top sendmail directory.

Switch to the libmilter directory and build "libmilter.a". This is required before the sendmail filter can be built. "sh Build" from that directory should do the trick.

1.3 Compile the Filter

1.3.1 Untar the Distribution

You should untar the distribution files to a separate directory where the filter can be built on its own. By default, the tar ball has all of the distribution files in a subdirectory called "MailCorral-x.x.x". We suggest that you put the filter in a subdirectory off of the main sendmail directory. For example, if you have a main sendmail directory that looks something like this:

```
SendMail
-rw-r--r--  1  718646 Jul 21 12:13 MailCorral-2.1.2.tar.gz
-rwxr--r--  1  290418 Sep 19  2001 sendmail-8.11.6-1.7.1.i386.rpm
drwxr-xr-x 22    4096 Sep 19  2001 sendmail-8.12.0
-rwxr--r--  1 1783911 Sep 14  2001 sendmail.8.12.0.tar.gz
```

You could cd to this directory and unpack the tar ball with one of the following commands:

tar -xvzf MailCorral-2.1.2.tar.gz
gunzip -c MailCorral-2.1.2.tar.gz | tar -xv

If you do so, your directory structure will now look something like this:

```
SendMail
drwxrwxr-x  2    4096 Jul 21 17:49 MailCorral-2.1.2
-rw-r--r--  1  718646 Jul 21 12:13 MailCorral-2.1.2.tar.gz
-rwxr--r--  1  290418 Sep 19  2001 sendmail-8.11.6-1.7.1.i386.rpm
drwxr-xr-x 22    4096 Sep 19  2001 sendmail-8.12.0
-rwxr--r--  1 1783911 Sep 14  2001 sendmail.8.12.0.tar.gz
```

1.3.2 Run "configure"

The distribution is meant to configure itself using a shell script produced by the GNU autoconf program. To configure the makefile which will build the filter, switch to the installed directory and type "./configure". If you're using "csh" on an old version of System V, you might need to type "sh ./configure" instead to prevent "csh" from trying to execute "configure" itself.

The "configure" shell script attempts to guess correct values for various system-dependent variables used during compilation. If you followed the notes above about where to install the filter, running "configure" is all you need do. If you have more than one version of sendmail or you put the filter in a directory not in the sendmail tree, you'll need to supply the name of the sendmail source directory. To do this, type:

./configure --with-sendmail=/home/joeblow/sendmail/sendmail-8.12.3

and replace the sample path name with the actual pathname for the sendmail source tree.

Running "configure" takes awhile. While running, it prints some messages telling which features it is checking for.

The "configure" shell script creates a shell script "config.status" that you can run in the future to recreate the current configuration, and a file "config.log" containing compiler output (useful mainly for debugging "configure"). It can also use an optional file (typically called "config.cache" and enabled with "--cache-file=config.cache" or simply "-C") that saves the results of its tests to speed up reconfiguring.

1.3.3 Specifying the System Type

There may be some features "configure" cannot figure out automatically but needs to determine for the type of machine the sendmail filter will run on. Usually, assuming the filter is being built to run on the same architecture as the build is done on, "configure" can figure that out but if it prints a message saying it cannot guess the machine type, give it the "--build=*type*" option. *type* can either be a short name for the system type, such as "sun4", or a canonical name which has the form:

cpu-company-system

where *system* can have one of these forms:

os or *kernel-os*

See the file "config.sub" for the possible values of each field.

If you want to use a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e. the platform on which the generated programs will eventually be run) with "--host=*type*".

1.3.4 Sharing Defaults

If you want to set default values for "configure" scripts to share, you can create a site shell script called "config.site" that gives default values for variables like "CC", "cache_file", and "prefix". "configure" looks for "PREFIX/share/config.site" if it exists, then "PREFIX/etc/config.site" if it exists. Or, you can set the "CONFIG_SITE" environment variable to the location of the site script. A warning: although this "configure"

script does so, not all "configure" scripts look for a site script.

1.3.5 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to "configure". However, some packages may run "configure" again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the "configure" command line, using "VAR=*value*". For example:

./configure CC=/usr/local2/bin/gcc

will cause the specified gcc to be used as the C compiler (unless it is overridden in the site shell script).

To find out what options "configure" recognizes and how to control it, type "./configure --help" for more details.

1.3.6 Altering the Makefile

If you need to hack the Makefile that is generated by "configure", here is a description of its variables:

SRCDIR	The source directory containing the sendmail filter source and all of its related files. The filter will be built in this directory.
SENDMAILDIR	The directory where the source for the version of sendmail that is being used is found. Usually, this is relative to the current directory (above it), but an absolute path may also be given.
OS	The name of the directory, off of the sendmail directory, where the OS-specific version of sendmail is built (look around, you'll find it). It is this directory that you delete when you need to rebuild sendmail, by the way. If you are cross-compiling the filter, there may be more than one directory for the different operating systems and hardware platforms that are targeted. Configuration attempts to pick the right one, based on the target.
BINDIR	The binary directory where sendmail and the filter are installed into the system.
SMINIT	The sendmail initialization script that is used to stop and start sendmail.
CONFIGDIR	The directory where the sendmail configuration files are found (e.g. /etc/mail).
MANDIR	The directory where the man pages are to be installed (e.g. /usr/share/man).

1.3.7 Altering Fixed Constants and Variables

The sendmail filter uses a number of more or less fixed constants that are set in a header file that is included in the filter modules when they are compiled. All of these constants will probably only need to be changed once, if at all, to reflect your installation preferences. The header file is called "smfopts.h". It contains system specific information, the texts of all error messages generated, lists of all MIME types and attachments filtered, etc.

The first time you run configure to install the sendmail filter, "smfopts.h" is created in your source directory.

You are free to hack it to your heart's content. When you install subsequent releases of the sendmail filter into the same source directory, "smfopts.h" will not be overwritten. Rather, a separate file will be created called "smfopts.h.new" that will contain the new header file. If any changes have been made to the header file, you will be issued a warning at compile time and instructed to compare the two files. This can be done with diff, for example:

diff smfopts.h.new smfopts.h

Diff will show you what changes you need to make to the your "smfopts.h" to bring it up to date with the latest version. Be advised, however, that you need not update "smfopts.h" to the latest version. You can safely ignore the warning message, since "smfopts.h" has a default file ("smfopts_def.h") which contains all of the latest constants, etc. Anything missing from "smfopts.h" will be supplied by "smfopts_def.h" so that the compile will still proceed. However, you will probably want to keep you copy of "smfopts.h" up to date to prevent the chance that the mail filter's behavior will not be quite up to snuff.

When you are hacking "smfopts.h", note that we use a setting of five for tabs so the source will look weird unless you set your tabs to five also). Here is a list of the constants and variables that you might want to change:

TECHSUPPORT	The name of the person whom users should contact if they need assistance with filtered messages.
REMAILROBOT	The username of the message remailer robot. This name will be used in the instructions to recipients of modified messages, telling them how to get the unmodified message mailed to them. Essentially, they send a message to the username given here.
POSTMASTER	The name of the person to whom warning messages should be emailed if any viruses or unknown filter types are found. Don't define this, if you don't want these messages.
DOMAIN	The name or names of the local domain. Used for detecting whether a message is being mailed locally so that virus filtering may be bypassed or not (see the "-e" and "-i" options). May be a single domain name or a list of domain names, separated by commas. It may also include file names. Any name that begins with a '/', '~' or '.' is assumed to be a file name. If a file name is given, the file should contain a list of local domain names, one per line. Blank lines and comments beginning with '#' are ignored. Typically, this feature would be used to point to the file "/etc/mail/local-host-names" or the same place where sendmail gets local domain name information from. The domain name "localhost" is forced into the list at the end, if it isn't specified. This name is always assumed to be a local domain name by sendmail. The "-D" command line option or the "DomainList" config file parameter may also be used to specify domain information. If either is used, it overrides the value compiled in by this variable.
SUBJTAGVIRUS	A tag string that will be added to the front of the subject for any messages that might possibly contain a virus. Used to identify that message for special

	processing by any mailer receiving it. The default is "[VIRUS] ".
SUBJTAGSPAM	A tag string that will be added to the front of the subject for any messages that contain spam. Used to identify that message for special processing by any mailer receiving it. The default is "[SPAM] ".
SPAMKEY	A key string used to generate spam bypass tags. Pick a string (up to eight characters). Whoever has this key string may be able to pass spam through your system without checks so don't let it be known what it is.
SPAMHDRBYPASS	A mail header used to pass spam bypass tags. You can change this name to make it harder for a spammer, should they crack your key string (above), to pass spam. This header is temporarily added to messages sidelined because of spam, to allow them to be remailed without filtering at a later date.
REMAILHDRBYPASS	A mail header used to pass remail bypass tags. You can change this name to make it harder for an outside user, should they crack your key string (above), to bypass message filtering. This header is temporarily added to filtered messages, stored in the corral because they were altered to remove a virus. It allows them to be remailed without filtering at a later date.
SPAMHDRSTATS	A mail header used to hold spam statistics for those messages that contain spam. All remailed spam will have this header, which gives the sendmail filter's reasons for tagging the message as spam. The use of third party spam arbitration packages may cause other headers to be added.
SPAMSTATSTPL	A template that defines how the spam statistics, in the spam statistics header, are formatted for display.
VIRUSHDRSTATS	A mail header used to hold virus statistics for those messages that contain viruses. All remailed messages with viruses will have this header, which gives the sendmail filter's reasons for tagging the message as containing a virus. The use of third party virus arbitration packages may cause other headers to be added.
VIRUSSTATSTPL	A template that defines how the virus statistics, in the virus statistics header, are formatted for display.
MSGEVALERR	A message that is used to inform the user of any evaluation errors that occur in the logical expressions that they supply for evaluating the results of spam and virus checks.
ARBITRONTIMEOUT	The default number of seconds that the filter should wait for any external arbitrons (both third party and programmable) to return a decision about whether the portion of the message that it was asked to scan contains a virus or is spam or not.
XENVFROMHDR	A mail header that is added by the "-Xenv" option to give the name of the sender, as passed to sendmail on the envelope.
XENVTOHDR	A mail header that is added by the "-Xenv" option to give the names of all of the recipients, as passed to sendmail on the envelope.
XVERSIONHDR	A mail header that is added by the "-Xver" option to show the name of the sendmail filter and its version number.
MAXADDRESSLENGTH	

	The maximum length of a mail address that will be handled by the filter. The current value of 255 should be sufficient.
MAXSUBJECTLENGTH	The maximum length of a message subject that will be handled by the filter. The current value of 255 should be sufficient.
MAXDATELENGTH	The maximum length of message date stamps that will be handled by the filter. The current value of 32 should be sufficient.
CORRAL_DIR	The directory where all of the corralled messages and spam will be placed (per the templates below). You can change the path name but be careful that the path name and substituted text in the templates does not come to more than 255 bytes.
RECV_NAME_TEMPLATE	The template used to determine where a saved copy of the actual message received, for any filtered messages, should be placed and how it should be named. You can change the template but be careful that the path name for the corral directory (above) and substituted text in the sprintf part does not come to more than 255 bytes. Also, be certain to allow for three components to be substituted into the name (i.e. leave the "%s" and "%X" components there in some form).
SPAM_NAME_TEMPLATE	The template used to determine where saved copies of any spam messages received should be kept and how they are named. Same rules as for RECV_NAME_TEMPLATE.
REPL_NAME_TEMPLATE	A template used to make up lock file names for replies to spammers. Same rules as for RECV_NAME_TEMPLATE.
DEBUG_FILE_NAME	The name of the file where debugging information is written. Defining this name causes debugging information to always be written to this file. For transient debugging, the "-d" command line option or the "DebugFile" config file parameter is available.
DEBUG_SENDMAIL	A flag that can be defined to debug sendmail itself, as well as the filter.
PROGNAME	The name of the sendmail filter, used for error reporting.
PIDFILE	The name of the PID file that is used to track the filter's running status.
GLOBALOPTIONS	The name or names of the global sendmail filter configuration file. The names are searched in descending order and the search stops at the first one that is found.
LOCALOPTIONS	The name or names of the local sendmail filter configuration file. The names are searched in descending order and the search stops at the first one that is found. The character '~' signifies the home directory of the recipient of the message.
chMsg*xxx*	All of the variables that begin "chMsg" contain the text of the various messages that the filter inserts into filtered mail messages. You can change these messages to read however you'd like but please be aware of any substitutions into them and retain the same number of substitutions in your replacement message text.
AcceptableTypes	This variable defines the table of acceptable MIME types that can appear inline in a message. All others are considered unacceptable (by virtue of the

	fact that they might conceivably contain a virus).
FilteredFiles	This variable defines the table of file extensions which are examined for viruses and the actions to be taken if they are found in a message.
RejectedHTMLs	This variable defines the table of dangerous HTML tags that will always be laundered out of a message, regardless of where they are found. Note that many mail readers will interpret these tags, despite their being found in non-HTML messages, hence the reason for their removal.
AcceptableHTMLs	This variable defines the table of harmless and therefore acceptable HTML tags that will never be laundered out of a message. The implication here is that, if a tag isn't found on this list, it will be laundered out. This list only applies inside <html> and </html> tag pairs.

1.3.8 Compiling and Installing

After making any changes to the makefile and "smfopts.h", run make to compile the filter. You can do this as "JoeUser" because there is nothing special taking place at this step. Simply type "make" in the install directory.

To install the filter, become super-duper user and type "make install". This will also restart sendmail which will have no effect until you hack its startup script (in Section 1.5). However, if you make future changes to the filter, "make install" will install it and restart it.

Once you have installed the filter, you can remove the program binaries and object files from the source code directory, if you wish, by typing "make clean".

1.4 Configure Sendmail

Hack the mc file for "sendmail.cf" (this is way better than hacking "sendmail.cf" directly, which is definitely not for the faint of heart). Start by copying the canned configuration supplied with the OS, if you haven't already done so. For example:

```
cp /usr/lib/sendmail-cf/cf/redhat.mc /usr/lib/sendmail-cf/cf/mysys.mc
```

Hack /usr/lib/sendmail-cf/cf/mysys.mc, and make sure the following lines are present or are added before the "MAILER" definitions:

```
define(`_FFR_MILTER', `1')dnl
INPUT_MAIL_FILTER(`filter1',`S=inet:2526@localhost, F=R')dnl
define(`confINPUT_MAIL_FILTERS',`filter1')dnl
```

The port number (2526) is the port that sendmail will use to talk to the filter on. You can change it to whatever you like, so long as you also change the port number used to start the filter (see Section 1.5).

Build a new "sendmail.cf" by running the M4 macro processor over the ".mc" file. You can use the following commands.

```
m4 mysys.mc >mysys.cf
cp mysys.cf /etc/sendmail.cf
```

Note that the default values for timeouts between sendmail and the filter are sometimes too small (i.e. 10 seconds). If you experience problems with timeouts (usually when the spam arbitron takes too long to identify a message), you may want to use:

INPUT_MAIL_FILTER(`filter1',`S=inet:2526@localhost, F=R, T=S:10s;R:60s;E:5m')dnl

A full description of the sendmail parameters that apply to the filter can be found at www.sendmail.org/~gshapiro/8.10.Training/milterconfig.html.

1.5 Hack the Startup Script

Hack the sendmail startup script (e.g. for Linux, "/etc/rc.d/init.d/sendmail") and add something to this effect:

```
# Start the mail filter
echo -n $"Starting sendmailfilter: "
/usr/sbin/sendmailfilter \
     -h -i -p inet:2526@localhost >/dev/nul 2>&1 &
FILTVAL=$?
[ $FILTVAL -eq 0 ] && success "sendmailfilter startup" \
     || failure "sendmailfilter startup"
echo
     .
     .
     .
echo -n $"Shutting down sendmailfilter: "
killproc sendmailfilter
echo
```

You want to start the filter after sendmail is started and stop it before sendmail is stopped. The port number (2526) is the same one that was defined in the sendmail configuration file (Section 1.4, above). A sample startup file for Linux is supplied in "sendmail.cf", included with this package. A sample startup script for BSD is supplied in "rc.sendmail", also included with this package.

You may want to set the optional flag "-D" on the startup line for the sendmail filter. We run with '-D "/etc/mail/local-host-names"' to set the list of local domain names. Also, if you're going to be using a spam arbitron daemon (e.g. SpamAssassin), set the "-s" parameter to use the port that spamd is listening on (e.g. "-s spamd:2527@localhost"). And, if you're going to be using a virus arbitron daemon (e.g. ClamAV), set the "-v" parameter to use the port that clamd is listening on (e.g. "-v clamd:2528@localhost").

If you are an ISP and you have mail users that don't really exist (i.e. have no home directory) or are delivering mail to other machines, you might want to look at the "-u" and "-q" parameters respectively. Setting the "-u" parameter is especially important if you have users with no home directories and you are going to be using Spam Assassin to scan mail messages for spam.

Restart sendmail so that it reads the new config file and starts the filter. For example, on Linux:

/etc/rc.d/init.d/sendmail restart

Alternately, on Linux, you can copy the "sendmailfilter.rc" file to "/etc/rc.d/init.d/sendmailfilter" and run "chkconfig --add sendmailfilter". This will start/stop the filter separately from sendmail, hopefully in the correct order.

1.6 Install the Message Remailer

MailCorral ships with exact copies of the essential spam notification and message handling programs of SpamCorral, its sister spam handling product. SpamNotify is described in section 1.7 (below), while this section describes MailRelease.

If you wish to allow the recipients of filtered messages to remail them unattended from the corral to themselves, you will need to install MailRelease, the optional message remailer. You will also need to enable this feature by including the "-rm" option on the filter command line or using "AutoRemail" in the config file.

To set up message remailing, you first need to create a message handler robot and configure sendmail to allow forwarding to work, as well as configuring the message remailing utility to run the way you want.

1.6.1 Configure the Message Remailer

Hack the source file MailRelease.pl and change the appropriate variables therein (an easier way is described in the SpamCorral documentation. which shows how to alter this file as well as set up foreign language support). Be sure to make any changes to this file, before you run "make install" so that they will be copied from the build directory to the appropriate install locations. Whichever method you use, you can change these variables:

$CORRAL	The path name of the directory where the original copies of filtered messages are saved until they are released or deleted. This should match the directory used in the CORRAL_DIR manifest constant set in smfopts.h (see section 1.3.7).
$LOG_FILE	The name of the log file where information about all messages that are remailed, etc. is written. You will need to define this variable to and set it to a valid file name to turn on logging (which is off by default).
$REMAIL_HDR_BYPASS	The remail header string that is used to indicate that a message can be remailed. This string must match the tag placed in the message by MailCorral (defined in smfopts.h).
$MAXHEADERLEN	The maximum number of message header bytes to read while looking for remail tags in compressed, corralled messages. If the remail tag isn't found after reading this many bytes, the message cannot be released for remailing. Note that this only applies to compressed messages. There is a tradeoff between the speed of releasing messages and the number of bytes read while looking for the remail tag. If the users are releasing large numbers of messages, you might to reduce this variable from its default setting of 16384 to a smaller value. However, be advised that smaller values (e.g. 4096) might cause some messages with long delivery lists (common with some types of spam) to be unreleasable when compressed. $IGNORE_DOMAINS This variable should be defined to any value, if you wish MailRelease to ignore the user's domain name when processing release requests. Normally, the domain name of the requestor is checked against a list of valid domain names, before they are allowed to release corralled mail. This feature is normally turned on, since it isn't troubling to most users and turning it off is a moderate security hole. $CHECK_USER_NAMES This variable

should be defined to any value if you wish MailRelease to check that a user's name matches the recipient of the message before it can be remailed. This feature defaults to off, since it proved to be more trouble than it was worth to most users. However, it is a big security hole. $DELETE_ON_RELEASE This variable should be defined to any value if you wish MailRelease to delete messages from the corral, once they are released for remailing. This keeps the corral cleaner but it can cause problems if the message is not delivered for some reason. Consequently, this feature defaults to off and released messages are kept in the corral until the regularly scheduled cleanup program finds and deletes them. @LOCAL_DOMAINS The list of domain names that are considered local to this site. Messages will only be remailed to users who send mail from one of the approved domains. This prevents outside users from remailing messages containing viruses, etc.

The list may also include file names, which can contain lists of local domain names, one per line. Typically, this feature would be used to point to the file "/etc/mail/local-host-names" or the same place where sendmail gets local domain name information from.

$ERRORXXXX A series of error messages that can be sent back to the user, if release fails for some reason or another.

Test out any changes you make to this program by running it standalone. In production, it will be running from a ".forward" file so it is likely to fail silently. Note that this test assumes that you've installed the filter and gotten it working and that it has corralled some messages in the corral, as you will be trying to release one of those messages during the test.

To test MailRelease.pl, make up a test email message in a file and pass it through the script. Your test email message must include a from header and a line that matches a filtered message file in the mail corral (the message must have been filtered with the "-rm" option turned on). Note that it is of the utmost importance that there be a blank line between the "From:" line and the "File:" line in the test message, just as there would be in a real mail message. For example:

From: joe@localhost

File: /var/spool/MailCorral/recv_from_mrvirus@virusesrus.com_3F9AE27C

If your test file was called "testrel", as root, try the following command:

./MailRelease.pl <testrel

Note that you should pick a filtered message that you don't really care about, for this test, since it will be deleted once it is remailed. If all is working, "joe" should see the remailed message.

Warning: if the filtered message contains a virus, you should be extremely careful, since you will be remailing the virus through the system without any filtering. It will arrive unscathed and fully functional in Joe's mailbox.

1.6.2 Create a Message Handler Robot

The message handler robot is actually a separate userid that is created for the sole purpose of releasing filtered messages to the general public whenever users ask it to do so.

A copy of each filtered message is added, untouched, to the message corral by the sendmail filter when a received message is altered to protect against viruses, etc. The altered message sent to the recipient indicates how the user can have the unaltered copy of the message remailed to them by sending a message to a particular user (the message handler robot).

The user sends a message to the userid that you create in this section. The userid has a .forward file which directs all incoming mail to the message handler ("MailRelease.pl"). This script scans the mail that it receives, from the other users, and releases the appropriate unfiltered message to them for delivery.

First, you must set up a new userid with a name of your choice. If you altered the REMAILROBOT variable in Altering Fixed Constants and Variables step, you should pick a userid that matches what you chose there. Otherwise, the default name is "mailrobot". As an example, to set up a new userid under Linux:

useradd -c "Message handling robot" -m mailrobot

Note that it is very important that the userid which you create have a valid shell named in its /etc/passwd entry. This shell need not be one that actually works (if you wish to keep people from logging in to the robot's userid) but it must be one that is named in /etc/shells. The default shell, assigned by new user creation, should work but be aware of this fact if you assign any specialty shells to this userid. Sendmail will not forward mail to MailRelease.pl if this isn't the case.

Next, we are going to copy "MailRelease.pl" to either the mail robot's home directory or a subdirectory under it (henceforth, we assume the home directory). But, before we do this, make certain that whatever directory you choose is not writable, except to the owner (no, not even the owner's group can have write permissions). If the directory is writable by anyone but the owner, sendmail will not forward mail to MailRelease.pl

Now, copy "MailRelease.pl" to the mail robot's home directory. Give it the right kind of permissions, etc. to allow sendmail to execute it when it reads the ".forward" file. Generally, sendmail runs as the recipient of a message so this script should have permissions that look like "-rwsrwxr-x" and it should have ownership of root/mailrobot. Note that this program is running as setuid root to allow it to manipulate the files that it needs to mess with. Be careful if you make any modifications to this program. Also, on SuSE there is a permission system which will reset the setuid bit each time SuseConfig is run so you will have to change the permissions in /etc/permissions.easy to 4775 for MailRelease as well, otherwise they will have to be reset manually each time SuseConfig is run.

Pay particular attention to the operation of the setuid bit. Some operating systems (e.g. Linux) ignore the setuid bit on shell scripts, for security reasons. If yours is one of these operating systems, Perl may attempt to simulate the action of setuid. If it does, MailRelease.pl will run correctly when invoked under a non-root userid. If Perl doesn't simulate setuid, you will need to run the setuid version of Perl. Normally, the "configure" script determines this fact and makes the appropriate changes to MailRelease.pl. However, if it doesn't you will have to hack the first line of MailRelease.pl and point it at "suidperl" (probably the same path as is already there but with the name of "suidperl" instead [if you can't find suidperl, it may not be installed, in which case you will need to get it and install it -- under RedHat Linux, suidperl comes as a separate RPM, called perl-suidperl]). You should check that MailRelease.pl can release messages from the corral when run from the userid mailrobot to make sure setuid is working (run the same test as above but do "su mailrobot" first).

On some systems, suidperl will not work either (e.g. on RedHat 7, with Perl 8.5.0, I could not get it to work under any circumstances). In this case, create the following C wrapper ("MailRelease.c"):

```
#define REAL_PATH "/home/mailrobot/MailRelease.pl"
main(ac, av)
   char **av;
{
execv(REAL_PATH, av);
}
```

Compile it with your favorite C compiler (e.g. "cc -o MailRelease MailRelease.c"). Give this program the setuid root permissions instead of MailRelease.pl.

Set up a ".forward" file under this userid that reads:

"|MailRelease.pl"

Note that in some cases, sendmail requires that the full path name of the executable file be used in the ".forward" file and that the name used match exactly the name that is pointed to by the symlink in the "/usr/adm/sm.bin" or "/etc/smrsh" file (see the next section). In that case, the ".forward" file should look like this:

"|/home/mailrobot/MailRelease.pl"

Or, if you had to resort to compiling a C wrapper for the shell script, change the ".forward" file to read:

"|MailRelease"

Make sure that the permissions are "-rw-------" and the owner/group is set to the userid (e.g. "mailrobot") that you created above or forwarding won't work. Sendmail checks the ownership and permissions, for security reasons, before it will do any forwarding.

1.6.3 Additional Sendmail Configuration

Sendmail now has additional security that prevents ".forward" files from executing programs directly. It uses a special shell ("smrsh") that performs extra security checking before allowing a program to be run from a ".forward" file. This shell is very restrictive, almost to the point of being ridiculous, resulting in a good many reasons why it should prevent MailRelease.pl from working (many of which fail silently). You should first read "man smrsh" for more information about "smrsh" and then follow these instructions carefully to get it working.

Note that, if you don't follow these instructions to the letter, there is a very good chance that sendmail will simply drop the message to your mail robot or, just as bad, deliver it to the mail robot's local mailbox (this is a good thing to check ["mail -u mailrobot"], by the way, if the sendmail log says your messages are being delivered but MailRelease.pl is, none the less, failing to release anything). You will get no error message or warning as to why this happens. Sendmail's attitude is that you made a mistake so its not going to do what you wanted and it isn't going to give you any clues why, either. If you end up reading its code to see why its not forwarding messages to your mail robot, you'll see that it even has a flag that it sets (DONTBLAMESENDMAIL) to indicate its not its fault. Too funny! Hopefully, if you follow these instructions scrupulously, it won't come to that.

When you built sendmail, smrsh should have been built along with it. If not, build it now. If the sendmail install did not do so already, copy smrsh to a convenient location (e.g. "/usr/sbin/smrsh"), probably the same directory as sendmail resides in. Apply execute permissions so that the world can execute smrsh, if they aren't

already there.

For forwarding of any kind to work, the shell chosen for the userid that is forwarding must be a "valid" shell. This means that the shell named in the userid's entry in /etc/passwd must be listed in the file /etc/shells. Verify this fact. The shell chosen need not be one that actually works (if you wish to keep people from logging in to the robot's userid) but it must be one that is named in /etc/shells. The default shell, assigned by new user creation, should work but be aware of this fact if you assign any specialty shells to this userid.

To get smrsh to execute programs, you need to create a directory named "/usr/adm/sm.bin". You must give it ownership and permissions of root/root/755 and put into it soft links to any programs which will be executed by a ".forward" file. Be very careful what programs you put in this directory because programs therein can be executed by mail messages if a hapless user decides to add them to their ".forward" files.

In our case, we need to put in a soft link to the message handling robot. For example:

ln -s /home/mailrobot/MailRelease.pl /usr/adm/sm.bin/MailRelease.pl

Or, if the C wrapper was used to get MailRelease to run setuid root, try something like this:

ln -s /home/mailrobot/MailRelease /usr/adm/sm.bin/MailRelease

Alternatively, some versions of sendmail use "/etc/smrsh" to implement this feature. You can look at the source code for sendmail (specifically, see the code in smrsh.c) or try both of them until you get it to work.

The sendmail configuration file must be altered to enable message forwarding. This is done by adding some lines to the ".mc" file for "sendmail.cf" (this is way better than hacking "sendmail.cf" directly, which is definitely not for the faint of heart) and running it through M4. Start by copying the canned configuration supplied with the OS, if you haven't already done so and add the following:

FEATURE(`smrsh', `/usr/sbin/smrsh')dnl

Note that, in the above line, you should change "/usr/sbin/smrsh" to point to wherever you actually installed smrsh.

Also note that the default "cost" for mail passed to the program mailer that invokes smrsh is "expensive". This has the unwanted effect of placing mail forwarded to programs into the mail queue where it can sit for hours before being delivered. I cannot even speculate why this choice was made for the default but you may not wish it to be so.

If you would like your mailrobot to respond quickly to requests, you will need to change the expense of the local mailer. First, you should check your sendmail.cf file to see if the expensive flag is, indeed, one of the flags being passed to the program mailer. If you see the letter 'e' in the flags, defined by the parameter "F=", for the "prog" mailer, you'll need to remove it. For example, if your config file reads something like:

```
Mprog,     P=/usr/sbin/smrsh, F=lsDFMoqu9e,
    S=EnvFromL/HdrFromL, R=EnvToL/HdrToL, D=$z:/,
    T=X-Unix/X-Unix/X-Unix, A=smrsh -c $u
```

You will need to change the expense of this mailer. To do this, add the following line to the ".mc" file:

define(`LOCAL_SHELL_FLAGS', `u9')dnl

Be sure that this line occurs before the line that says "MAILER(`local')", since that macro uses the values set by it. And, if there is no definition for a local mailer, be sure to add it.

Compile the configuration and install it, per the directions in the "Configure Sendmail" section, above.

If you prefer to hack the "sendmail.cf" file directly, you should add something that looks like this to the end of the file:

```
Mprog,     P=/usr/sbin/smrsh, F=lsDFMoqu9,
    S=EnvFromL/HdrFromL, R=EnvToL/HdrToL, D=$z:/,
    T=X-Unix/X-Unix/X-Unix, A=smrsh -c $u
```

Its time to try smrsh to see if will run MailRelease.pl. To do this, logon as JoeUser (i.e. not root) and run MailRelease through smrsh, using the following command (or whatever is appropriate for where you installed smrsh):

```
/usr/sbin/smrsh -c /home/mailrobot/MailRelease.pl
```

Due to the fact that smrsh looks up all of the programs it executes in /usr/adm/sm.bin, it might seem to you that the actual path used to execute MailRelease.pl is unimportant. One might think that only the program name matters, and you might even be correct for some of the earlier versions of smrsh. However, it would now appear that smrsh is checking to see that the path that you use to execute the program matches exactly the path pointed to by the symlink in /usr/adm/sm.bin. So, you should use the correct path wherever you reference MailRelease.pl (e.g. in the command above and in the .forward file).

If you can't get smrsh to execute MailRelease.pl, check all of the permissions, etc. and keep trying until it works. If it does execute, type a couple of characters (it reads from stdin) and give it an eof. It should do nothing but terminate. Or, if you like, you can feed it the test message you made up earlier and see if it really works, although here we're basically just testing to see that smrsh will load the program.

For the next step, you'll probably want to turn on logging by aiming $LOG_FILE at a file (e.g. "/var/log/mailrobot"). Once this is done, send a message to mailrobot that has the name of a corralled piece of mail as its subject line. For example:

```
mail -s /var/spool/MailCorral/recv_from_ew@jg.sp_3E2278A4 mailrobot
```

You should see a log entry for the request and, if the message is a real one, it should get released. If no log entry is created, you may need to run MailRelease with suidperl (see the "Create a Message Handler Robot" section, above). You should also check to see that the message is not in the queue, waiting to be delivered (see above for sendmail configuration changes needed to circumvent this), that the permissions on MailRelease are "-rwsrwxr-x" and that it has ownership of root/mailrobot.

If MailRelease doesn't work, its usually not its fault. There are a lot of things that must be right before sendmail will condescend to execute the program. Please check them all before blaming MailRelease.

1.6.4 Alternative to Message Handler Robot

Kai Schaetzl has suggested an alternative to setting up a separate userid for the message handler robot. Although my personal preference is the separate userid approach, it is somewhat difficult to set up and get working. Kai's alternative is easier and works fine on SuSE (we haven't tried it elsewhere). Basically, here's what he had to do:

To the aliases file (/etc/aliases or /etc/mail/aliases), add this alias (or something like it):

mailrobot: |/usr/sbin/MailRelease.pl

Set the setuid bit for MailRelease. If you'd like to restrict access to this program, use the following permissions:

```
-rwsr-x---  1  root daemon  14831 Mar 11 18:08 MailRelease.pl
```

Make sure that /usr/sbin/suidperl has the setuid bit on, as well. With this arrangement, mail is piped through MailRelease by sendmail as "daemon:daemon". That's the reason for setting MailRelease to group daemon. If the permissions are not set correctly, you'll get an unknown sendmail error 162. Sendmail itself doesn't seem to be running as "daemon:daemon", maybe as "mail:mail", so the shell for piping may be different from the user sendmail runs as.

Note that the same warning about the SuSE permission system applies here. This system will reset the setuid bit each time SuseConfig is run so you will have to change the permissions in /etc/permissions.easy to 4755 for MailRelease, otherwise they will have to be reset manually each time SuseConfig is run.

1.7 Install the Optional Spam Notifier

MailCorral ships with exact copies of the essential spam notification and message handling programs of SpamCorral, its sister spam handling product. MailRelease is described in section 1.6 (above). This section describes the optional spam notifier SpamNotify.

If you are corralling spam and you wish to receive regular notifications of the corralled spam, you should install SpamNotify. If you don't corral spam or don't care about notifications, you can skip this section. For more complete documentation on this program, see the SpamCorral documentation.

1.7.1 Configure the Spam Notifier

To configure the spam notifier, you can hack the source file SpamNotify.pl and change the appropriate variables therein (an easier way is described in the SpamCorral documentation. which shows how to alter this file as well as set up foreign language support). Be sure to make any changes to this file, before you run "make install" so that they will be copied from the build directory to the appropriate install locations. Whichever method you use, you can change these variables:

$CORRAL	The path name of the directory where suspended spam is saved until it is released or deleted. If you are using MailCorral, this should match the directory used in the SPAM_NAME_TEMPLATE manifest constant set in smfopts.h.
$STAMPFILE	The name of a file to use for holding the timestamp of the last spam notification message. You may pick any name you like, provided it doesn't collide with the names used by the mail fitler (e.g. none of the names in the message file name templates in MailCorral's smfopts.h).
$SPAMROBOT	The userid of the spam handler robot (that will be created in Section 1.6.2, above). However, you probably won't have to change this variable if you are using a version of sendmail that forwards mail to programs via smrsh. The

reason is because SpamNotify automatically detects which userid is being used for the spam handling robot, at runtime. The value set by this variable is only used as a fallback, default. Note that, if a value is set in the config file, it is taken as the absolute name and the results of automatic detection are ignored.

$MAXHEADERLEN

The maximum number of message header bytes to be read while processing messages for notification. If the end of the headers isn't found after reading this many bytes, the message is skipped and no notification is sent.

There is a tradeoff between the speed of processing messages and the number of bytes read while looking for headers. If there is a large amount of corralled spam, you might want to reduce this variable from its default setting of 16384 to a smaller value. However, be advised that smaller values (e.g. 4096) might cause some messages with long delivery lists (common with some types of spam) to be skipped.

$DATEFMT

The date/time format string that is to be passed to sprintf whenever date stamps are formatted for display. This value consists of the format pattern string, followed by the variables to be substituted into it. Note that the double quotes around the format string and the dollar signs in the variable names must be escaped, since this value is evaluated multiple times. If you don't escape these items, your date stamps will come out broken or all zeros.

Within this parameter, you can use the variables: "$Year" (the full, four-digit year number); "$Mon" (the two-digit month number); "$MDay" (the two-digit day of the month); "$Hour" (the twenty-four hour clock); "$Min" (the minutes in the hour); "$Sec" (the seconds in the minute). The default value is:

\"%04d-%02d-%02d %02d:%02d:%02d\", \$Year, \$Mon, \$MDay, \$Hour, \$Min, \$Sec

which gives a date stamp that looks like "2003-04-18 03:27:05". If you live in Europe, you might want to try:

\"%02d.%02d.%04d %02d:%02d\", \$MDay, \$Mon, \$Year, \$Hour, \$Min

which gives a date stamp that looks like "18.04.2003 03:27".

$NOTIFYSUBJ

The subject line to be used on notification messages sent to the users about spam held for them.

$NOTIFYMSG

The text of the message that is sent to the users, describing the spam that is waiting for them and what to do about it. This is a multi-line message that begins the body of the notification message.

$XXXXTAG

The various tags that are used in the body of the notification message to identify the components of each piece of spam. Things like the from, subject and date headers.

1.7.2 Set up Periodic Notifications

Spam that is corralled by the filter is placed in the corral, pending release for remailing by the recipient or, ultimately, deletion by the cleanup prcoess (after a set period of time has elapsed). In order for the recipients of the spam to know what they have been sent, so that they can decide whether they want it remailed to them or not, the spam notifier should be run at regular intervals.

The spam notifier is run by an entry in your cron table, probably once or twice a day (depending on how often your users want to see what spam they've received). I like to do these kinds of things at the end of the day, when the load on my systems is low. Here's the crontab entry that I use:

```
# Once a day at 01:00, notify the users of all of the spam received in
# the last 24 hours.
00 1 * * * root /home/mailrobot/SpamNotify.pl
```

Alternately, if you are expecting huge amounts of spam, you may wish to partition the corral (this puts less load on the file system and avoids "out of resources" type crashes when processing the corralled spam). Here is a crontab entry that requests 24 hour partitioning (the default):

```
# Once a day at 01:00, notify the users of all of the spam received in
# the last 24 hours. Partition the corral at the same time.
00 1 * * * root /home/mailrobot/SpamNotify.pl --Partition
```

Note that the filter also has a partition setting which applies to all corralled messages (viruses as well as spam). If you use this setting (and, if you want partitioning, you should), you needn't worry about SpamNotify doing any partitioning, although it doesn't hurt to do it both places. SpamNotify will partition any corralled messages that didn't get partitioned by the filter (e.g. accidentally), otherwise it will do nothing. See the "Partition" or "-pn" options elsewhere in this document.

1.8 Set up Periodic Cleanup Jobs

The sendmail filter generates an unaltered copy of every email message that is not delivered asis. You should add something to your cron table to clean these files up after a reasonable length of time.

The script "filterclean" cleans up all of the older filtered messages, retaining those that are less than thirty days old (note that the directory name must match the template name in smfopts.h):

```
# Once a day at 00:30, remove all of the filtered mail messages that are
# over 30 days old.
30 0 * * * root /etc/mail/filterclean
```

Optionally, you may hack this script to enable compression of all messages older than 7 days old. To do this, uncomment (by removing the '#' from the beginning of the line) the last line in the script that reads:

```
#/usr/bin/find /var/spool/MailCorral/ -not -name \*.gz -depth -type f -mtime +7 -exec gzip \{\} \;
```

If you create debugging files (enabled by default), you should clean them up weekly using logrotate. Here's the lines you need to add to the logrotate config file (/etc/logrotate.conf):

```
/var/log/sendmailfilter {
    missingok
    notifempty
}
```

1.9 Configuring Local Options

The operation of the sendmail filter can be altered by configuration parameters in two configuration files. One is a global configuration file that applies to all of the messages processed by the filter. The other is a local configuration file that only applies to the messages for an individual recipient. Typically, there is one such file for each recipient, in their home directory.

Unless the GLOBALOPTIONS and/or LOCALOPTIONS variables in smfopts.h are changed or the "-C" flag is used on the command line, the global configuration file will be "/etc/mail/sendmailfilter.cf" or "/etc/sendmailfilter.cf" and the local configuration file will be ".sendmailfilter", in the recipient's home directory.

Either of these files can contain filtering options, spam processing options, message inserts text, white lists, black lists and spam delivery options. In addition, the globals options file can contain other startup options, including those that can be specified on the command line.

You will probably want to update the global options files with spam processing information specific to your installation. For example, it is suggested that, if you constantly receive spam from a particular domain, you should add that domain to the global black list. See the Configuration section of the documentation for more information on which options can be set in these files.

The configuration files can be edited at any time. The sendmail filter must be restarted for new global options to take effect. Changes to the local configuration file take effect with the receipt of the next message destined for the user in question.

1.10 Installing the RPM on RedHat Linux

MailCorral is available for installation on RedHat Linux in RPM form, which reduces the complexity of the installation considerably. The RPM is built on RedHat 7.0 and 8.0 and requires sendmail 8.12.0 and above. There should be no reason why it won't work on later versions of RedHat Linux, as there aren't any real OS dependencies in the filter. Similarly, it should work on later versions of sendmail (such as 8.12.9).

To install the filter, download the appropriate RPM for your system and place it in a suitable directory on your machine. As root or some other equally omnipotent user, type the following command:

rpm -U /xxx/MailCorral-vvv.i386.rpm

where "xxx" is the directory where you saved the RPM and "vvv" is the version information for the RPM you downloaded (e.g. "1.1.2-1.8.0"). If you prefer, you can install the RPM using a visual package manger such as GnoRPM.

The RPM will set the following filter parameters to the values noted. Most of them can be overridden in the configuration file (see the Configuration section of the documentation for more information on which options can be set). The settings are:

TECHSUPPORT	The name of the person whom users should contact if they need assistance with filtered messages. Set to "postmaster".
REMAILROBOT	The username of the message remailer robot. Set to "mailrobot". A user account for "mailrobot" is also created by the RPM.
POSTMASTER	The name of the person to whom warning messages should be emailed if any viruses or unknown filter types are found. Left undefined so that no warning messages are sent.
DOMAIN	The name or names of the local domain. Set to a bogus domain name of "yourdomain.com". You will need to override this with the "-D" command line option or the "DomainList" config file parameter.
CORRAL_DIR	The directory where all of the corralled messages and spam will be placed. Set to "/var/spool/MailCorral".
DEBUG_FILE_NAME	The name of the file where debugging information is written. This name is left undefined. If you want to create a debugging file, the "-d" command line option or "DebugFile" config file parameter are available.
GLOBALOPTIONS	The names of the global sendmail filter configuration files. Set to "/etc/mail/sendmailfilter.cf;/etc/sendmailfilter.cf".
LOCALOPTIONS	The name of the local sendmail filter configuration file. Set to "~/.sendmailfilter".

The RPM installs the following files into the directories noted:

/etc/mail/filterclean	A shell script that is run by the daily cron job to clean up any corralled filtered messages and spam that are older than 30 days. You may alter this script to reduce or increase the number of days that corralled messages are kept.
/etc/rc.d/init.d/sendmailfilter	A startup script that is run, at boot time or when the runlevel is changed, to start/stop the filter. You may want edit this script to change the options used to start the filter.
/usr/sbin/MailRelease.pl	A Perl script that is used by the mailrobot to automatically release mail that is corralled because it contains viruses, etc.
/usr/sbin/SpamNotify.pl	A Perl script that is run at regular intervals (daily) to send out notification messages of all spam that is corralled since the last time it was run.
/usr/sbin/sendmailfilter	The sendmail filter itself.
/usr/share/doc/MailCorral-xxx	The documentation directory. The string "xxx" is replaced with the version of MailCorral contained in the RPM.

In addition to installing files, the RPM carries out most of the work necessary to configure the filter. It configures the following items:

The RPM copies "/usr/lib/sendmail-cf/cf/redhat.mc", the RedHat supplied sendmail macro configuration file, to "/usr/lib/sendmail-cf/cf/MailCorral.mc" and hacks it so that the filter will be invoked by sendmail on port 2526. It then compiles the macro file to yield "/usr/lib/sendmail-cf/cf/MailCorral.cf".

The RPM copies adds the filter startup script "/etc/rc.d/init.d/sendmailfilter" into the appropriate start/stop directories at the right runlevels so that the filter will be started/stopped at boot time and when runlevels are

changed.

A userid of "mailrobot" is created and set up, if it doesn't exist. This userid will be configured to receive mail messages sent to it requesting that corralled messages be released and to carry out these requests.

The SpamNotify script will be copied to the home directory of "mailrobot" and set up to run at daily intervals through a link placed in /etc/cron.daily.

The corral cleanup script will be set up to run daily (whenever files in "/etc/cron.daily" are run). This will cause corralled messages over 30 days old to be deleted.

All that user need do, after the RPM is installed, is complete the hacking of "MailCorral.mc". Compare it to your sendmail config file and make whatever changes are necessary to configure sendmail as you currently have it set up. If you prefer, add the three lines needed to start the filter to your existing configuration file. The "Configure Sendmail" section (above) talks about how to make the changes and then compile and install the configuration file in the sendmail directory.

2. Sendmail Filter

2.1 Description

The MailCorral sendmail filter is a robust virus/spam filter program that runs as part of sendmail (using the milter interface) to filter out viruses and spam from all mail delivered on the site running sendmail. The filter handles all currently known-malicious attachments plus attached and inline HTML. It also handles archiving of received messages, if desired, and can be used to transparently send and receive encrypted email.

This program can be installed as part of sendmail and left to run unattended. It will render harmless any viruses found in delivered email and notify the user inline, in the message itself, of its actions. A backup copy of the unfiltered message is kept for a fixed length of time, just in case filtering rendered it truly unusable.

Spam identification is carried out on two levels. The sendmail filter has a simple spam detection scheme, based on white and black lists, that is always enabled (unless the "-ss" option is set or "SpamFastPath" is set to "No"). You may run the filter with only this mode of spam detection enabled, which, despite its simplicity, is amazingly effective if you work at configuring it properly. Otherwise, more elaborate Spam identification techniques can be applied via a built-in connection to one of the popular spam recognition packages. The spam recognition package is only called if the fast path through the white/black list processing does not recognize a message as being spam, hence spam recognition can be very economical.

Once spam is identified, there are three delivery options which may be selected. Spam can simply not be delivered. Or, it can be delivered with a marker in the subject to identify it as such. Finally, in lieu of being delivered, it can be redirected to a corral where it can be released at a later time. In this delivery mode, as in the other two, spam handling runs completely unattended by your system administrators. If you use the optional spam handling package SpamCorral, users can be automatically sent periodic summary messages which will allow them to release for delivery only those pieces of spam that they actually want to see. The rest of the spam is deleted after a short holding period.

Virus identification is also carried out on two levels. The sendmail filter has a simple virus detection scheme, based on lists of acceptable attachment types and MIME types, that is always enabled (unless the "-vv" option is set or "VirusChecker" is set to "No"). You may run the filter with only this mode of virus detection enabled, which, also despite its simplicity, works quite well. Otherwise, more elaborate Virus identification techniques can be applied via a built-in connection to one of the popular virus recognition packages.

2.2 Features

Here is a list of what we consider to be the most useful features of the product. What is left unsaid is that it is basically install and forget. Once you set it up and it is running to your satisfaction you should not normally ever have to touch the filter again.

- Stops viruses and spam before they ever get to the recipients. This can reduce network traffic (for spam) and guarantee that a virus will not get launched by mistake.
- Filters all plain text and MIME components of the message.
- Removes harmful inline HTML, including that embedded in plain text and renders harmful attachments innocuous (by renaming).
- Decodes all encoded plain text and HTML MIME components to find viruses and spam hidden by encoding.
- Can filter all messages or only those from outside.

- Optionally, can send virus detection reports directly to the postmaster.
- Has built-in, fast black and white list processing as well as a statistical identifier for spam.
- Can invoke the popular spam arbitration daemon to determine if a message contains spam, using the built-in interface.
- Only the message body (i.e. text portions of the message, not binary attachments) are passed to the spam arbitron. For large attachments, this results in a significant speedup.
- The built-in black/white list processor and statistical identifier acts as a fast path front end for the spam arbitration daemon, giving at least a threefold performance improvement for domains in those lists and messages that meet the statistical tests.
- Has built-in, fast virus detection lists based on file and MIME types known to be potentially harmful if opened indiscreetly.
- Can invoke the popular virus arbitration daemon to determine if a message contains a virus, using the built-in interface.
- Allows the user to use their own spam and virus arbitration methods via a programmable interface.
- Evaluation of spam and virus filtration results is through user supplied, logical expressions that allow them to set the order of processing and check the arbitron results programmatically.
- The expression evaluator optimizes the expressions so that fast paths and early outs do not require calling of all arbitrons.
- Filtration messages are given in plain language, inline, in the message received.
- The text of filtration messages is configurable in the global configuration file as well as on a per-user basis. Foreign languages are easily accomodated.
- Message filtration, spam filtration and message disposition options are settable on a per-user basis. Permits tailoring of the filter's actions for each individual message recipient.
- Retains an unaltered copy of all filtered (i.e. altered) mail for a set period of time (chosen by you), just in case filtering renders it unusable. No need to worry about the consequences of filtering.
- Unaltered messages can be easily released, by the recipient, for delivery, untouched, if the optional remailer is installed. How-to instructions are provided in the filtered message itself.
- Three delivery modes are possible for spam: deliver it but tag it as such in the subject; put it straight into the trash; sideline it in the corral for possible later delivery.
- Spam that is not delivered but corralled instead may be remailed using SpamCorral or simply deleted on a periodic basis.
- Spam may be automatically replied to with a rejection notice or silently discarded.
- Rejected spam uses a pacing mechanism that prevents the ping ponging of rejections, rejection rejections, etc. Once a spammer's periodic quotient of rejection notices are sent, further spam from them is silently discarded. You pick the pacing threshold and period.
- Using SpamCorral, users can be sent periodic (usually daily) summary notification of all spam received and corralled in that period.
- As a result of notification, users can release any spam they want to see, from the corral, for remailing to them. Important spam (is there such a thing) or mistakenly labeled spam is not lost.
- Directly delivered and remailed spam is tagged (i.e. "[SPAM]") in the subject so that it can be easily separated by recipients (e.g. a subject line filter can send it to a special mailbox).
- Received messages can be copied to an archive that is maintained by the filter.
- Alternately, received messages can be emailed through regular channels to third party archive programs.
- The list of recipients whose mail is to be archived is fully configurable on an individual user basis, domain name basis, etc., using lists and pattern matching.
- Additional support is provided for third party archive programs by filtering out unwanted bounce messages from them.
- Transparent encryption and decryption of outbound and inbound messages can be done using the popular public domain privacy guard package.
- Robust code - runs reliably, without human intervention.

- High performance - written entirely in C and completely reentrant.

2.3 How it Works

This program is invoked by the milter interface of sendmail, upon startup. It registers callback routines for the various message processing actions, as defined by the milter interface. Upon return to sendmail, the callbacks are called by sendmail whenever it has messages to deliver. The callbacks filter the messages for noxious entities (e.g. viruses) and warn the user of their presence by inserting warnings into the body text of the message. In some cases (e.g. attachments), the offending entities are altered or deleted to render them harmless.

Basic spam checking is undertaken by MailCorral using white and black lists to compare against the sender and recipient of a message. This basic technique can be effective against spam that is always sent from the same domain. If attention is paid to how the lists are set up, one can arrive at a simple yet functioning spam filter. Regardless of what other spam checking is done, this basic checking is always done (unless the "-ss" option is set or "SpamFastPath" is set to "No") whether to provide the only means of spam identification or to provide a fast path, in front of a spam arbitration daemon, that improves performance.

A second built-in spam check is done by performing a statistical anyalysis of the text of the message, looking for certain features that spammers are known to employ (mainly to see if they can circumvent spam filters). This check is done essentially at no cost, while the message is being otherwise processed and can provide a second fast path that avoids the need to make an expensive call to a spam arbitration daemon.

Further spam checking can be enabled by informing MailCorral about a spam arbitration daemon such as SpamAssassin. If this is done, a request for spam determination is sent to the arbitron, via a pipe, which includes the headers and all text portions of the message (binary file attachments are omitted).

If either the built-in white/black list processing or the arbitron determine that the message is spam, it is disposed of according to the disposition options chosen. Three choices are available: deliver the spam, suitably tagged as such; uncerimoniously place the spam in the trash can; divert the spam to a corral where it can be remailed to the recipient at a later date.

The first two disposition options are self-explanatory. The third option causes the spam not to be sent directly to the recipient but instead written to a corral directory. The corralled message is given all of the usual filtering, before it is corralled, hence it is ready for immediate remailing, should the recipient so decide (see SpamCorral).

Basic virus checking is carried out by MailCorral using a set of lists that identify file attachment types and MIME types that are known to be potentially harmful. Depending on the type found: nothing is done; a warning is given; the attachment is renamed to prevent it being opened; or, for known really nasty items, the attachment is deleted. This basic technique can work quite well if the users are careful and pay attention to the cautionary messages inserted. Regardless of what other virus checking is done, this basic checking is always done unless the "-vv" option is set or "VirusChecker" is set to "No".

Further virus checking can be enabled by informing MailCorral about a virus arbitration daemon such as ClamAV. If this is done, a request for virus determination is sent to the arbitron, via a pipe, which includes all attachments and inline components where a virus could lurk. If it determines that the attachment or component is infected, it is deleted by MailCorral and the recipient is notified of the fact. This ensures that the user cannot inadvertently open a malicious attachment, regardless of how disinclined they are to heed warnings.

Basic archiving of received messages is provided by the filter. It can create and manage an archive directory on disk that contains some or all of the messages received, depending on its configuration options. A single file is created for each message and a directory structure is built that allows thousands or even millions of messages to be stored and easily managed. An exact copy of each message, before filtering, is archived.

Support for third party archive programs is also available. If requested, the filter can email an exact copy of each received message to a third party archiver through normal email channels. This allows the archiver to reside on the local system or elsewhere. Further support for such archivers is provided through the filter's capability to delete unwanted bounce messages from third party archivers, should they becoume unavailable due to problems such as network failures.

A connection to the open source privacy guard library GnuPG allows the filter to transparently encrypt outgoing messages and decrypt incoming messages using the user's key ring. This allows encryption/decryption to be used even with mail readers that don't directly support it.

To do all of the above, this program must be used in conjunction with a version of sendmail that has milter support and must be set up in the sendmail config file to be called by milter. In addition, spam recognition requires the use of a spam arbitration daemon (unless you wish to insert your own code into the filter) to decide which messages are, indeed, spam.

2.4 Filtered Items

MailCorral tries to disable all harmful items found in the email messages it processes, whether they be attachments or objects embedded directly in the messages. The disabled items, such as viruses and other malicious bits of executable code, can often damage or destroy any system that receives the messages. Disabling these items prevents them from having their otherwise disastrous effect.

For a complete list of all items removed, see the filter tables in the program code (smfopts.h). The synopsis is:

- All inline MIME types that aren't attachments are checked to see if they are in a table of acceptable types ("AcceptableTypes" in smfopts.h). They are rejected, if not. In general, certain text types, postscript and PDF documents, certain images and certain video streams are acceptable inline.
- There are other, popular inline MIME types that we would like to accept but at least one well-known mail reader erroneously converts them into something it shouldn't, thereby allowing an application which shouldn't be allowed to do so to handle them. Allowing them through would let a malicious person send an apparently harmless object, with one of these MIME types, that was actually a virus. In general, these unfortunately handled MIME types are certain encryption signatures (ironic, isn't it), certain kinds of audio streams and one kind of video stream. If your mail readers don't provide this "feature", rejection of these MIME types can be turned off (see "NOT_BRAIN_DEAD" in smfopts.h).
- Attachments that are well known to contain viruses are removed. These are attachments such as "explorer.doc", "resume.doc" and any screen saver files (extension ".scr") which have been used time and again to carry viral material.
- Any attachments that are identified by a virus arbitron as containing viral signatures are removed. The virus arbitron is chosen by the "-v" command line parameter or the "VirusProto" configuration file option.
- Attachments that are directly executable are very likely to contain viruses, consequently they are neutralized. Think about it. There are lots of them, such as ".exe", ".com" and ".dll" files plus interpreted languages such as Perl, Java and Visual Basic (the entire list is in smfopts.h in the table "FilteredFiles") in addition to all of the various scripting and batch languages.

- Indirectly executable attachments are more insidious. These are files that you wouldn't normally think of as being executable but which can contain code that is run when the file is opened. Such things as compiled help text, install, setup and configuration files, OLE and control panel extensions all allow code of one form or another to be run when they are opened. These hidden executable types are altered to make them relatively harmless.
- Attachments, that may be executable because they can contain macros and which could therefore contain viruses are given a warning. These tend to be text formatting languages, spreadsheets, database description languages and certain kinds of installation files.
- Unknown attachments and unknown embedded MIME types are given a warning and rejected.
- Just as with inline MIME types, attachments whose otherwise innocuous extensions may be converted into executable types by some lame-brain mail programs are given a warning. Not doing so would allow an application which shouldn't be, to be allowed handle them, thereby letting a malicious person send an apparently harmless attachment that was actually a virus. In general, these unfortunately handled attachment types are certain audio and video objects and GNU tar files (see the list in smfopts.h). If your mail readers don't provide this "feature", rejection of these attachment types can be turned off (see "NOT_BRAIN_DEAD" in smfopts.h).
- Attachments and embedded MIME types that are known to be innocuous (e.g. ".jpg" or "image/jpeg") are left as is.
- HTML, whether sent as a MIME encoded entity or embedded directly into a plain text message, offers the possibility to launch a virus payload as soon as it is received by most email readers. They typically treat HTML as an integral part of the message and immediately interpret/display the HTML when rendering the message. As a result of this behavior, all HTML is laundered to remove harmful tags.
- The HTML tags <iframe>, <object>, <script>, <applet> and <embed> are always removed. These tags are frequently used to deliver virus payloads.
- According to the table ("RejectedHTMLs" in smfopts.h), the tags enumerated therein are also removed if they are found anywhere (even embedded in non-HTML text). Typically, these are potential virus launcher tags such as <frame>, <img>, <layer>, <ilayer> and <link>.
- Optionally, if HTML filtering is on, the tags given in the table ("AcceptableHTMLs" in smfopts.h) are allowed while all others (except conditional tags, see below) are rejected. In addition to this, before an acceptable tag is passed, the tag parameters found in the table ("GlobParms") are checked and threatening ones are removed as noted. Basically, the tags that are accepted are font, formatting, layout, block and heading tags.
- In addition to checking for tag names, certain global parameters are checked on all tags. Presently, the only global parameter that is checked is the "style" parameter. The subparameters allowed for style are defined in the table ("StyleParms" in smfopts.h). To summarize, most of the style subparameters are allowed except for the ones that are likely to embed or link to foreign objects. This permits most innocuous messages to be formatted correctly using style sheets.
- Several conditional HTML tags are allowed (based on the table "ConditionalHTMLs" in smfopts.h). For these tags, acceptability depends on the parameters and subparameters in the tag. The individual parameter tables ("BodyParms" and "MetaParms") govern which parameters are kept. As with the global parameters, any subparameters that cannot embed or link to foreign objects are allowed.
- All encoded text and HTML objects are unencoded for purposes of virus and spam checking, since spam and viral payloads are frequently encoded to obscure their purpose. Alterations to the message body are reencoded to preserve the color and flavor of the message.
- The message subject is checked as well as the body. Several viruses are known to span their launchers across the message subject and body to circumvent virus scanners, a technique in which they are ably assisted by some mail readers that treat subject content as part of the message body in certain circumstances.
- Messages that are determined to be spam are optionally delivered, discarded or sent to the corral. Spam determination is governed by the spam arbitron chosen by the "-s" and "-ss" command line parameters or the "SpamProto" and "SpamFastPath" configuration file options.

Note that the messages that apply to each of these tests are found in the same file as the tests themselves (smfopts.h). The English text of each message is pretty self-explanatory. Furthermore, each of the messages may have its text overridden in the global and local configuration files, as well as by the foreign language options. The configuration file documentation has a description of each message and its default text.

Please bear in mind that, while every attempt was made to create code that would nullify all of the known malicious items at the time that MailCorral was written, virus writers and their ilk are very creative. As the, dare we say art, of virus writing progresses, new and improved viruses may determine ways to get past MailCorral. However, just as sex without a condom is more dangerous than with, email without a filter is more dangerous than with. MailCorral is sure better than nothing (some testimonial, huh).

In an effort to ensure that as much nasty material as possible is caught, MailCorral is designed to pass the validation suite that BSM Development offers elsewhere. If you think you've found a virus that isn't caught by MailCorral, please send it to us and we'll update the validation suite as well as the filter. This will not only help MailCorral to filter out all of the latest virus "technology" but also assist anyone else needing to verify that their mail filter is up to date, since the validation suite is made available to everyone.

If you are sending us a sample, you could try emailing the virus directly to BSM Development but this is unlikely to work if our virus filters are alert. Instead, an approach that obfuscates the true nature of the message should be employed. My preferred method is to edit the mail inbox with a text editor and copy the entire message containing the virus, from the first header to the last line of the MIME attachments. Save this entire message, headers and all, in a text file and then zip the text file. Attach the zipped file to the mail message that you send to us.

Many people may find the filter criteria employed by MailCorral to be fairly harsh (e.g. we often receive complaints asking why HTML messages are tagged and altered). Although we spend much time evaluating viral payloads and are constantly trying to eliminate noise filtration, we realize that some of the items the filter catches may be viewed by the man in the street as acceptable. In MailCorral's defence, most of the parameters chosen for the filter are there because actual viruses have been observed using the filtered items as delivery mechanisms. You are more than welcome to tune the filter parameters but don't do so lightly. Turning off any of them will certainly increase your risk of exposure and it may only be a matter of time before a virus slips through the sieve. Weigh the annoyance factor carefully against the cost of restoring a system damaged by a virus. We believe that being overly cautious is much better than being careless, in this case.

Finally, if you use a signature based virus arbitrion (such as ClamAV), you may be able to relax the built-in criteria that MailCorral uses and rely on the fact that the arbitron will reliably find all of the viruses but with less false positives. This should prove less intrusive to your users. On the other hand, there's something to be said about the belt and suspenders too approach, since we've found that the users will be even more upset when a virus gets through, abuses their email address book, steals their credit card numbers, signs them up for membership on several spam blacklists and then deletes all of their data.

2.5 Message Remailing

When a message is altered to remove a harmful item, an unaltered copy of it is kept in the mail corral for a short length of time. If a message was altered in such a way as to render it unusable, it can be retrieved in its pristine state. Optionally, a message handling robot can be set up that will allow the recipients of altered messages to retrieve them from the corral and remail them directly to themselves in unaltered form.

Normally, this option is not enabled, since it is dangerous to allow possible viruses to be delivered to the recipient without filtering. The usual method of operation is to have the recipient request a Tech Support

person to deliver the message. This allows Tech Support to peruse the message and ensure that it isn't harmful before sending it on its way.

In certain high volume mail delivery situations, however, it may not be desirable to require such human intervention, given the quantity of mail that may be involved. In these situations, automated delivery of the recipient's unaltered mail may be preferable, providing all concerned are made aware of the danger of releasing live viruses from the corral.

The message remailer robot is invoked by the recipient of the message through a mail message sent to the robot. The altered message contains instructions on how to mail the robot as well as a link which many mailers will follow to instantly create the required mail message. Once the mail is sent to the robot, it responds by sending the unaltered message.

2.6 Spam Handling

Enhanced spam handling (other than the basic black/white list processing built in) is carried out in cooperation with a spam arbitration daemon (arbitron), such as SpamAssassin) or one or more of the user's programmable arbitrons. MailCorral prepares a representative message that contains all of the components of each received message that are important from the standpoint of spam arbitration. It then sends a request for spam determination, via a pipe to the arbitron, which contains the representative message and the arbitron renders a decision as to whether the message is spam or not. The arbitron's decision is final. Or, in the case of the user's programmable arbitrons, the message is passed in a file but the rest of the details are the same.

The received message is processed to create the representative message by first removing any attachments and inline components that are not directly viewable by typical mail readers (e.g. not text/plain, text/html, etc.). This is done for performance reasons, since large attachments do not contribute measurably to the spam determination process but they would normally require transmission to the arbitron. Then, any MIME entities that are encoded are decoded so that the arbitron is dealing only with plain text.

Prior to invoking the spam arbitron, the user's .sendmailfilter configuration file plus the local and global Spam Assassin (if the spam arbitron chosen is spamd) configuration files are read to extract white and black list information for the recipient of the message. This information is then acted upon to determine if a message is spam. Next, if the white and black lists don't result in a spam/non-spam determination, statistical information (see "Statistical Tests", below) about the message is examined to look for spam-like qualities that indicate the message is spam. If a determination can be made about a message without invoking any arbitrons, this is done to speed up spam processing, since calling spam arbitrons is generally slow.

Upon detection of a message's sender on a black list, the detection of spam through statistical methods or the receipt of the arbitron's decision, the message is disposed of according to the options "-sc", "-sd" and "-st". If the "-st" (trash) option is chosen, the message is dumped and that's that. If one of the other two options are chosen, the results from black list processing or from the arbitron are inserted into the message, depending on which of the three options "-s1", "-s2" or "-s3" are chosen.

Message formatting for level one (command line "-s1" or config file "SpamLevel 1") consists of inserting a single header into the message, giving spam percentage values (any message with a value over 100% is considered to be spam) and inserting a paragraph into the message body, explaining that the message is spam (this is all that will be done for a black listed message, regardless of what message formatting level is chosen). Level two formatting (command line "-s2" or config file "SpamLevel 2") adds any pertinent headers generated by the spam arbitron to the message as well. Level three formatting (command line "-s3" or config file "SpamLevel 3") inserts a report, if any, from the spam arbitron into the message body, as a paragraph,

following the explanatory paragraph inserted by level one.

If the immediate delivery option ("-sd") is chosen, the spam is tagged with a subject prefix of "[SPAM]:" and sent on its merry way, after any virus filtering, etc. is done. If the corral option ("-sc") is chosen, the spam is filtered and prepared for delivery and then sidelined in a corral directory where a program such as SpamCorral can find it and send out receipt notifications.

Meanwhile, sendmail is instructed to reply to the sender or not, depending on the value of the "-r" parameter. A value of zero causes no replies to be sent to the sender. A value greater than zero causes no more than that many replies to be sent to the sender in a predetermined period. After that threshold is reached no more replies are sent until the end of the period passes. If a reply is sent, it has a major and minor result code of "550" and "5.7.1" plus a suitable message text that says the message is spam. Perhaps it is a bit naive to expect that spammers will do anything about this kind of deliver notification so this feature is off by default.

2.6.1 Statistical Tests

The statistical tests that are applied to messages to render a spam determination are fairly simple and, hence, quite rapid and yield a very high true positive score when applied to messages sent via email. They rely on the fact that normal email messages do not employ many of the tricks used by spammers to circumvent spam filtering. By employing these tricks to get around content-based filters, the spammers are essentially flagging their email messages as spam (how thoughtful of them). Here is a description of the tests applied (all tests are applicable to HTML messages only):

Content-based spam identifiers (e.g. Bayesian) look at the words in a message to identify whether it is spam or not. Spammers insert HTML comments into the middle of words (e.g. Via<!--junk-->gra) to break the words up so that content-based identifiers will not see the trigger words that indicate spam. Regular email messages never insert comments into the middle of words so a ratio of the number of embedded comments to words gives a good indication of spamishness. Even a very small ratio is an excellent indicator.

Spammers use tables extensively, even going to the point of aligning single characters in table colums to create words that are not detected by content-based spam identifiers. Regular email messages do not use tables to the extent that spammers do so a high ratio of table tags to regular words can indicate spam.

Image spam presents a message through an image that is loaded from a Web server, thereby resulting in a message with zero or almost no text content, hence nothing is available for filtering. To a lesser extent, text-based spam (which is basically advertising) employs images to present a more visual message, since visual messages are apparently more appealing. Regular email, although it often includes images, seldom has these images embedded inline in a message and does not include inline images in high proportions to text. Hence, a high ratio of inline images to regular text is a good predictor of spam.

The strength of a good spam campaign can be automatically judged by embedding links to feedback Web sites in such a manner that, when an email message is opened, the Web site is accessed and the recipient's identity is transmitted. In this way, a spammer can know who has received their spam and opened it. Links of this nature are embedded into a message as innocuous images. However, real images seldom include email addresses or parameters containing identifiers. Thus, any images that include such information should receive a high spam score.

Similarly, links to Web pages that include email addresses as parameters (not "mailto:" type links but links to a CGI script or Web page) probably indicate a spammer attempting to provide feedback to themselves with the recipient's email address. Consequently, these types of links are assumed to be indicative of spam.

Perhaps there are legitimate reasons to encode plain text and HTML as if it were binary data (e.g. alternate character sets) but the most common use of this technique has been to obscure spam and viruses from detection by scanners. This being the case, any text or HTML MIME entities that are encoded Base64 are given a high statistical spam score which, while not high enough to win a message a spam label all on its own, it is high enough to ensure that any other indiscretions will push it over.

Note that the final statistical score for a message is the sum of all of the statistical tests that are enabled. These tests are employed because they often provide a rapid determination of spam and can work very well, in many instances. Mind you, not everyone is bound to agree with these statistical definitions of spam so any and all of the tests can be disabled or adjusted to fit individual preferences. The statistical filtering criteria can be tuned by altering the values assigned to the "StatXxxx" options in the configuration file (see the "Spam Processing Options" section for more information on these options).

2.6.2 Spam Report

If a message is found to be spam, information to this effect is added to the message before it is sent on to the user (or corralled in the corral). The text that is added to the message is governed by the values that you set in smfopts.h, as well as settings in the system and user configuration files. However, for the purposes of this section, we will assume all of the standard settings are kept.

If message formatting is set to level one, either through the command line parameter "-s1" or the config file setting "SpamLevel 1", the spam report consists of inserting a single header into the message, giving spam percentage values (any message with a value over 100% is considered to be spam) and inserting a paragraph into the message body, explaining that the message is spam (this is all that will be done for a black listed message, regardless of what message formatting level is chosen). Here is an example:

```
 To: jdoe@anon.com
Subject: A hard man is good to find
Date: 2009 Jan 17
X-Spam-Stats: Local 0%, System 0%, Scanner 150%, Score 150%.

This message matched the criteria for spam, determined by your personal
spam filter parameters or the global or system spam filter parameters.

The regular body of the message follows....
```

Note that the percentage values in the X-Spam-Stats header represent the results of the various tests against the message, as follows:

Local	The result of applying the user's white and blacklists. If the blacklist matches, the value is 100%. Otherwise, it is 0%. Generally, unless modified by a special "SpamRule", a hit on a blacklist results in all other checks being bypassed, as does a hit on a whitelist.
System	The result of using Mailcorral's built-in spam scanners (e.g. statistical scan).
Scanner	The result of scanning the message with any third party (e.g. spamd) or programmable arbitrons. The value returned by the arbitrons is scaled so that a determination of "Spam" results in a score of 100%.
Score	The maximum of the other three spam values or, in other words, the score that got the message flagged as spam or not.

If message formatting is set to level two, either through the command line parameter "-s2" or the config file setting "SpamLevel 2", the spam report consists of inserting the same header as above into the message, along with any headers returned by the arbitrons, and inserting a paragraph into the message body, explaining that the message is spam. Here is an example:

```
 From: Joe Blow <joe@blow.com>
To: jdoe@anon.com
Subject: A hard man is good to find
Date: 2009 Jan 17
X-Spam-Flag: YES
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on
    space-port.homeworld
X-Spam-Level: *******
X-Spam-Status: Yes, score=7.5 required=5.0 tests=AWL,DRUGS_ERECTILE,
    HTML_MESSAGE,IMPOTENCE,MIME_HTML_ONLY,NO_RELAYS autolearn=no version=3.2.5
X-Spam-Report:
    -0.0 NO_RELAYS              Informational: message was not relayed via SMTP
     2.6 IMPOTENCE              BODY: Impotence cure
     0.0 HTML_MESSAGE           BODY: HTML included in message
     2.3 MIME_HTML_ONLY         BODY: Message only has text/html MIME parts
     2.4 DRUGS_ERECTILE         Refers to an erectile drug
     0.2 AWL                    AWL: From: address is in the auto white-list
X-Spam-Stats: Local 0%, System 0%, Scanner 150%, Score 150%.

This message matched the criteria for spam, determined by your personal
spam filter parameters or the global or system spam filter parameters.

The regular body of the message follows....
```

If message formatting is set to level three, either through the command line parameter "-s3" or the config file setting "SpamLevel 3", the spam report consists of all of the above plus any reports returned by the arbitrons, inserted after the paragraph that is inserted into the message body. Here is an example:

```
 From: Joe Blow <joe@blow.com>
To: jdoe@anon.com
Subject: A hard man is good to find
Date: 2009 Jan 17
X-Spam-Flag: YES
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on
    space-port.homeworld
X-Spam-Level: *******
X-Spam-Status: Yes, score=7.5 required=5.0 tests=AWL,DRUGS_ERECTILE,
    HTML_MESSAGE,IMPOTENCE,MIME_HTML_ONLY,NO_RELAYS autolearn=no version=3.2.5
X-Spam-Report:
    -0.0 NO_RELAYS              Informational: message was not relayed via SMTP
     2.6 IMPOTENCE              BODY: Impotence cure
     0.0 HTML_MESSAGE           BODY: HTML included in message
     2.3 MIME_HTML_ONLY         BODY: Message only has text/html MIME parts
     2.4 DRUGS_ERECTILE         Refers to an erectile drug
     0.2 AWL                    AWL: From: address is in the auto white-list
X-Spam-Stats: Local 0%, System 0%, Scanner 150%, Score 150%.

This message matched the criteria for spam, determined by your personal
spam filter parameters or the global or system spam filter parameters.

Spam detection software, running on the system "space-port.homeworld", has
identified this incoming email as possible spam.  The original message
has been attached to this so you can view it (if it isn't spam) or label
similar future email.  If you have any questions, see
postmaster for details.

Content preview:  Untitled Document Herbal Alternative for Erectile Dysfunction
Men of Iron has been featured on over 100 TV News and Top Radio stations
across America, and we know why... It REALLY works! Visit Our Web Site Click
Here: Learn about our special offer! [...]
```

```
Content analysis details:   (7.5 points, 5.0 required)

pts rule name              description
---- ---------------------- --------------------------------------------------
-0.0 NO_RELAYS              Informational: message was not relayed via SMTP
 2.6 IMPOTENCE              BODY: Impotence cure
 0.0 HTML_MESSAGE           BODY: HTML included in message
 2.3 MIME_HTML_ONLY         BODY: Message only has text/html MIME parts
 2.4 DRUGS_ERECTILE         Refers to an erectile drug
 0.2 AWL                    AWL: From: address is in the auto white-list

The original message was not completely plain text, and may be unsafe to
open with some email clients; in particular, it may contain a virus,
or confirm that your address can receive spam.  If you wish to view
it, it may be safer to save it to a file and open it with an editor.

The regular body of the message follows....
```

2.7 Virus Handling

Enhanced virus handling (other than the basic checking of file type and MIME type lists that is built in) is carried out in cooperation with a virus arbitration daemon (arbitron), such as ClamAV) or one or more of the user's programmable arbitrons. MailCorral selects all of the attachments and inline components of each received message that are likely places for a virus to lurk. It then sends a request for virus determination, via a pipe to the arbitron, which contains each of the attachments and/or components and the arbitron renders a decision as to whether any of them conatins a virus or not. The arbitron's decision is final. Or, in the case of the user's programmable arbitrons, the attachments and/or components are passed in files but the rest of the details are the same.

After all virus checking (either internal or external arbitron or both) is done, any attachments or components marked as viruses are deleted, while the others may be marked with a warning or have their names permuted to prevent them accidentally being opened/lauched. An unaltered version of the message is placed in the corral where it can be retrieved later on, if necessary.

2.8 Command Line Parameters

This filter is actually invoked by the system startup script for sendmail. It is a daemon which listens, on the port specified, for milter requests from sendmail. The command line parameters that are passed to the filter from the startup script (see Section 1.5) govern the filter's actions, plus define the communications connection with sendmail and the message type arbitrons. They are:

-A Supplies the name of a file or list of files, separated by semi-colons, that contains the name of all of the local aliases, for the purposes of allowing the filter to determine whether local mail is actually deliverable or not (non-deliverable local mail will not be filtered but is left untouched, instead, thereby leaving sendmail to make the actual determination whether the mail can be delivered or not). The name supplied by this parameter overrides any name supplied by the "AliasList" option in the config file.

Usually, one would point this option at the standard aliases file (e.g. "/etc/aliases") employed by sendmail. The filter is capable of reading and processing this file, except for two small differences (which should have no real effect on the utility of the file): it does not support includes; lines must be continued by a '\' in the last position on any continued line. The filter

also assumes that any name in this file is not bogus (i.e. that mail will actually be deliverable to any user named therein).

It is important, in order for filtering to work properly, that the filter know when a local user really exists and when they do not. Sendmail does some of the work, before it calls the filter, by determining whether the user is local but it does not determine if mail can actually be delivered to the user (until later on, that is). Thus, unles the filter decides this for itself, it could do work unnecessarily or, worse, create corral files for nonexistant users.

The filter looks up all local users in the password file to see if they are valid. If so, it assumes that mail can be delivered to them. However, this is insufficient. Aliases will not be found in the password file, yet delivery to them is valid. Hence the need for the filter to know what alias names are used.

-C Specify the name of the global options file to be used at startup. The default names, if not supplied, are: "/etc/mail/sendmailfilter.cf;/etc/sendmailfilter.cf".

-c Gives the name of the local options DBM database to be used to resolve lookups for individual user options. The options for all users are stored in this one file, under username keys.

The database lookup feature is not used, if this option isn't specified. If it is, the local filter and SpamAssassin config files are ignored and only the DBM database is used instead. This option overrides the "ConfigDB" config file option.

The database name given should not include the ".dir" and ".pag" extensions, since these are added gratuitously by DBM. The database itself can be built by the "ConfigEdit.cgi" program (see the chapter on "User Support") or directly by a program of your choice.

Note that the name used to look up the user options depends on the setting of the "-q" or "QualifyNames" flag.

-D This option supplies the list of domain names that are used to determine whether mail is being delivered locally or not. Any domain names in this list are considered local, for purposes of the "-e" or FilterExt and "-i" or FilterInt options. This option overrides the "DomainList" config file option.

The list may include one or more domain names, separated by commas. It may also include file names. Any name that begins with a '/', '~' or '.' is assumed to be a file name.

If a file name is given, the file should contain a list of local domain names, one per line. Blank lines and comments beginning with '#' are ignored. Typically, this feature would be used to point to the file "/etc/mail/local-host-names" or the same file where sendmail gets local domain name information from.

The domain name "localhost" is forced onto the list at the end, if it isn't specified. This name is always assumed to be a local domain name by sendmail.

Note that the filter attempts to automatically determine whether a recipeint's name is local or not, regardless of the contents of this list so using it is partly supplemental. While there is no harm in giving the filter a list of all local domain names (e.g. "/etc/mail/local-host-names"), it is not strictly necessary. However, if you have domains that are not local to this system but

are still considered internal to your network, you might wish to include them in this list so that messages delivered to addresses in these domains will not get filtered, when sent from local users. Be aware, however, that some spammers spoof the same domain name for the sender as the recipient, thereby bypassing spam checking entirely, if the domain list includes the domains of local recipients.

It is much better, for the purposes of bypassing filtering from internal users, to include numeric IP addresses or address ranges within the domain list. If the domain list includes numeric IP addresses, the sender's IP address only (recipient IP addresses are not checked) will be compared against all of the IP addresses in the domain list. Any that match will be considered local senders.

Numeric IP addresses can be of the form "n.n.n.n" or "n.n.n.n/m". In the first case, the IP address of the sender must match the address given exactly. In the second case, the value "m" is a mask (range 0-32) that indicates how many high-order bits in the address are checked. For example, "192.168.1.0/24" will match all addresses in the range "192.168.1.0" to "192.168.1.255" while "192.168.0.0/16" will match all addresses in the range "192.168.0.0" to "192.168.255.255". The mask "/0" is equivalent to the mask "/32".

-Da — Add to the list of users that are included in the local domain (i.e. internal users) any sender that validates to SMTP via an AUTH protocol (whichever one it is). This option may be useful to ISPs who have many external users who need to be treated as virtual internal users when they connect to SMTP. This command line option has the same effect as supplying the value "Auth" to the "DomainIncl" global configuration parameter.

-Dl — Treat any sender that connects to SMTP via the local IP address (127.0.0.1) as if they were included in the local domain (i.e. an internal user). This command line option has the same effect as supplying the value "Local" to the "DomainIncl" global configuration parameter.

-d — Turn on debugging (regardless of whether a debug file name was compiled into the filter). The name of the file where the debugging information is written must also be supplied as the value of this parameter. This will override any compiled-in file name as well as any value set by the "DebugFile" config file option.

Note that, if you expect error messages from options processing to be written to this debug file, this option must appear first on the command line. If this option isn't first, errors from option processing are always written to the sendmail syslog file (wherever that is) anyway.

-d0 thru -d4 — Set the debug level to 0 thru 4, overriding any value specified by the "DebugLev" configuration file option. Increasing the debug level turns on tracing of progressively more and more detailed debugging information. Zero turns it off. One traces high level work. Two traces basic work. Three traces low level work. Four traces message transport. If you'd like your debug file to get filled up fast, pick level 3 or 4. The default is level 2.

-e — Turn on filtering of messages sent externally (i.e. messages from someone inside the domain to someone outside the domain). Any setting by the "FilterExt" option in the config file is overridden.

-h — Turn on HTML filtering for all HTML tags, regardless of where they occur in a message. Only really harmless tags are left intact (e.g. <p>,
). Only the setting of the "ReplaceHTML" option in the global config file is overridden by this flag.

If this flag is on, the filter removes all HTML that could be nasty. This means looking at tag

names plus individual parameters within the tags. The tables found in smfopts.h are used to do these checks. If this flag is off, only the tags that are really nasty (plus embedded comments in tag names) are removed. The really nasty tags are: iframe, object, script, applet and embed, since they can launch code. These tags (and anything in between them and their terminators) are always removed, if you ask for virus checking.

-i Turn on filtering of messages sent internally (i.e. messages from someone inside the domain to someone else also inside the domain). Any setting by the "FilterInt" option in the config file is overridden.

-k Keep all messages filtered, regardless of whether they are altered or not, in the corral. This option can be useful for debugging or for anyone needing a complete audit trail of all messages processed. If the "KeepAll" option is set in the config file, it is overridden.

Please bear in mind, when using this option, that only messages that are filtered can be kept. If a message is not filtered because it is bypassed (e.g. internal -> external), it will not be kept. Sorry, that's just the way it has to be. However, if you use the either the "ArchiveDirectory" or "ArchiveProgram" configuration file option to archive messages, they do not suffer from this restriction.

-L Set the directory used for language support to the name given. This parameter is used in conjunction with the "Language" local configuration option (see that option for details) as well as the "X-Accept-Language" header in received messages. It overrides any value set in the global configuration file by the "LanguageDirectory" option.

If a message is received that includes an "X-Accept-Language" header, the language value supplied in that header is used to set the name of the language configuration file used by the filter to generate message inserts. The language value found is appended to the language directory supplied by this option and then the suffix ".cf" is appended.

Typically, the language values used in "X-Accept-Language" headers are two character names, as specified by the mail/MIME RFCs. Note that, prior to composing the file name, the the language value is lowercased to insure consistency. The language value in the header is used exactly to compose the file name. For example, if this option is set to "/etc/mail" and the "X-Accept-Language" header contains "fr", the following file will be processed:

/etc/mail/fr.cf

The file is opened and processed just like a regular local configuration file, except that the only parameters which are valid are those starting with "Msg" (from the Message Formatting Options section of the Configuration chapter), that is to say all of the message insert texts. The idea is to allow a separate configuration file to be created for each language that is supported and for this file to allow the text of all of the message inserts in it to be written in that language. When a message is received in a particular language, the text of all of the message inserts used is loaded from that language's configuration file.

-os Sum all of the local user options from the configurations of all of the recipients of a message to arrive at one, single set of options to be used for processing the message. If supplied, this parameter overrides "SumOptions" in the config file.

Normally, options from the configuration of the first or only local recipient of a message are used, in conjunction with the global options, to process each message (messages are only processed once, regardless of how many local recipients a message is bound for and non-local

recipients are irrelevant, since some other system will deliver the message to them). If this option is chosen, the configurations of all of the recipients of a message are read and then all of their options are summed up to create a compound set of options that represent all of their preferences. This compound set of options is then used to process the message (it is still only processed once).

Option summation takes place as follows: for switches ("AutoRemail", "ReplaceHTML", "SpamAlways" and "SpamFastPath"), the "No" setting overrides the "Yes" setting; for the "UnknownDispo" option, the "MIME" setting overrides the "Ignore" setting, while the "Warn" setting overrides both; for the "SpamDel" option, the "trash" mode is overridden by the "deliver" mode which is overridden by the "corral" mode; for the "SpamLevel" option, the highest level is chosen; for the "StatXxxxRatio" options, the highest threshold found is used and for the "StatXxxxBoost" options, the lowest boost value found is used; for the "AddXTags" option, the "Yes" setting overrides the "No" setting for each tag ("envelope", "version") that is set in any configuration processed; and only the first recipient's settings are used for the "Language", "Msg*" and "SpamProto" options.

-p The communications protocol to use in talking to sendmail. This must match the value given in the sendmail configuration file via the mail filter option. For example, if sendmail's m4 config file says:

INPUT_MAIL_FILTER(`filter1',`S=inet:2526@localhost, F=R')dnl

the value passed via "-p" should be:

-p inet:2526@localhost

According to the sendmail documentation, the possibilities are:

{unix|local}:/path/to/file *A named pipe.*
inet:port@{hostname|ip-address} *An IPV4 socket.*
inet6:port@{hostname|ip-address} *An IPV6 socket.*

One way or another, this option must be specified in order for the filter to operate. It must either be supplied via this command line switch or in the global config file via "SendmailProto". This parameter cannot be empty. The command line switch overrides the config file option.

-p1 thru -p24 Select the interval to partition the corral into. The number given is the number of hours to use for the partition interval. Acceptable values are 1, 2, 3, 4, 6, 8, 12 and 24. A value must be specified, there is no default. This value overrides any global config file value set by the "Partition" option.

Partitioning allows the number of filtered messages, stored in the corral, to be dramatically increased. Normally, an unpartitioned corral will hold all of the messages filtered on a small to medium sized system. However, on some large systems (e.g. those used in production environments and by ISPs), the number of messages that must be stored in the corral can easily exceed reasonable limits for the file system and preclude notification and message handler programs from being able to examine and process corralled messages.

Partitioning circumvents this problem by splitting the corral into separate subdirectories, based on the time interval specified. If a value of 24 is chosen, for example, the corral will be

split into chunks that each contain 24 hours worth of corralled messages. These smaller subdirectories can be more easily processed by notification and message handler programs.

All filtered messages are partitioned with this option. Spam, may also be partitioned post facto (e.g. if the corral fills up and you need to partition immediately as a quick fix to a space problem) by the separate notification program specific to it. A reasonable partition interval is 24 hours. Lesser intervals may be chosen if you still experience problems with the volume of messages in the corral.

-q Use or do not use fully qualified names (i.e. recipient name plus domain name) for any configuration database lookup of user options that is done in conjunction with the "-q" parameter. This parameter overrides the "QualifyNames" configuration file option.

Users of this option typically have virtual users who do not have any real username or home directory on the machine where mail is delivered, hence the need to store their configuration information in a common database. Since it is possible for the same username to exist in two domains (e.g. "custserv"), the domain information must be used to qualify the username and ensure that the name used to store configuration information is unique.

Note that spam saved in the spam corral is automatically saved using only the recipient's name (domain information is stripped) for local users and using the fully qualified name for all others. This is done so that all spam to a single local recipient, regardless of how it was addressed to them, will be corralled under a single name. The advantage of this behavior is that it results in the need to send only a single notification message to each spam recipient when telling them about the spam they've received. However, bear in mind that even using this scheme, alias names will have their spam stored as a separate user, regardless of who the alias resolves to.

-r If spam filtering is done, the number of replies that should be sent to a spammer (per day), telling them that what they are sending is spam. Once the threshold is reached for a particular spammer in a single day, all subsequent messages from them are just dumped with no reply. The default is 0 (i.e. always just dump all spam). The maximum is 25. Any value set by "SpamReplies" in the configuration file is overridden.

This option is necessary because some spammers use autoresponders to reply to any mail sent to them. This may well appear as spam too, in which case, a reply will be sent to it. Do you see the potential for ping pong? I do.

-rm Allow automatic remailing of filtered messages. Without this option, whenever a message is altered, a copy of it is kept and the user is supplied with the name of the file where it is stored so that it can be used to fetch the message. They are also given the name of a Tech Support person whom they must contact to retrieve the message. This is the safest way of dealing with viruses, since the Tech Support person can quiz the user to see that they know what they're doing before releasing a possibly infected file.

With this option, the user is given the user name of a mail handling robot that can release the message to them. Upon receipt of a message from the user, with the subject supplied, the mail handling robot will release the original content of the message to the system for remailing to the recipient. Note that this method of handling infected messages, while completely automatic, is dangerous, since the unsuspecting user can have possibly infected messages delivered directly to them without filtering.

The global config file option "AutoRemail" only may be overridden by this switch.

-s The communications protocol and port to use in talking to a spam arbitration daemon. The protocol must be one that this filter knows about and the port must match the one that the daemon will be listening on.

Currently, the supported protocols are:

internal - The internal, fast path spam checker.
spamd - The Spam Assassin daemon.

Except for the "internal" protocol, the port number and host name or IP address must follow the protocol name, separated by a colon and an at sign. So, the entire parameter should look like one of the following:

internal

or

proto:port@host

For example, if you are running spamd and it is listening on port 2527 on the local machine, the value passed via "-s" should be:

-s spamd:2527@localhost

By supplying a valid protocol and port, you will cause all messages received to be sent to the spam arbitration daemon. If it thinks the message is spam, it will be treated in as such. If you don't supply this parameter, only the internal spam checks will be performed (blacklists/whitelists and statistical).

Note that when **any** spam arbitron is used, the internal, fast path spam checker is always run (unless the "-ss" option is set or "SpamFastPath" is set to "No"), prior to calling the arbitron selected, in an attempt to speed up spam checking by bypassing the overhead involved with calling the arbitron.

An optional timeout value may be supplied for any of the arbitrons chosen. If this parameter is supplied, it must follow the protocol, port number and host name immediately and be enclosed in parenthesis. For example:

-s spamd:2527@localhost(20)

The value given is the time, in seconds that MailCorral is prepared to wait for the entire transaction with the spam arbitron. Essentially, this is the total time for the arbitron to respond to the spam determination request -- it includes the elapsed time for all of the reads and writes to the arbitron. In the above example, the time allotted to SpamAssassin would be twenty seconds.

The default timeout is set to 30 seconds, if no value is supplied. You may pick any positive value but consider this. Sendmail is only prepared to wait for an answer from the filter for so long. If the arbitron timeout is to be of any use, it should be set to some value smaller than the

timeout given to sendmail in its configuration file (no value in the sendmail configuration file means a default timeout of 10 seconds).

A good choice would be to give the arbitron 60-80% of the timeout allotted to the filter in the sendmail configuration file. This will ensure that the filter can still complete the job of filtering a message in the time allowed, despite the fact that the arbitron times out while making its decision. And, if a virus arbitron is also being used (see the "-v" option or "VirusProto"), the time allocated to both the arbitrons should be split between them, as you see fit. In that case a good choice would be to give 30-40% to each of the two arbitrons.

Note that a time limit may be supplied to the internal arbitron but there isn't much reason for this, since it executes extremely quickly. When calculating how much time to allow the filter and its arbitrons, you can neglect the time consumed by the internal arbitron.

The global config file option "SpamProto" can be overridden by this command line parameter.

-s1 thru -s3 Set the level of spam reporting to one thru three, overriding any global config file value set by the "SpamLevel" option. The default is "-s1".

Level one causes a single header line to be inserted in any message that is found to contain spam, giving the spam processing statistics as three percentage values.

Level two causes the same header line described under "-s1" to be inserted in any message that is found to contain spam. It also inserts any headers generated by the spam arbitron (selected by "-s") into the message.

Level three does everything that levels one and two do, plus it formats any report generated by the spam arbitron for insertion into the message body as a paragraph of text.

-sa Always add the spam report to mail, regardless of whether it is spam or not. The global option "SpamAlways" may be overridden by this switch.

-sc Set the spam delivery mode to "corral", the default. This mode will cause any spam received to be sidelined in the spam corral, where it can be processed by a spam notification program and/or released for delivery by the recipient at a later date. The choice made by the global option "SpamDel" may be overridden by this switch.

-sd Set the spam delivery mode to "deliver". This mode will cause any spam received to be marked as such and then delivered to the recipient without further delay. The default is "-sc". Any value set by the global option "SpamDel" is overridden by this switch.

-ss Turn off the spam fast path check, prior to calling the designated spam arbitron. The statistical spam fast check is turned off by this flag, too. If the designated arbitron is "internal" this flag has no effect, since it is possible to turn off the use of the internal check as the designated arbitron by not specifying it at all. This switch will override the "SpamFastPath" config file option.

-st Set the spam delivery mode to "trash". This mode will cause any spam received to be tossed directly in the trash can, straight away. I like it! Unfortunately, the default is "-sc". The choice made by the global option "SpamDel" may be overridden by this switch.

-t Turn on filtering of messages transiting the system (i.e. messages from someone outside the domain to someone else outside the domain). The transit setting is typically used by ISPs who

are delivering mail for many other systems on a mail handling gateway. Any setting by the "FilterTrans" option in the config file is overridden.

-u A proxy userid to use in looking up mail disposition and spam processing options. Normally, the recipient's userid, stripped of domain information, is used to look up user-specific mail disposition options plus spam processing options in their home directory. However, if the recipient doesn't have a home directory and if a proxy userid is given, the options in the proxy userid's home directory will be used instead.

This option might be useful to ISPs, who process mail for many users but who do not have userids set up for all of them. Usually, mail delivery is governed by user-specific options found in a ".sendmailfilter" file in each recipient's home directory. However, if the recipient has no home directory, the ".sendmailfilter" file in the proxy userid home directory is used.

A similar situation arises with Spam Assassin, if spamd is used to arbitrate whether a message is spam. Spam Assassin looks in the recipient's home directory for options such as whitelists. If the user doesn't have a home directory, the home directory of the proxy userid is used instead.

This option overrides any value set by the "ProxyUser" config file option.

Note that the values set for the "IgnoreSpam" and "IgnoreVirus" options in a proxy user's configuration have no effect on users who have no configuration. Such users will have their viruses and spam filtered by default and there is no way to avoid this, short of setting up configuration information for them. It was felt that virus and spam filtration was too important to get bypassed by accident.

-ui Select a disposition of "Ignore, for unknown file types found as attachments to a message. Selecting a disposition of "Ignore" will cause attachments with unknown file types to be ignored and treated as if they were perfectly OK. No warning will be issued for any unknown file types. This setting is more than somewhat dangerous, since it is quite possible that a file type which the filter doesn't know about may be harmful. Not receiving any warning could allow a message recipient to inadvertently open a harmful attachment. This option overrides any disposition choice made by the "UnknownDispo" option in the global configuration file. The "-um" and "-uw" flags also affect the disposition choice.

-um Set the disposition for unknown file types, found as attachments to a message, to "MIME". The "MIME" disposition will cause the filter to take additional steps to determine the file type, before issuing a warning. It will look at the MIME type for the attachment and, if it is one of the acceptable types, no warning will be issued. Otherwise, a warning will be issued, as below, for the "-uw" disposition. This option overrides any disposition choice made by the "UnknownDispo" option in the global configuration file. The "-ui" and "-uw" flags also affect the disposition choice.

-uw Set the disposition for unknown file types, found as attachments to a message, to "Warn". A disposition of "Warn" will cause a warning to be inserted into the message whenever an attachment is found with an unknown file type (i.e. it has no extension or an extension that is not known). This is the default setting, since it is quite possible that a file type which the filter doesn't know about may be harmful and the assumption is made that it is best to warn about these things. This option overrides any disposition choice made by the "UnknownDispo" option in the global configuration file. The "-ui" and "-um" flags also affect the disposition choice.

-v The communications protocol and port to use in talking to a virus arbitration daemon. The protocol must be one that this filter knows about and the port must match the one that the daemon will be listening on.

Currently, the supported protocols are:

clamd - The ClamAV daemon.
internal - The internal virus checker.

Except for the "internal" protocol, the port number and host name or IP address must follow the protocol name, separated by a colon and an at sign. So, the entire parameter should look like one of the following:

internal

or

proto:port@host

For example, if you are running clamd and it is listening on port 2528 on the local machine, the value passed via "-v" should be:

-v clamd:2528@localhost

By supplying a valid protocol and port, you will cause all messages received to be sent to the virus arbitration daemon. If it thinks the message contains a virus, it will be treated in an appropriate manner. If you don't supply this parameter, only the internal virus checks will be performed (MIME type/attachment type 0 matching).

Note that when **any** virus arbitron is used, the internal virus checker is always run (unless the "-vv" option is set or "VirusChecker" is set to "No"), prior to calling the arbitron selected, but its results will not necessarily cause the virus arbitron to be bypassed. Only in the case of MIME entities or attachments that the internal virus checker marks for deletion will the external arbitron be bypassed, and such instances are admittedly quite rare.

An optional timeout value may be supplied for any of the arbitrons chosen. If this parameter is supplied, it must follow the protocol, port number and host name immediately and be enclosed in parenthesis. For example:

-v clamd:2528@localhost(20)

The value given is the time, in seconds that MailCorral is prepared to wait for the entire transaction with the virus arbitron. Essentially, this is the total time for the arbitron to respond to the virus determination request -- it includes the elapsed time for all of the reads and writes to the arbitron. In the above example, the time allotted to ClamAV would be twenty seconds.

The default timeout is set to 30 seconds, if no value is supplied. You may pick any positive value but consider this. Sendmail is only prepared to wait for an answer from the filter for so long. If the arbitron timeout is to be of any use, it should be set to some value smaller than the timeout given to sendmail in its configuration file (no value in the sendmail configuration file means a default timeout of 10 seconds).

A good choice would be to give the arbitron 60-80% of the timeout allotted to the filter in the sendmail configuration file. This will ensure that the filter can still complete the job of filtering a message in the time allowed, despite the fact that the arbitron times out while making its decision. And, if a spam arbitron is also being used (see the "-s" option or "SpamProto"), the time allocated to both the arbitrons should be split between them, as you see fit. In that case a good choice would be to give 30-40% to each of the two arbitrons.

Note that a time limit may be supplied to the internal arbitron but there isn't much reason for this, since it executes extremely quickly. When calculating how much time to allow the filter and its arbitrons, you can neglect the time consumed by the internal arbitron.

The global config file option "VirusProto" can be overridden by this command line parameter.

-va Always add the virus report to mail, regardless of whether it contains any viruses or not. The global option "VirusAlways" may be overridden by this switch.

-vv Turn off the internal virus checker, prior to calling the designated virus arbitron. If the designated arbitron is "internal" this flag has no effect, since it is possible to turn off the use of the internal checker as the designated arbitron by not specifying it at all. This switch will override the "VirusChecker" config file option.

-X Indicate which additional headers should be added to any messages delivered. The string that follows tells which headers should be added. Multiple header names may be supplied in a comma separated list. The choices are:

env[elope]	Add headers giving the envelope information, such as from and to address.
ver[sion]	Add a version header for MailCorral.

For example: "-Xenv,ver".

This option overrides any value set by the global "AddXTags" config file option. However, the local config file may be used to turn off the value set by this parameter, on an individual user basis.

2.9 Performance Expectations

No filtering of mail messages can be expected to come for free, especially not the extensive level of filtering that is done by MailCorral. However, the filter was built with performance in mind and it has been shown to perform very well in production environments.

A typical user of MailCorral is an ISP, with thousands of users, who runs sendmail on a dedicated (or mostly dedicated) mail server. In real environments, we have observed sustained loads of 2-4 email messages per second (which translates approximately to 10,000 messages per hour or a quarter million messages per day). The performance numbers stated below are typical for server loads such as this and you can expect similar numbers on your server.

Adding filtering to the email server increases the load on the system such that the filter (plus the spam and virus arbitrons) consumes between 25-45% of the CPU. The filter itself usually accounts for 15-20% of the CPU and the typical spam arbitron (e.g. SpamAssassin) usually accounts for 5-10% of the CPU. The virus arbitron can account for another 10-15% of the CPU.

To put it another way, if your CPU is more than 55-60% busy running sendmail, you may experience some performance problems by adding filtering. If it is less than 50% busy, you should see no impact. When

considering an upgrade to a machine running flat out, a 65% increase in horsepower should easily accomodate filtering.

The guidelines above should help you in planning your mail filtering setup. If you would like more specific performance numbers, please contact BSM Development and tell us what you'd like to see.

3. Configuration

The operation of the sendmail filter can be altered by configuration parameters in two configuration files, one a global configuration file that applies to all of the messages processed and the other a local configuration file that only applies to the messages for an individual recipient (typically, there is one such file for each recipient, in their home directory). Usually the global configuration file is "/etc/mail/sendmailfilter.cf" or "/etc/sendmailfilter.cf" and the local configuration file is ".sendmailfilter", in the recipient's home directory. However, these file names may be changed when the filter is built, at compile time, by editing GLOBALOPTIONS and LOCALOPTIONS in "smfopts.h". See the Installation Section for more information.

The format of the configuration files is the same as that employed by many Unix-style configuration files. Comments are allowed and are indicated by '#'. They may appear on a line by themselves or at the end of any line with a parameter on it. Once the '#' is seen, everything that follows it is ignored until the end of the line. Blank lines are permissible and are ignored. Leading and trailing whitespace is ignored. Configuration parameters must appear one to a line. The parameter name must be separated from its value by at least one whitespace. The presence of a switch type parameter name is sufficient to set its default setting. Multiple occurrences of a parameter are permissible but multiple values must be specified by repeating the parameter name. Multiple values may not appear on a single line.

Parameter values may be continued on multiple lines by ending each line, except for the last, with '\'. The continuation may appear anyplace in the parameter value. Whitespace preceding the '\' and beginning the continued line is thrown away.

The message string parameters accept quoted strings that are identical to C format. Each string must begin and end with a quote. If a second quote appears after the first, accumulation of the parameter is ended until another quote appears. The standard escape sequences "\"", "\\", "\r", "\n" and "\t" are recognized. The sample in the Sample Configuration File Section shows all this.

The configuration files can be edited at any time. The sendmail filter must be restarted for new global options to take effect. Changes to the local configuration file take effect with the receipt of the next message destined for the user in question.

3.1 Global Configuration

Global configuration information is kept in one of the two files "/etc/mail/sendmailfilter.cf" or "/etc/sendmailfilter.cf" (unless changes were made to GLOBALOPTIONS in "smfopts.h" prior to compiling the filter). The first file found, in the order shown, is read and processed.

All of the Filtering Options and Spam Processing Options are available in this file. With one exception (the "-C" option, for obvious reasons), all of the command line parameters have parallel options in the global configuration file. Global config options are overridden by any options supplied on the command line.

3.2 Local (User Specific) Configuration

Local configuration information is generally kept in ".sendmailfilter", in the recipient's home directory (unless a change was made to LOCALOPTIONS in "smfopts.h" prior to compiling the filter). If the recipient doesn't have a password file entry (found in "/etc/passwd") or home directory or there are no permissions on this directory, the proxy username may be used instead, to look up options that apply to this user. The proxy username works just like a regular user name, in that the options file for it will be processed, if found and

applied to the recipient of a message.

Only some of the Filtering Options (the ones marked with an asterisk) and all of the Spam Processing Options are available in this file. All of the command line parameters that are applicable to individual message processing have parallel options in the local configuration file. Local config options override all other options, both from the global config file and any options supplied on the command line.

3.3 Filtering Options

The message filtering options apply to all messages that are seen by the filter. They control every aspect of filtering from how the filter talks to sendmail to what types of messages should be filtered to how the message are disposed.

Either of the configuration files can contain the message disposition options. All of the other options are only available in the global options file. Here is the list of message filtering options available in the configuration files:

AddXTags * — none|env|ver Indicate which additional headers should be added to any messages delivered. The string that follows tells which headers should be added. Multiple header names may be supplied in a comma separated list. The choices are:

- env[elope] — Add headers giving the envelope information, such as from and to address.
- ver[sion] — Add a version header for MailCorral.

For example: "AddXTags env,ver".

An empty parameter value may be specified in the local (user) options file to turn off any XTags that were turned on globally. In the global options file, this parameter cannot be empty. The global option may be overridden by the "-X" command line option.

AliasList — Supplies the name of a file or list of files, separated by semi-colons, that contain the names of all of the local aliases, for the purposes of allowing the filter to determine whether local mail is actually deliverable or not (non-deliverable local mail will not be filtered but is left untouched, instead, thereby leaving sendmail to make the actual determination whether the mail can be delivered or not). The name supplied by this option is overridden by the "-A" command line parameter.

Usually, one would point this option at the standard aliases file (e.g. "/etc/aliases") employed by sendmail. The filter is capable of reading and processing this file, except for two small differences (which should have no real effect on the utility of the file): it does not support includes; lines must be continued by a '\' in the last position on any continued line. The filter also assumes that any name in this file is not bogus (i.e. that mail will actually be deliverable to any user named therein).

It is important, in order for filtering to work properly, that the filter know when a local user really exists and when they do not. Sendmail does some of the work, before it calls the filter, by determining whether the user is local but it does not determine if mail can actually be delivered to the user (until later on, that is). Thus, unles the filter decides this for itself, it could do work unnecessarily or, worse, create corral files for nonexistant users.

The filter looks up all local users in the password file to see if they are valid. If so, it assumes that mail can be delivered to them. However, this is insufficient. Aliases will not be found in the password file, yet delivery to them is valid. Hence the need for the filter to know what alias names are used.

AutoRemail[ing] * — Yes|No Allow or disallow automatic remailing of filtered messages. Without this option, whenever a message is altered, a copy of it is kept and the user is supplied with the name of the file where it is stored so that it can be used to fetch the message. They are also given the name of a Tech Support person whom they must contact to retrieve the message. This is the safest way of dealing with viruses, since the Tech Support person can quiz the user to see that they know what they're doing before releasing a possibly infected file.

With this option, the user is given the user name of a mail handling robot that can release the message to them. Upon receipt of a message from the user, with the subject supplied, the mail handling robot will release the original content of the message to the system for remailing to the recipient. Note that this method of handling infected messages, while completely automatic, is dangerous, since the unsuspecting user can have possibly infected messages delivered directly to them without filtering.

The global option only may be overridden by the "-rm" command line switch.

ConfigDB — Gives the name of the local options DBM database to be used to resolve lookups for individual user options. The options for all users are stored in this one file, under username keys.

The database lookup feature is not used, if this option isn't specified. If it is, the local filter and SpamAssassin config files are ignored and only the DBM database is used instead. This option may be overridden by the "-c" command line option.

The database name given should not include the ".dir" and ".pag" extensions, since these are added gratuitously by DBM. The database itself can be built by the "ConfigEdit.cgi" program (see the chapter on "User Support") or directly by a program of your choice.

Note that the name used to look up the user options depends on the setting of the "-q" or "QualifyNames" flag.

DebugFile — Turn on debugging (regardless of whether a debug file name was compiled into the filter) and set the name of the file where the debugging information is written to the name supplied as the value of this parameter. This will override any compiled-in file name but, will in turn be overridden by the "-d" command line option.

Note that, if you expect error messages from options processing to be written a debug file, a debug file name must be complied into the filter or the "-d" command line option must be used and it must appear first on the command line. If that isn't done, errors from option processing are always written to the sendmail syslog file (wherever that is) anyway.

DebugLev[el] — 2|0|1|3|4 Set the debug level to 0 thru 4. Increasing the debug level turns on tracing of progressively more and more detailed debugging information. Zero turns it off. One traces high level work. Two traces basic work. Three traces low level work. Four

traces message transport. If you'd like your debug file to get filled up fast, pick level 3 or 4. The default is level 2. This option may be overridden by the "-d0" thru "-d4" command line options.

DomainIncl[udes] Auth|Local A comma separated list (no intervening whitespace is allowed) of one or more additional classes of users that are included in the local domain. All of the items in the list are orred together to give a cumulative list of additional user classes. At present, there are two acceptable values, those being "Auth" and "Local". The "-Da" command line option has the same effect as supplying the value "Auth" to this parameter, while the "-Dl" option is the same as "Local".

If "Auth" is chosen, any sender that validates to SMTP via an AUTH protocol (whichever one it is), will be treated as an internal user. This option may be useful to ISPs who have many external users who need to be treated as virtual internal users when they connect to SMTP.

If "Local" is chosen, any sender that connects to SMTP via the local IP address (127.0.0.1), will be treated as an internal user.

DomainList This option supplies the list of domain names that are used to determine whether mail is being delivered locally or not. Any domain names in this list are considered local, for purposes of the "-e" or FilterExt and "-i" or FilterInt options. This option may be overridden by the "-D" command line option.

The list may include one or more domain names, separated by commas. It may also include file names. Any name that begins with a '/', '~' or '.' is assumed to be a file name.

If a file name is given, the file should contain a list of local domain names, one per line. Blank lines and comments beginning with '#' are ignored. Typically, this feature would be used to point to the file "/etc/mail/local-host-names" or the same file where sendmail gets local domain name information from.

The domain name "localhost" is forced onto the list at the end, if it isn't specified. This name is always assumed to be a local domain name by sendmail.

Note that the filter attempts to automatically determine whether a recipeint's name is local or not, regardless of the contents of this list so using it is partly supplemental. While there is no harm in giving the filter a list of all local domain names (e.g. "/etc/mail/local-host-names"), it is not strictly necessary. However, if you have domains that are not local to this system but are still considered internal to your network, you might wish to include them in this list so that messages delivered to addresses in these domains will not get filtered, when sent from local users. Be aware, however, that some spammers spoof the same domain name for the sender as the recipient, thereby bypassing spam checking entirely, if the domain list includes the domains of local recipients.

It is much better, for the purposes of bypassing filtering from internal users, to include numeric IP addresses or address ranges within the domain list. If the domain list includes numeric IP addresses, the sender's IP address only (recipient IP addresses are not checked) will be compared against all of the IP addresses in the domain list. Any that match will be considered local senders.

Numeric IP addresses can be of the form "n.n.n.n" or "n.n.n.n/m". In the first case, the IP address of the sender must match the address given exactly. In the second case, the value "m" is a mask (range 0-32) that indicates how many high-order bits in the address are checked. For example, "192.168.1.0/24" will match all addresses in the range "192.168.1.0" to "192.168.1.255" while "192.168.0.0/16" will match all addresses in the range "192.168.0.0" to "192.168.255.255". The mask "/0" is equivalent to the mask "/32".

FilterExt[ernal] — Yes|No Turn on or off filtering of messages sent externally (i.e. messages from someone inside the domain to someone outside the domain). This option may be overridden by the "-e" command line option.

FilterInt[ernal] — Yes|No Turn on or off filtering of messages sent internally (i.e. messages from someone inside the domain to someone else also inside the domain). This option may be overridden by the "-i" command line option.

FilterTrans[it] — Yes|No Turn on or off filtering of messages transiting the system (i.e. messages from someone outside the domain to someone else outside the domain). The transit setting is typically used by ISPs who are delivering mail for many other systems on a mail handling gateway. This option may be overridden by the "-t" command line option.

IgnoreVirus[es] — Yes|No Turn on or off the filtering of any viruses found in messages for this user. If this flag is set, viruses will be ignored even if they are found. You asked for enough rope to hang yourself. Here it is! Be very careful about using this option in the global configuration file as it will turn off all virus filtering. Also, be aware that the "SumOptions" or "-os" flag has no effect on this flag. Rather, the setting of this flag, whether on or off, is applied to each user individually.

Note that, for efficiency sake, there is only one copy of any modified email message. This being the case, it is possible that a message containing a warning about a virus, which is sent as well to multiple users who wish to be warned about both viruses and spam, whould be sent to a recipient who has "IgnoreVirus" set to "Yes" and "IgnoreSpam" set to "No". This recipient will see the virus notification, despite the fact that they've requested not to. In this case, the wishes of the many outweigh the wishes of the few.

KeepAll — Yes|No When turned on (set to "Yes"), this option will keep all messages filtered, regardless of whether they are altered or not, in the corral. This option can be useful for debugging or for anyone needing a complete audit trail of all messages processed. The setting of this option may be overridden by the "-k" command line option.

Please bear in mind, when using this option, that only messages that are filtered can be kept. If a message is not filtered because it is bypassed (e.g. internal -> external), it will not be kept. Sorry, that's just the way it has to be. However, if you use the either the ArchiveDirectory or ArchiveProgram option to archive messages, they do not suffer from this restriction.

Language * — The name of the language configuration file used by the filter to generate message inserts is set by this option. The name given is appended to the language directory supplied by the "-L" or "LanguageDirectory" options and then the suffix ".cf" is appended. Note that case is important, since the name given is used exactly to compose the file name. For example, if the "-L" option is set to "/etc/mail" and the

user selects "Polish" via this parameter, the following file will be processed:

/etc/mail/Polish.cf

The file is opened and processed just like a regular local configuration file, except that the only parameters which are valid are those starting with "Msg" (from the Message Formatting Options section, below), that is to say all of the message insert texts. The idea is to allow a separate configuration file to be created for each language that is supported and for this file to allow the text of all of the message inserts in it to be written in a particular language. When the user selects a language, in their local configuration, the text of all of the message inserts used for them is loaded from the language configuration file.

This option may only be specified in the local configuration file or the DBM configuration database.

LanguageDir[ectory] Set the directory used for language support to the name given. This parameter is used in conjunction with the "Language" local configuration option (see that option for details) as well as the "X-Accept-Language" header in received messages. It is overridden by the "-L" command line option.

If a message is received that includes an "X-Accept-Language" header, the language value supplied in that header is used to set the name of the language configuration file used by the filter to generate message inserts. The language value found is appended to the language directory supplied by this option and then the suffix ".cf" is appended.

Typically, the language values used in "X-Accept-Language" headers are two character names, as specified by the mail/MIME RFCs. Note that, prior to composing the file name, the the language value is lowercased to insure consistency. The language value in the header is used exactly to compose the file name. For example, if this parameter is set to "/etc/mail" and the "X-Accept-Language" header contains "fr", the following file will be processed:

/etc/mail/fr.cf

The file is opened and processed just like a regular local configuration file, except that the only parameters which are valid are those starting with "Msg" (from the Message Formatting Options section, below), that is to say all of the message insert texts. The idea is to allow a separate configuration file to be created for each language that is supported and for this file to allow the text of all of the message inserts in it to be written in that language. When a message is received in a particular language, the text of all of the message inserts used is loaded from that language's configuration file.

Paranoid Yes|No Turn on or off filtering of messages from all senders. Normally, if this option isn't turned on, messages from trusted users such as root, mailer-daemon and other daemons, sent from the local machine are not filtered. The reason for this is because, filtering such messages can be annoying and there is often no need for them to be filtered. However, if you are running in an environment where nobody can be trusted (e.g. an ISP), you probably don't want to let any messages, even from normally-trusted users, pass without filtering.

Partition

24|1|2|3|4|6|8|12 Select the interval to partition the corral into. The number given is the number of hours to use for the partition interval. Acceptable values are those shown. If a value is not specified, the default is 24 hours. This value is overridden by the command line option "-pn", where "n" is the partition interval.

Partitioning allows the number of filtered messages, stored in the corral, to be dramatically increased. Normally, an unpartitioned corral will hold all of the messages filtered on a small to medium sized system. However, on some large systems (e.g. those used in production environments and by ISPs), the number of messages that must be stored in the corral can easily exceed reasonable limits for the file system and preclude notification and message handler programs from being able to examine and process corralled messages.

Partitioning circumvents this problem by splitting the corral into separate subdirectories, based on the time interval specified. If a value of 24 is chosen, for example, the corral will be split into chunks that each contain 24 hours worth of corralled messages. These smaller subdirectories can be more easily processed by notification and message handler programs.

All filtered messages are partitioned with this option. Spam, may also be partitioned post facto (e.g. if the corral fills up and you need to partition immediately as a quick fix to a space problem) by the separate notification program specific to it. A reasonable partition interval is 24 hours. Lesser intervals may be chosen if you still experience problems with the volume of messages in the corral.

ProxyUser[id]

A proxy userid to use in looking up mail disposition and spam processing options. Normally, the recipient's userid, stripped of domain information, is used to look up user-specific mail disposition options plus spam processing options in their home directory. However, if the recipient doesn't have a home directory and if a proxy userid is given, the options in the proxy userid's home directory will be used instead.

This option might be useful to ISPs, who process mail for many users but who do not have userids set up for all of them. Usually, mail delivery is governed by user-specific options found in a ".sendmailfilter" file in each recipient's home directory. However, if the recipient has no home directory, the ".sendmailfilter" file in the proxy userid home directory is used.

A similar situation arises with Spam Assassin, if spamd is used to arbitrate whether a message is spam. Spam Assassin looks in the recipient's home directory for options such as whitelists. If the user doesn't have a home directory, the home directory of the proxy userid is used instead.

This option may be overridden by the "-u" command line option.

Note that the values set for the "IgnoreSpam" and "IgnoreVirus" options in a proxy user's configuration have no effect on users who have no configuration. Such users will have their viruses and spam filtered by default and there is no way to avoid this, short of setting up configuration information for them. It was felt that virus and spam filtration was too important to get bypassed by accident.

QualifyNames

Yes|No Use or do not use fully qualified names (i.e. recipient name plus domain name) for any configuration database lookup of user options that is done in

conjunction with the "ConfigDB" option. This option may be overridden by the "-q" command line parameter.

Users of this option typically have virtual users who do not have any real username or home directory on the machine where mail is delivered, hence the need to store their configuration information in a common database. Since it is possible for the same username to exist in two domains (e.g. "custserv"), the domain information must be used to qualify the username and ensure that the name used to store configuration information is unique.

Note that spam saved in the spam corral is automatically saved using only the recipient's name (domain information is stripped) for local users and using the fully qualified name for all others. This is done so that all spam to a single local recipient, regardless of how it was addressed to them, will be corralled under a single name. The advantage of this behavior is that it results in the need to send only a single notification message to each spam recipient when telling them about the spam they've received. However, bear in mind that even using this scheme, alias names will have their spam stored as a separate user, regardless of who the alias resolves to.

ReplaceHTML * Yes|No Turn on or off filtering for all HTML tags, regardless of where they occur in a message. Only really harmless tags are left intact (e.g. <p>,
). The global option may be overridden by the "-h" command line option.

If this flag is on, the filter removes all HTML that could be nasty. This means looking at tag names plus individual parameters within the tags. The tables found in smfopts.h are used to do these checks. If this flag is off, only the tags that are really nasty (plus embedded comments in tag names) are removed. The really nasty tags are: iframe, object, script, applet and embed, since they can launch code. These tags (and anything in between them and their terminators) are always removed, if you ask for virus checking.

SendmailProto[col] The communications protocol to use in talking to sendmail. This must match the value given in the sendmail configuration file via the mail filter option. For example, if sendmail's m4 config file says:

INPUT_MAIL_FILTER(`filter1',`S=inet:2526@localhost, F=R')dnl

the value of SendmailProto should be:

SendmailProto inet:2526@localhost

According to the sendmail documentation, the possibilities are:

{unix|local}:/path/to/file *A named pipe.*
inet:port@{hostname|ip-address} *An IPV4 socket.*
inet6:port@{hostname|ip-address} *An IPV6 socket.*

One way or another, this option must be specified in order for the filter to operate. It must either be supplied in the global config file or via the "-p" command line switch. This parameter cannot be empty. The command line switch overrides the config file option.

SumOptions — Yes|No — Turn on or off summation of local user options. When turned on, this causes all of the local user options from the configurations of all of the local recipients of a message to be summed to arrive at one, single set of options to be used for processing the message. If supplied, the "-os" command line parameter overrides this option.

Normally, options from the configuration of the first or only local recipient of a message are used, in conjunction with the global options, to process each message (messages are only processed once, regardless of how many local recipients a message is bound for and non-local recipients are irrelevant, since some other system will deliver the message to them). If this parameter is set to "Yes", the configurations of all of the recipients of a message are read and then all of their options are summed up to create a compound set of options that represent all of their preferences. This compound set of options is then used to process the message (it is still only processed once).

Option summation takes place as follows: for switches ("AutoRemail", "ReplaceHTML", "SpamAlways" and "SpamFastPath"), the "No" setting overrides the "Yes" setting; for the "UnknownDispo" option, the "MIME" setting overrides the "Ignore" setting, while the "Warn" setting overrides both; for the "SpamDel" option, the "trash" mode is overridden by the "deliver" mode which is overridden by the "corral" mode; for the "SpamLevel" option, the highest level is chosen; for the "StatXxxxRatio" options, the highest threshold found is used and for the "StatXxxxBoost" options, the lowest boost value found is used; for the "AddXTags" option, the "Yes" setting overrides the "No" setting for each tag ("envelope", "version") that is set in any configuration processed; and only the first recipient's settings are used for the "Language", "Msg*" and "SpamProto" options.

UnknownDispos[ition] * W[arn]|M[IME]|I[gnore] — Chooses the disposition for unknown file types, found as attachments to a message. The global option only may be overridden by the "-uw", "-um" and "-ui" command line options.

A disposition of "Warn" will cause a warning to be inserted into the message whenever an attachment is found with an unknown file type (i.e. it has no extension or an extension that is not known). This is the default setting, since it is quite possible that a file type which the filter doesn't know about may be harmful and the assumption is made that it is best to warn about these things.

The "MIME" disposition will cause the filter to take additional steps to determine the file type, before issuing a warning. It will look at the MIME type for the attachment and, if it is one of the acceptable types, no warning will be issued. Otherwise, a warning will be issued, as above, for the "Warn" disposition.

A disposition of "Ignore" will cause attachments with unknown file types to be ignored and treated as if they were perfectly OK. No warning will be issued for any unknown file types. This setting is more than somewhat dangerous, since it is quite possible that a file type which the filter doesn't know about may be harmful. Not receiving any warning could allow a message recipient to inadvertently open a harmful attachment.

3.4 Archiver Support Options

The archiver support options are built into the filter to provide support for email archiving programs that are used to archive a copy of some or all messages that are passed through a system, usually in a database or file store. Messages may be directed to the archiver via the filter or they may be sent by an email system (such as Scalix) directly to the archiver. If the later option is used, the archiver support options can be used to deal with unwanted bouncebacks in the event that the archiver is not available.

Only the global options configuration file can contain the archiver support options. Here is the list of archiver support options available in the global configuration file:

ArchiveDir[ectory]

Using this parameter in the global configuration file will cause the filter to store a duplicate copy of all messages that it processes, in the directory named in this parameter. Each message is given a name that is based on the sender's name, plus an additional uniqueness key that renders collisions highly improbable. The archive directory is partitioned automatically, once a day, to allow it to be more easily managed and prevent the archive directory from growing too big for the file system to manage. Should you wish to split the archive more often, see the "ArchivePartition" parameter for more information about how to do it.

Messages in the archive directory are stored exactly the way they are delivered to sendmail. In all probability, the lawyers will want to be pouring over each one with a fine tooth comb so it is important that each message appear exactly as sent. Thus, each archive file is a flat text file that contains the entire email message: headers; straight text; and MIME encoding. Furthermore all messages can be archived, including those that bypass normal filtering when checks such as the internal/external/transit checks controlled by the "FilterExternal", "FilterInternal" and "FilterTransit" options indicate that a message should not be filtered.

Note that this parameter is mutually exclusive with the "ArchiveProgram" parameter. Whichever one comes last in the config file wins.

ArchivePartition

24|1|2|3|4|6|8|12 Select the interval to partition the archive directory into. The number given is the number of hours to use for the partition interval. Acceptable values are those shown. If a value is not specified, the default is 24 hours.

Partitioning allows the number of messages, stored in the archive directory, to be dramatically increased. Normally, an unpartitioned archive directory will hold all of the messages archived on a small to medium sized system. However, on some large systems (e.g. those used in production environments and by ISPs), the number of messages that must be stored in the archive directory can easily exceed reasonable limits for the file system and preclude additional messages from being archived.

Partitioning circumvents this problem by splitting the archive directory into separate subdirectories, based on the time interval specified. If a value of 24 is chosen, for example, the archive directory will be split into chunks that each contain 24 hours worth of archived messages. These smaller subdirectories can be more easily processed by any of the programs in the archiver suite.

All archived messages are partitioned with this option. A reasonable partition interval is 24 hours. Lesser intervals may be chosen if you still experience problems with the

volume of messages in the archive directory. If you are looking for a particular archived message, the partitioned subdirectories are named based on the partition interval time. The subdirectory name is simply the time of the beginning of each interval, in seconds from the epoch, expressed as an 8-digit hexadecimal number.

ArchiveProg[ram] Using this parameter in the global configuration file will cause the filter to send a duplicate copy of all messages that it processes, via regular email, to the archive program whose address is given. Several MTAs (e.g. Scalix) employ this method to deliver messages for archiving to an archive program. Consequently, many of the email archive programs are capable of accepting messages to be archived as incoming mail. You will need to consult your archiver's documentation about how to set this up.

On the MailCorral side, one simply assigns to this parameter the email address of the archive program and, optionally, enumerates the recipient addresses of those whose mail is to be archived, using the "archive_to" parameter.

One word of caution, though. Recall that the duplicate copy of the message, bound for the archiver, is delivered via regular email channels. This means that it makes a second pass through the filter. In order that the duplicate message itself is not also duplicated and archived, and so as to avoid processing (and possibly altering) the duplicate message a second time, the filter looks for all messages bound for the archiver's email address and bypasses them.

There are two points to note, then. The first is that all messages sent to the archiver's email are bypassed, not just those that MailCorral duplicates and forwards to it. So, if someone externally sends a message to the archiver's email address, it goes right through, untouched by the filter. This could be construed as a feature because it allows outsiders to also send messages directly to the archiver for archiving. However, on the flip side of the coin, it could instead be construed as a less than ideal way of operating. But, it was felt that all messages bound for the archiver's email address were probably going to be sent, by the archive program, straight through to the archive repository anyway (i.e. it wouldn't be trying to actually read them) so there was no disadvantage to giving them all a free pass.

The second point is that the address that you supply to this parameter must match those that sendmail will ultimately make up as the duplicate message's delivery address. In general, if the sendmail configuration file has domain name masquerading turned on (the "DM" option in the sendmail.cf file or the "MASQUERADE_AS" option in the sendmail.mc file) and you are archiving to a local recipient, the address given can either be the full name (i.e. username, at sign, domain name) or it can simply be the username, all by itself. If you have some other, clever way of getting stuff to local users, don't use it here. Otherwise, for recipients on other systems, you must use their correct username, an at sign and their full domain name (there's a good chance your mail won't get delivered properly, if you don't do this, so it shouldn't be a big deal).

Failure to use an address for this parameter, that sendmail and the filter can identify will result in a message sending loop, whenever a message is sent to the archive. Fortunately, sendmail is smart enough quit the loop before it loops forever and fills the spool will billions and billions of messages, but it will deliver about 24 copies of

each one before it gives up for taking too many hops. It is advisable to test the filter carefully, after setting this parameter, in a non-production environment with a single test message. You should be able to send a message directly to the archiver's email address and have it not be filtered. Then, you should be able to send a message to any user (whose email is being archived) and have a duplicate show up in the archiver's mailbox. You can easily test this after you've set up the archiver's mailbox but before you tell the archive program to catch all mail delivered to its mailbox. This will allow you to use the regular mail command to see what is being delivered before everything becomes automagic.

Duplicated messages are sent to the archiver exactly the way they are delivered to sendmail. In all probability, the lawyers will want to be pouring over each one with a fine tooth comb so it is important that each message appear exactly as sent. Thus, each duplicated message contains the entire original email message: headers; straight text; and MIME encoding. Furthermore all messages can be archived, including those that bypass normal filtering when checks such as the internal/external/transit checks controlled by the "FilterExternal", "FilterInternal" and "FilterTransit" options indicate that a message should not be filtered.

Note that archiving messages using this option is somewhat of a resource pig. The entire message is buffered in memory before it is sent to the archiver. Not the least of which, one reason for doing it this way is to allow us to send everything to sendmail at once, without having to worry about sendmail/milter errors leaving the pipe hanging. It also obviates the need to flush the pipe or try to retrieve half-sent messages if an error that we catch ourselves occurs. It was felt that this reliability was important and, in this day and age and, memory is so cheap and so abundant that, even with 10MB email messages, the performance and reliability benefits were well worth it.

Also note that this parameter is mutually exclusive with the "ArchiveDirectory parameter". Whichever one comes last in the config file wins.

IgnoreBounces

If this parameter is present in the global configuration file, the filter will look for bounceback messages from sendmail and/or any other MTAs that send them through this system. Bounceback messages are sent by an MTA when it tries to deliver a message but, for some reason, the recipient is unavailable. The MTA sends a new email message back to the original sender informing them of the problems encountered while trying to deliver their message.

This option is included mainly to support certain email archiving schemes which send messages to be archived to an archive program through regular email delivery channels (e.g. Scalix uses this scheme). The MTA that is archiving its messages makes a copy of each message, alters the to address to point to the archive program and injects the message into the regular email delivery stream. All well and good, providing the archive program stays alive and makes itself available. However, should it become unavailable (e.g. through a network failure), the regular email delivery stream necessitates that a bounceback message be generated. Since the from address of the duplicated message was never altered, the bounceback goes back to the original sender of the message. This probably comes as a huge surprise to them.

When a bounceback message is identified (bouncebacks, by convention all come from a common source which makes it possible to identify them), the list comprised of all the address enumerated by IgnoreBounces options, will be consulted. If the recipient of the bounced message, as determined by the address mentioned in the delivery report of the bounceback message, matches this option's value, it will be cancelled and never delivered. The option value may contain file name globbing style patterns. For example: "*@archiver.{com|org|net}". It may be repeated as many times as necessary to list all of the addresses to be bounced.

archive_to — Normally, by default, all messages passing through the filter are archived, if archiving is turned on. Using this option, it is possible to fine tune whose messages are actually archived. If this option is present, only those addresses explicitly mentioned in one or more "archive_to" parameters will be archived. All others will not.

If the recipient of a message matches this option's value, their messages will be added to the archive. Otherwise, their messages will be delivered in the normal manner but not archived. The option value may contain file name globbing style patterns. For example: "archiveme@*.*".

3.5 Spam Processing Options

Spam processing options are employed by the internal spam identifier that is invoked when the spam arbitron is chosen to be "internal" (see "-s" or "SpamProto") or an external spam arbitron (such as spamd) is chosen and the fast path is not turned off (see the "-ss" option or "SpamFastPath"). The "SpamRule" option specifies in a logical expression how the results of the various arbitrons are interpreted, if the default rule is not sufficient. The spam delivery options are applicable to all messages recognized as spam, regardless of how recognition is accomplished.

Either of the configuration files can contain white lists, black lists and/or spam delivery options. By properly tuning the global options file with black list information about the domains from which you constantly receive spam, you can build an effective spam blocking filter at very little runtime cost. By having your users update their local options files with black or white list information, they can further tune the behavior of the built-in spam identifier. Here are the spam processing options available in the configuration files:

IgnoreSpam * — Yes|No Turn on or off the filtering of any spam found in messages for this user. If this flag is set, spam will be ignored even if it is found. Be aware that the "SumOptions" or "-os" flag has no effect on this flag. Rather, the setting of this flag, whether on or off, is applied to each user individually.

Note that, for efficiency sake, there is only one copy of any modified email message. This being the case, it is possible that a message containing a warning about spam, which is sent as well to multiple users who wish to be warned about both viruses and spam, whould be sent to a recipient who has "IgnoreSpam" set to "Yes" and "IgnoreVirus" set to "No". This recipient will see the spam notification, despite the fact that they've requested not to. In this case, the wishes of the many outweigh the wishes of the few.

SpamAlways * — Yes|No Always add the spam report to mail, regardless of whether it is spam or not. The global option may be overridden by the "-sa" command line option.

SpamDel[ivery] * — C[orral]|D[eliver]|T[rash] Set the spam delivery mode to one of the three choices shown. The global option only may be overridden by the "-sc", "-sd" and "-st" command line options.

The "Corral" mode will cause any spam received to be sidelined in the spam corral, where it can be processed by a spam notification program and/or released for delivery by the recipient at a later date.

The "Deliver" mode will cause any spam received to be marked as such and then delivered to the recipient without further delay.

The "Trash" mode will cause any spam received to be tossed directly in the trash can, straight away. I like it! Unfortunately, the default is "Corral"

SpamFast[Path] * — No|Yes Turn off or on the spam fast path check, prior to calling the designated spam arbitron. The statistical spam fast check is turned on or off by this flag, too. If the designated arbitron is "internal" this flag has no effect, since it is possible to turn off the use of the internal check as the designated arbitron by not specifying it at all. The global option only may be overridden by the "-ss" command line option.

SpamLevel * — 1-3 Set the level of spam reporting to the number chosen (a value of 1 through 3). The global option only may be overridden by the "-s1" thru "-s3" command line options.

Level one (the default) causes a single header line to be inserted in any message that is found to contain spam, giving the spam processing statistics as three percentage values.

Level two causes the same header line described under level one to be inserted in any message that is found to contain spam. It also inserts any headers generated by the spam arbitron (selected by "SpamProto") into the message.

Level three does everything that levels one and two do, plus it formats any report generated by the spam arbitron for insertion into the message body as a paragraph of text.

SpamProto[col] * — none|internal|spamd The communications protocol and port to use in talking to a spam arbitration daemon. The protocol must be one that this filter knows about and the port must match the one that the daemon will be listening on.

Currently, the supported protocols are:

internal - The internal, fast path spam checker.
spamd - The Spam Assassin daemon.

Except for the "internal" protocol, the port number and host name or IP address must follow the protocol name, separated by a colon and an at sign. So, the entire parameter should look like one of the following:

internal

or

proto:port@host

For example, if you are running spamd and it is listening on port 2527 on the local machine, the value for SpamProto should be:

SpamProto spamd:2527@localhost

By supplying a valid protocol and port, you will cause all messages received to be sent to the spam arbitration daemon. If it thinks the message is spam, it will be treated as such. If you don't supply this parameter, only the internal spam checks will be performed (blacklists/whitelists and statistical).

Note that when **any** spam arbitron is used, the internal, fast path spam checker is always run (unless the "-ss" option is set or "SpamFastPath" is set to "No"), prior to calling the arbitron selected, in an attempt to speed up spam checking by bypassing the overhead involved with calling the arbitron.

An optional timeout value may be supplied for any of the arbitrons chosen. If this parameter is supplied, it must follow the protocol, port number and host name immediately and be enclosed in parenthesis. For example:

SpamProto spamd:2527@localhost(20)

The value given is the time, in seconds that MailCorral is prepared to wait for the entire transaction with the spam arbitron. Essentially, this is the total time for the arbitron to respond to the spam determination request -- it includes the elapsed time for all of the reads and writes to the arbitron. In the above example, the time allotted to SpamAssassin would be twenty seconds.

The default timeout is set to 30 seconds, if no value is supplied. You may pick any positive value but consider this. Sendmail is only prepared to wait for an answer from the filter for so long. If the arbitron timeout is to be of any use, it should be set to some value smaller than the timeout given to sendmail in its configuration file (no value in the sendmail configuration file means a default timeout of 10 seconds).

A good choice would be to give the arbitron 60-80% of the timeout allotted to the filter in the sendmail configuration file. This will ensure that the filter can still complete the job of filtering a message in the time allowed, despite the fact that the arbitron times out while making its decision. And, if a virus arbitron is also being used (see the "-v" option or "VirusProto"), the time allocated to both the arbitrons should be split between them, as you see fit. In that case a good choice would be to give 30-40% to each of the two arbitrons.

Note that a time limit may be supplied to the internal arbitron but there isn't much reason for this, since it executes extremely quickly. When calculating how much time to allow the filter and its arbitrons, you can neglect the time consumed by the internal arbitron.

An empty parameter value may be specified in the local (user) options file to turn off a spam arbitron that was turned on globally. In the global options file, this parameter cannot be empty. The global option only can be overridden by the "-s" command line parameter.

SpamReplies 0-25 If spam filtering is done, the number of replies that should be sent to a spammer (per day), telling them that what they are sending is spam. Once the threshold is reached for a particular spammer in a single day, all subsequent messages from them are just dumped with no reply. The default is 0 (i.e. always just dump all spam). The maximum is 25. May be overridden by the "-r" command line option.

This option is necessary because some spammers use autoresponders to reply to any mail sent to them. This may well appear as spam too, in which case, a reply will be sent to it. Do you see the potential for ping pong? I do.

SpamRule * expression An expression that specifies the rule used to decide whether a message is considered spam or not. If the expression evaluates to true, the message is spam. If it evaluates to false, it is not spam.

Normally, this expression is not specified and the system makes up a rule of its own, depending on the values chosen for "SpamFastPath" and "SpamProtocol". For example, the system typically uses:

(SpamFast != 1) && ((SpamFast > 1) || (SpamStats >= 100) || (Spamd >= 100))

You may specify your own rule, if you wish to change the order in which the spam checkers are called, alter the weighting of the various spam checkers or invoke a programmable spam arbitron of your own choosing (see ProgArbCode). And, in the user configuration file, you may specify this option without any rule to turn off any global spam rule and revert back to the system's default rule.

The expression supplied should be a logical expression that evaluates to either true (message is spam) or false (message isn't spam). If it contains spaces for readability, it should be enclosed in double quotes.

There are a number of predefined variables and constants that you can use to build the rule. These are:

Spamd The result from calling spamd (typically SpamAssassin). The results are normalized so that any number greater than or equal to 100 is usually considered spam. Using this variable causes spamd to be called.

SpamFast The result from calling the spam fast check. The results are: 0 for OK; 1 for deliver the message (i.e. it is whitelisted); 2 for local options have determined the message is spam (i.e. the sender is in the user's blacklist); 3 for global options have determined the message is spam (i.e. the sender is in the global blacklist). Using this variable causes the spam fast check to be called.

SpamStats The result from calling the statistical spam check. The results are normalized so that any number greater than or equal to 100 is usually considered spam. Using this variable causes the statistical spam check to be called.

SPAM_OK

The OK return code from any of the programmable spam arbitrons (depends on the variable used). May be used like this:

MyArb == SPAM_OK

SPAM_FOUND The spam found return code from any of the programmable spam arbitrons (depends on the variable used). May be used like this:

MyArb == SPAM_FOUND

Any variable that is not in the list above causes the programmable arbitron code supplied by you (see ProgArbCode) to be invoked and passed one or more components of the message (see ProgArbRequires). The result from the programmable arbitron is set into the variable, once the arbitron returns. Programmable arbitron results can be one of: SPAM_OK or SPAM_FOUND. You might include your own programmable arbitrons in the spam rule like this:

```
(SpamFast != 1) && ((SpamFast > 1) || (SpamStats >= 100)
|| (Spamd >= 100) || (MySpam1 == SPAM_FOUND)
|| (MySpam2 == SPAM_FOUND))
```

Note that the spam rule is evaluated in the usual manner with the first term that renders the expression unequivocally true or false causing it to exit without calling any of the other arbitrons. Thus, you can use the order of evaluation to optimize your spam checking (e.g. in the above expression, the programmable arbitrons are only invoked if the two internal spam checks and spamd decide the message isn't spam).

StatEmbedRatio * 50, 0-1000 If statistical spam filtering is carried out by the internal fast path spam filter, this parameter sets the weighting and threshold for the ratio of embedded comments or escape sequences to words in the message, for the message to be considered spam. The value is in parts per thousand. The message's embedded ratio is multiplied by 100 and divided by the value given. This will cause any value above the threshold value to yield a spam value of 100%. A value less than the threshold may still lead to a determination of spam, since all of the spam values are summed. A value of zero disables this spam check.

Since any occurrence of embedded comments or escape sequences in words is an excellent predictor of spam, the default threshold is set fairly low at 50 embeds per 1000 words.

StatHREFRatio * 200, 0-1000 This parameter sets the weighting and threshold for the ratio of HREFs to words in the message, for the message to be considered spam, if statistical spam filtering is carried out by the internal fast path spam filter. The value is in parts per thousand. The message's HREF ratio is multiplied by 100 and divided by the value given. This will cause any value above the threshold value to yield a spam value of 100%. A value less than the threshold may still lead to a determination of spam, since all of the spam values are summed. A value of zero disables this spam check.

A large number of HREFs (links to Web pages) in a message is a good indication that a message may be spam, since spammers often link their message to a Web site with the actual content or to tracking pages that update their database when spam is read. Keep in mind, these are actual references to Web pages in HTML tags, not references to them in the text of a message. Hence, someone sending a message that says,

"Here's the URL you wanted ...", should not trigger this test. The default setting is a moderate 200 HREFs per 1000 words to allow a generous number of links for legitimate reasons but still contribute to the spam score if a message includes links.

StatImageRatio * 100, 0-1000 If statistical spam filtering is carried out by the internal fast path spam filter, this parameter sets the weighting and threshold for the ratio of images to words in the message, for the message to be considered spam. The value is in parts per thousand. The message's image ratio is multiplied by 100 and divided by the value given. This will cause any value above the threshold value to yield a spam value of 100%. A value less than the threshold may still lead to a determination of spam, since all of the spam values are summed. A value of zero disables this spam check.

Since occurrences of images in messages are often a good predictor of spam, the default threshold is set fairly low at 100 images per 1000 words. A special case is recognized where a message contains no words but just images. This is assumed to be image spam and is given a score of 100%. However, recipients of certain kinds of newsletters that are composed entirely of images or a high proportion of images may wish to disable this test or set a higher threshold.

StatTableRatio * 250, 0-1000 This parameter sets the weighting and threshold for the ratio of tables to words in the message, for the message to be considered spam, if statistical spam filtering is carried out by the internal fast path spam filter. The message's table ratio is multiplied by 100 and divided by the value given, which is in parts per thousand. This will cause any value above the threshold value to yield a spam value of 100%. A value less than the threshold may still lead to a determination of spam, since all of the spam values are summed. A value of zero disables this spam check.

Spammers often use large numbers of table tags in a message to format their message for visual effect (it is, after all, advertising) or even to hide the text of the message from a content-based identifier. Tables do have legitimate reasons to be in a message, however, so a moderate threshold of 250 tables per 1000 words is the default.

StatImageParmBoost * 10, 0-100 If statistical spam filtering is carried out by the internal fast path spam filter, this parameter gives the spam score boost value (in percent) for any images that have parameters. For each image with parameters, the boost value is added directly to the final spam score. A value of zero disables this spam check.

Although there are legitimate reasons to use parameters in an image, most images are usually fixed URLs. Spammers frequently include an empty or innocuous image in a message that includes parameters telling them who the recipient of the message is. If the message is read and the image fetched from their Web server, the spammer can update their database to keep track of all of the actual recipients of their spam. The default boost is a moderate 10% for each image with parameters.

StatLinkEmailBoost * 50, 0-100 This parameter gives the spam score boost value (in percent) for any link (in images or HREFs) that contain an email address, If statistical spam filtering is carried out by the internal fast path spam filter. For each link containing an email address, the boost value is added directly to the final spam score. A value of zero disables this spam check. It is important to note that the boost is not given to HREFs containing email addresses as the result of a "mailto:" parameter.

Email addresses in links or especially in images are used by spammers as a feedback mechanism to identify recipients of spam. If the message is read and the image

fetched from their Web server, the spammer can update their database to keep track of all of the actual recipients of their spam. Since there is very little real reason for including an email address as a parameter in an image or link, the default boost is a high 50% for each image or link with an email address.

StatTextBase64Boost * 80, 0-100 If statistical spam filtering is carried out by the internal fast path spam filter, this parameter gives the spam score boost value (in percent) for any text or HTML MIME entities that are encoded Base64. For each text/HTML MIME entity that is so encoded, the boost value is added directly to the final spam score. A value of zero disables this spam check.

Perhaps there are legitimate reasons to encode plain text and HTML as if it were binary data but the most common use of this technique is to obscure spam and viruses from detection by scanners. This being the case, the default boost is an extremely high 80% for each text/HTML MIME entity that is so encoded.

all_spam_to * If the recipient of a message matches this option's value, they will always receive every message sent, regardless of whether it is spam or not. The option value may contain file name globbing style patterns. For example: "wantspam@*.*". It may be repeated as many times as necessary to list all of the addresses that should always have mail delivered to them, although there is no point to use it more than once in the local options file, since it only applies therein to the local user.

blacklist_from * If the sender of a message matches this option's value, it will be treated as spam. The option value may contain file name globbing style patterns. For example: "*@spammers.{com|org|net}". It may be repeated as many times as necessary to list all of the addresses to be blacklisted.

whitelist_from * If the sender of a message matches this option's value, it will never be treated as spam but will always be delivered instead. The option value may contain file name globbing style patterns. For example: "*@goodguys.{com|org|net}". It may be repeated as many times as necessary to list all of the addresses to be whitelisted.

Probably the best use for this option is in a user's local options file where it can override one of the global black list entries to allow mail from a user's favorite spammer to be delivered, despite the spammer's domain being on the global blacklist.

In addition to the sendmail filter's own configuration files, the filter will also process configuration files for certain spam arbitration daemons to support fast path detection of spam using the black and white lists of the daemon.

If the spam arbitron chosen is spamd, the local and global Spam Assassin configuration files are read to extract white and black list information. This information is then acted upon to determine if a message is spam, hopefully without invoking the arbitron. The Spam Assassin configuration files processed (and the order in which they are read) are:

~/.spamassassin/user_prefs	*Recipient local preferences.*
/usr/local/etc/spamassassin/local.cf /usr/pkg/etc/spamassassin/local.cf /usr/etc/spamassassin/local.cf /etc/mail/spamassassin/local.cf	*Installation overrides, set up by sysadmin.*

/etc/spamassassin/local.cf

/usr/local/share/spamassassin/60_whitelist.cf
/usr/share/spamassassin/60_whitelist.cf

System whitelist info, shipped with the product.

3.6 Virus Processing Options

The virus processing options determine whether the internal virus checker is to be invoked (when the virus arbitron is chosen to be "internal" via the "-v" or "VirusProto") or an external virus arbitron (such as clamd) is to be invoked. They also determine whether the internal virus checker is turned off when an external virus arbitron is used or run in addition to it (see the "-vv" option or "VirusChecker"). The "VirusRule" option specifies in a logical expression how the results of the various arbitrons are interpreted, if the default rule is not sufficient. Here are the virus processing options available in the configuration files:

VirusAlways * — Yes|No Always add the virus report to mail, regardless of whether it contains spam or not. The global option may be overridden by the "-va" command line option.

VirusCheck[er] * — No|Yes Turn off or on the internal virus checker, prior to calling the designated virus arbitron. If the designated arbitron is "internal" this flag has no effect, since it is possible to turn off the use of the internal checker as the designated arbitron by not specifying it at all. The global option only may be overridden by the "-vv" command line option.

VirusProto[col] * — none|clamd|internal The communications protocol and port to use in talking to a virus arbitration daemon. The protocol must be one that this filter knows about and the port must match the one that the daemon will be listening on.

Currently, the supported protocols are:

clamd - The ClamAV daemon.
internal - The internal virus checker.

Except for the "internal" protocol, the port number and host name or IP address must follow the protocol name, separated by a colon and an at sign. So, the entire parameter should look like one of the following:

internal

or

proto:port@host

For example, if you are running clamd and it is listening on port 2528 on the local machine, the value for VirusProto should be:

VirusProto clamd:2528@localhost

By supplying a valid protocol and port, you will cause all messages received to be sent to the virus arbitration daemon. If it thinks the message contains a virus, it will be treated as such. If you don't supply this parameter, only the internal virus checks will be performed (MIME type/attachment type matching).

Note that when **any** virus arbitron is used, the internal virus checker is always run (unless the "-vv" option is set or "VirusChecker" is set to "No"), prior to calling the arbitron selected, but its results will not necessarily cause the virus arbitron to be bypassed. Only in the case of MIME entities or attachments that the internal virus checker marks for deletion will the external arbitron be bypassed, and such instances are admittedly quite rare.

An optional timeout value may be supplied for any of the arbitrons chosen. If this parameter is supplied, it must follow the protocol, port number and host name immediately and be enclosed in parenthesis. For example:

VirusProto clamd:2528@localhost(20)

The value given is the time, in seconds that MailCorral is prepared to wait for the entire transaction with the virus arbitron. Essentially, this is the total time for the arbitron to respond to the virus determination request -- it includes the elapsed time for all of the reads and writes to the arbitron. In the above example, the time allotted to ClamAV would be twenty seconds.

The default timeout is set to 30 seconds, if no value is supplied. You may pick any positive value but consider this. Sendmail is only prepared to wait for an answer from the filter for so long. If the arbitron timeout is to be of any use, it should be set to some value smaller than the timeout given to sendmail in its configuration file (no value in the sendmail configuration file means a default timeout of 10 seconds).

A good choice would be to give the arbitron 60-80% of the timeout allotted to the filter in the sendmail configuration file. This will ensure that the filter can still complete the job of filtering a message in the time allowed, despite the fact that the arbitron times out while making its decision. And, if a spam arbitron is also being used (see the "-s" option or "SpamProto"), the time allocated to both the arbitrons should be split between them, as you see fit. In that case a good choice would be to give 30-40% to each of the two arbitrons.

Note that a time limit may be supplied to the internal arbitron but there isn't much reason for this, since it executes extremely quickly. When calculating how much time to allow the filter and its arbitrons, you can neglect the time consumed by the internal arbitron.

An empty parameter value may be specified in the local (user) options file to turn off a virus arbitron that was turned on globally. In the global options file, this parameter cannot be empty. The global option only can be overridden by the "-v" command line parameter.

VirusRule * expression An expression that specifies the rule used to decide whether any of the components of a message contain a virus or not. The expression is evaluated once for each message component that is tested for viruses. If the expression evaluates to true, that component is deemed to contain a virus. If it evaluates to false, the component is deemed not to contain a virus. A virus free message only results when all tested components do not contain any viruses.

Normally, this expression is not specified and the system makes up a rule of its own, depending on the values chosen for "VirusChecker" and "VirusProtocol". For example, the system typically uses:

(VirusCheck == DEL_DELETE) || Clamd

You may specify your own rule, if you wish to change the order in which the virus checkers are called, or invoke a programmable virus arbitron of your own choosing (see ProgArbCode). And, in the user configuration file, you may specify this option without any rule to turn off any global virus rule and revert back to the system's default rule.

The expression supplied should be a logical expression that evaluates to either true (message component contains a virus) or false (message component doesn't contain a virus). If it contains spaces for readability, it should be enclosed in double quotes.

There are a number of predefined variables and constants that you can use to build the rule. These are:

Clamd — The result from calling clamd (typically ClamAV). For each message component analyzed, clamd will return a value of true, if a virus was found within the component. Using this variable causes clamd to be called.

VirusCheck — The result from calling the internal virus checker. The result is one of the "DEL_" codes described below. Normally, one would only want to mark the message component as containing a virus if the "DEL_DELETE" code were returned by the internal virus checker. The internal virus checker results are always generated in the normal course of filtering the message so there is no performance penalty associated with using this variable.

DEL_OK — The OK return code from the internal virus checker or any of the programmable virus arbitrons (depends on the variable used). May be used like this:

MyArb == DEL_OK

DEL_WARN — The warning return code from the internal virus checker or any of the programmable virus arbitrons (depends on the variable used). Reject means that the message component should be kept as part of the message but a warning should be issued to tell the user to be careful when oppening it. May be used like this:

MyArb == DEL_WARN

DEL_REJECT — The reject return code from the internal virus checker or any of the programmable virus arbitrons (depends on the variable used). Reject means that the message component should be kept as part of the message but its determinant (e.g. file name extension) should be mangled so that it can't be accidentally opened by some

well-meaning program. May be used like this:

MyArb == DEL_REJECT

DEL_DELETE The virus found return code from the internal virus checker or any of the programmable virus arbitrons (depends on the variable used). May be used like this:

MyArb == DEL_DELETE

Any variable that is not in the list above causes the programmable arbitron code supplied by you (see ProgArbCode) to be invoked and passed one or more components of the message (see ProgArbRequires). The result from the programmable arbitron is set into the variable, once the arbitron returns. Programmable arbitron results can be one of: DEL_OK or DEL_DELETE. You might include your own programmable arbitrons in the virus rule like this:

```
(VirusCheck == DEL_DELETE) || Clamd
|| (MySpam1 == DEL_DELETE)
|| (MySpam2 == DEL_DELETE))
```

Note that the virus rule is evaluated in the usual manner with the first term that renders the expression unequivocally true or false causing it to exit without calling any of the other arbitrons. Thus, you can use the order of evaluation to optimize your virus checking (e.g. in the above expression, the programmable arbitrons are only invoked if the internal virus checker and clamd decide that the message component doesn't contain a virus).

3.7 Programmable Arbitron Options

The filter contains a number of built-in message arbitration routines. It also knows how to call several well-known arbitration programs. These routines and programs (called arbitrons herein) are used to determine the status of a message (i.e. whether it is spam and/or whether it contains a virus).

To provide the flexibility of using other arbitrons (even including homemade ones) the filter supports the concept of programmable arbitrons. Using the SpamRule or VirusRule expression, the order of arbitron invocation and the handling of return values can be defined programmatically in the configuration files (either global or user). The variables used in these rules indicate which arbitrons are to be invoked. If the standard, system-defined variable names are used in a rule, the standard, system-defined arbitrons are invoked. If any non-standard variable names are used, the programmable arbitrons defined by these options are used.

The programmable arbitron options (except for ProgArbLog) may only be set in the global options configuration file. This is because these options supply code that is actually executed by the filter so, aside from the obvious security repercussions, do you really want every Tom, Dick and Harry being able to supply code to the filter that it can execute (I thought not)? However, once a programmable arbitron and its corresponding variable is made available via these global options, any user can include it in their local rules to cause that arbitron to process their messages. Here is the list of programmable arbitron options available in the configuration files:

ProgArbCode filename[(timeout)] The filename given must be the name of a file that contains a chunk of Perl code which will be executed whenever a programmable arbitron is to be invoked (see the Programmable Arbitrons Chapter). This code can be passed one or

more sections of the message being filtered (depending on the options set by ProgArbRequires) and must make a go or no-go decision on whether the message is spam and/or contains a virus.

Note that the filename can only be set in the global configuration file. However, this option can be specified without any filename in the user configuration file to turn off programmable arbitron processing for that user only.

An optional timeout value may be supplied in parenthesis to give the amount of time, in seconds, that MailCorral should be prepared to wait for the each complete transaction with any one of the programmable arbitrons. Essentially, this is the total time for the arbitron to respond to the arbitration request -- it includes the elapsed time for all of the reads and writes to the arbitron.

The default timeout is set to 30 seconds, if no value is supplied. You may pick any positive value but consider this. Sendmail is only prepared to wait for an answer from the filter for so long. If the arbitron timeout is to be of any use, it should be set to some value smaller than the timeout given to sendmail in its configuration file (no value in the sendmail configuration file means a default timeout of 10 seconds).

A good choice would be to give the programmable arbitron 10-20% of the timeout alotted to the filter in the sendmail configuration file. This will ensure that the built-in and well-known arbitrons that the filter typically calls will have time to do their jobs and still allow the filter to complete the job of filtering a message in the time allowed.

ProgArbLog * [filename] Since programmable arbitron code will indubitably require debugging, this option provides a method whereby the user can specify a log file which will be managed by MailCorral on behalf of the programmable arbitrons. The file will be opened whenever a programmable arbitron is called and a handle to it made available. The user can write trace information to this file in order to debug their programmable arbitrons.

Note that this option, without a filename supplied, in the user's configuration file, will turn off logging of programmable arbitrons for that user only.

ProgArbReq[uires] flag[, flag[, ...]] This option specifies one or more flags that indicate which sections of the email message the programmable arbitrons require to be passed to them. The sections indicated are passed to all of the programmable arbitrons and the mechanism used for passing them is to write each section to a file so think carefully about which ones you really need. Here are the acceptable flags:

SpamHeaders	The message headers, preened to make spam processing easier.
SpamBody	All text portions (including text attachements) of the message, concatenated together and preened to spam processing easier (none of those bogus rules about how the attachments have funny separators or other such nonsense that Justin seems to like need apply -- just look at the text and see if it says Viagra).
MessageAll	Both the message headers and the message body concatenated together, separated by a single blank line. Yep, it looks like a regular text email message. Scan away without giving it another

thought.

AttachText	All of the text attachements, one to a file. May be individually accepted or rejected. Excellent for virus scanning.
AttachAll	All of the attachements, one to a file. May be individually accepted or rejected. Really excellent for virus scanning, if you can hack MIME decoding.

3.8 Message Formatting Options

Message filtering inserts text into filtered mail messages whenever it detects a virus or other suspicious entity or when the message is found to contain spam. The the text of these inserts is normally compiled into the filter via the "smfopts.h" file. However, the text of any of the inserts can be set from either of the configuration files. This allows the message inserts to be tailored at startup time via the global options file or at message delivery time, on an individual user basis, via the local options file.

The text inserts often have variable information substituted into them (the substitution type is indicated, where required). If this is the case, the rules for doing the substitution follow the conventions used by the C function sprintf. Otherwise, the text inserts themselves must be strings that begin and end with double quotes. If a second quote appears after the first, accumulation of the text insert is ended until another quote appears. The standard escape sequences "\"", "\\", "\r", "\n" and "\t" are recognized. The sample in the Sample Configuration File Section shows all this.

Each text insert is usually inserted as paragraph in the mail message. The insert should be formatted through the use of "\r\n" pairs to read properly if it is inserted into a plain text message (i.e. no lines longer than approximately 76 characters). Do not use any HTML tags (they will be supplied for you by the filter).

Typically, options are entered into the config file on a single line. However, for these message options, the inserted message text can be longer than a single line so you may wish to split it over multiple lines. To do so, simply end each line with '\'. The actual text of the message should be enclosed in double quotes. Concatenation of subsequent lines is done if the preceeding line is terminated with a quote and the next line begins at some point with another quote (this permits indented lines without the indents being included in the message text). Note that no text insert can be longer than 1000 characters and, if you use continuations, no continued line can be longer than 255 characters. The sample in the Sample Configuration File Section shows how to do it.

Obvious uses for the message formatting options are to customize the text shown to users to give a company look and feel or to translate the text into different languages. Here is the list of message formatting options available in the configuration files:

MsgConfig *	Describes attachments that are rejected because they can contain configuration information which may allow a virus to modify a recipient's system. Requires that one substitution (%s) be present for the attachment's type. The default message is: *A %s file, which supplies directives to system\r\nconfiguration or management programs. This might allow your system to be\r\ncompromised by a malicious outsider. There should normally be no reason\r\nfor anyone to send files of this sort to you. Verify with the sender of\r\nthe item what their intent was in sending you this file and that its\r\ncontent is safe before opening it.\r\n*
MsgDelete *	

Indicates that the filter found what it knows to be a virus in the message. Requires no substitutions. The default message is:

SENDMAIL FILTER VIRUS ALERT: This mail message has been scanned by a\r\nsendmail filter and was found to include attachments that are known to\r\ncontain or that actually contain viruses. These items have been deleted\r\nfrom the message. Less malicious items have been rendered relatively\r\nharmless by renaming them. The summary below describes all the attachments\r\nand explains how they are harmful. In the case of the renamed items, if\r\nyou are sure you know who they are from and that they are indeed harmless,\r\nyou can rename them back to their original name and open them. The\r\ndeleted files may only be retrieved by following the instructions for\r\nretrieving the original message. Please proceed with extreme caution.\r\nIf you have any questions or would like assistance, please contact\r\nTECHSUPPORT.\r\n

MsgExec * — Describes attachments that are rejected because they are executable and therefore will allow a virus to modify a recipient's system. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s file, which contains code that is executed\r\nas soon as you open it. This code can do whatever it likes, including\r\nwiping your hard drive, sending sensitive information back to its\r\noriginator and installing viruses on your machine. Verify with the sender\r\nof the item that its content is safe before opening it.\r\n

MsgExploit * — Describes attachments that are rejected because they may be able to exploit a security hole in the application which usually handles them. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s file, which may be able to exploit a\r\nvulnerability or security hole in the application which normally\r\nhandles it, thereby inducing the application to unintentionally execute\r\nvirus code in the file. This code can do whatever it likes, including\r\nwiping your hard drive, sending sensitive information back to its\r\noriginator and installing viruses on your machine. Verify with the sender\r\nof the item that its content is safe before opening it.\r\n

MsgFoundItem * — Names an attachment found that matches a filter criteria. Requires that two substitution (%s, %s) be present, the first for the attachment's name and the second for attachment's type. The default message is:

Found item %s, matching %s.\r\n

MsgFoundType * — Names an inline MIME type found that matches a filter criteria. Requires that one substitution (%s) be present for the inline MIME type's type. The default message is:

Found inline MIME type %s.\r\n

MsgHTML * — Indicates that embedded HTML was found in and removed from the message. Requires no substitutions. The default message is:

Embedded HTML was included in the message. Since most mail readers open\r\nand interpret HTML immediately, in a rather indiscriminate fashion,\r\neverything but the really innocuous tags have been removed. Hopefully,\r\nwhat remains should still be viewable yet be rendered harmless.\r\n

MsgHTMLScript * — Indicates that probably harmful embedded HTML was found in and removed from the message. Requires no substitutions. The default message is:

Potentially harmful embedded HTML was included in the message. The harmful\r\ntags have been removed. What remains will not execute in the way that the\r\nsender intended but that is the price paid for making it harmless.\r\n

MsgInlineMIME * — Describes inline MIME types that are rejected because they may be opened and executed immediately by many mail readers. Requires no substitutions. The default message is:

Inline MIME types are probably not harmful but it is possible they may\r\nbe so. Since many mail readers will open and interpret inline MIME\r\ntyped objects immediately, you have no chance to verify that the typed\r\nitem is harmless before it is opened. Innocuous MIME types are allowed\r\nby this filter but the ones listed above are either unknown or known\r\nto be potentially harmful. Consequently, the inline item has been\r\nrendered harmless (i.e. unopenable).\r\n

MsgMacro * — Describes attachments that are rejected because they may contain macros that can be executed by the application which is normally associated with them. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s, which may can contain harmful macros\r\nthat are executed as soon as you open it. Verify with the sender of the\r\nitem that its content is safe before opening it.\r\n

MsgNest * — Indicates that the filter encountered an error while processing the mail message. Requires no substitutions. The default message is:

SENDMAIL FILTER PROCESSING ERROR: This mail message was being scanned by a\r\nsendmail filter when a processing error occurred. It is entirely possible\r\nthat it might include attachments that are known to contain or are highly\r\nlikely to contain viruses. Had these items have been found, they would\r\nhave been rendered relatively harmless by renaming them. Unfortunately,\r\n due to the processing error, the renaming process might not have been\r\n entirely completed. The summary below describes which files were renamed\r\nand suggests how they might be harmful. You should proceed with extreme\r\ncaution when opening any of the attachments included with this message.\r\nMeanwhile, to report this filter failure or if you have any questions or\r\nwould like assistance, please contact TECHSUPPORT.\r\n

MsgNonSpam * — Indicates that the message didn't contain any spam but that a spam report was requested anyway. Requires no substitutions. The default message is:

This message did not match the criteria for spam, determined by your\r\npersonal spam filter parameters or the global or system spam filter\r\nparameters but you asked for a spam report always, so here it is.\r\n

MsgReject * — Indicates that the filter found what it thinks is a virus in the message. Requires no substitutions. The default message is:

SENDMAIL FILTER VIRUS ALERT: This mail message has been scanned by

a\r\nsendmail filter and was found to include attachments that are known to\r\ncontain or are highly likely to contain viruses. These items have been\r\nrendered relatively harmless by renaming them. The summary below describes\r\nthem and suggests how they might be harmful. If you are sure you know who\r\nthey are from and that they are indeed harmless, you can rename them back\r\n to their original name and open them. Please proceed with extreme caution.\r\nIf you have any questions or would like assistance, please contact\r\nTECHSUPPORT.\r\n

MsgRemailInst * — Indicates that the filter modified the message, for whatever reason and that a copy of the message is available for automatic remailing. Gives instructions about how to have the message remailed. Although there are two substitutions (%s, %s) present in the default message, both are for the name of the saved copy. Should you wish, your message may contain only one substitution. The default message is:

The original content of this message will be available for a short period\r\nof time by sending a message to REMAILROBOT and including the file name\r\n%s\r\nin the subject line. If your mail reader supports it, you can click on\r\nthe link: mailto:%s\r\nOtherwise, you will have to make up the message with the address and\r\nsubject yourself. Please bear in mind that the message may quite possibly\r\ncontain a virus but no warning will be given.\r\n

Further processing of the inserted text is done for HTML messages. If the inserted text contains the string "mailto:...\r\n", as does the default message, this string will be duplicated inside a set of link tags so that the recipient may click on the link and the message will be sent to the remail processing robot. If you wish to take advantage of this feature, the text of your message must include a string that begins with "mailto:" and ends with "\r\n". You could for example use the following message:

The original content of this message will be available for a short period\r\nof time by clicking the link:\r\nmailto:%s\r\n

MsgSaveLoc * — Indicates that the filter modified the message, for whatever reason and that a copy of the message is available from Tech Support. Indicates that the recipient should contact Tech Support and names the saved copy. Requires one substitution (%s) for the name of the saved copy. The default message is:

The original content of this message will be available for a short period\r\nof time from TECHSUPPORT. Contact them and give them this file name:\r\n%s\r\n

MsgSpam * — Indicates that the message contains spam. Requires no substitutions. The default message is:

This message matched the criteria for spam, determined by your personal\r\nspam filter parameters or the global or system spam filter parameters.\r\n

MsgSpecWarn * — Describes attachments that might be harmful. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s, which are not normally harmful but it is\r\nremotely possible that it may be so. Verify with the sender of the item\r\nthat its content is safe before opening it.\r\n

MsgSuspect *

Describes attachments that are rejected because they are strongly suspected of containing a virus, due to the way that the sender has tried to obscure their presence. There is one insert (%s) available in the message for the name of the attachment found. The default message is:

Found %s, a file which appears suspicious.\r\nVerify with the sender and TECHSUPPORT before opening this item.\r\nYou will need to rename the attachment to its proper name before opening\r\nit. You may also want to run additional virus scans against the\r\nattachment, in the mail reader's attachment directory, before opening it.\r\n

MsgTagged * — Describes attachments that are rejected because they can contain tags that cause code to be executed as soon as the attachment is opened. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s file, which may contain tags that cause code to be\r\nexecuted as soon as you open it. This code can do whatever it likes,\r\nincluding wiping your hard drive, sending sensitive information back to\r\nits originator and installing viruses on your machine. Verify with the\r\nsender of the item that its content is safe before opening it and bear in\r\nmind that tagged data of this nature is frequently used to deliver viruses\r\nso use extreme caution.\r\n

MsgUnknownItem * — Names an attachment found that doesn't match any filter criteria. Requires that one substitution (%s) be present for the attachment's name. The default message is:

Found unknown item %s, no filter criteria.\r\n

MsgUnknownWarn * — Describes attachments that are of an unknown type which is probably not harmful but may be so. Requires no substitutions. The default message is:

A file of unknown type which is probably not harmful but it is possible\r\nthat it may be so. Verify with the sender of the item what the file is\r\nand that its content is safe before opening it.\r\n

MsgUnnamedItem * — Indicates that an attachment was found that is unnamed and doesn't match any filter criteria. Requires that no substitutions. The default message is:

Found unnamed item, no filter criteria.\r\n

MsgVirus * — Describes attachments that are rejected because they have a history of being virus delivery vehicles. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s, which has frequently been chosen as a\r\nvirus delivery vehicle in the past and is almost certain to contain a virus.\r\nVerify with the sender and TECHSUPPORT before opening anything. You\r\nwill need to retrieve the original message in order to obtain any\r\nattachments that you wish to proceed with opening.\r\n

MsgVirusFound * — Indicates that a MIME entity or attachmed file was found to actually contain a virus, by the virus scanner. One substitution (%s) must be present, which will be replaced by the entity type, followed by the word "entity", or the attachment's name preceded by the word "file". The default message is:

Virus found in %s.\r\n

MsgVirusScanned * Describes attachments and/or MIME entities that are rejected because they were found to contain a virus. Thre are no substitutions. The default message is:

The MIME entities and/or files listed above were scanned by a virus\r\nscanner and found to contain actual viruses. Do not, under any\r\ncircumstances, unless you are absolutely certain you know what you are\r\ndoing, open any of them. Verify with the sender and TECHSUPPORT\r\nbefore opening anything. You will need to retrieve the original message\r\nin order to obtain any attachments that you wish to proceed with opening.\r\n

MsgWarning * Indicates that the filter found HTML tags, which it thinks might be harmful, in the message. Requires no substitutions. The default message is:

SENDMAIL FILTER WARNING: This mail message has been scanned by a sendmail\r\nfilter and was found to include objects and/or HTML tags that may be of\r\na malicious nature. The summary below describes them and suggests the\r\naction you should take with respect to them. If you have any questions or\r\nwould like assistance, please contact TECHSUPPORT.\r\n

MsgWordHandled * Describes attachments that are rejected because they are erroneously interpreted by MicroSoft Word on some systems. Since Word will execute macros in the attachments, the attachments are rejected. Requires that one substitution (%s) be present for the attachment's type. The default message is:

A %s, which on some systems may be handled\r\nincorrectly by Microsoft Word. This would allow a malicious person to\r\nsend an attachment, with this file type, that was actually a Word\r\ndocument. Since Word would handle this document instead of the intended\r\napplication, this would present a backdoor through which harmful macros\r\nthat are executed as soon as you open the attachment could be sent.\r\nVerify with the sender of the item that its content is safe before\r\nopening it. You may also need to rename the item to its proper name,\r\nif your email program renames it to \".doc\".\r\n

3.9 Sample Configuration File

What follows is an example of a fairly typical local (user) configuration file. It turns on or off some filtering and spam options that aren't usually set that way by default. In the interests of brevity, it changes the text of most of the message inserts to be less verbose. Finally, it blacklists all messages from a particularly nasty spammer.

```
 #
# Filtering options.
#
AutoRemailing
ReplaceHTML No
SpamAlways
SpamDelivery Deliver
SpamLevel 3
#
# Filtering messages.
#
MsgWarning \
  "SENDMAIL FILTER WARNING: This mail message has been scanned by a\r\n"\
```

```
  "sendmail filter and was found to include objects and/or HTML tags that\r\n"\
  "may be of a malicious nature.  The summary below describes them and\r\n"\
  "suggests the action you should take with respect to them.\r\n"
MsgReject \
  "SENDMAIL FILTER VIRUS ALERT: This mail message has been scanned by a\r\n"\
  "sendmail filter and was found to include attachments that are known to\r\n"\
  "contain or are highly likely to contain viruses.\r\n"
MsgSaveLoc \
  "The original content of this message will be available for a short\r\n"\
  "period of time in %s.\r\n"
MsgRemailInst \
  "The original content of this message will be available for a short\r\n"\
  "period of time by sending a message to spamrobot mentioning:\r\n"\
  "%s\r\n"\
  "in the subject line.  Or you can click on the link:\r\n"\
  "mailto: spamrobot?%s\r\n"
MsgInlineMIME \
  "Inline MIME types are probably not harmful but it is possible they may\r\n"\
  "be so.  Consequently, the inline item has been rendered harmless.\r\n"
MsgVirus \
  "A %s, which has frequently been chosen as a virus\r\n"\
  "delivery vehicle in the past and is almost certain to contain a virus!\r\n"
MsgWordHandled \
  "A %s, which on some systems may be handled\r\n"\
  "incorrectly by Microsoft Word.\r\n"
MsgMacro \
  "A %s, which may can contain harmful macros\r\n"\
  "that are executed as soon as you open it.\r\n"
MsgSpecWarn \
  "A %s, which are not normally harmful but it is\r\n"\
  "remotely possible that it may be so.\r\n"
MsgExec \
  "A %s file, which contains code that is executed\r\n"\
  "as soon as you open it.\r\n"
MsgConfig \
  "A %s file, which supplies directives to system\r\n"\
  "configuration or management programs.\r\n"
MsgExploit \
  "A %s file, which may be able to exploit a\r\n"\
  "vulnerability or security hole in the application which normally\r\n"\
  "handles it.\r\n"
MsgTagged \
  "A %s file, which may contain tags that cause code to be\r\n"\
  "executed as soon as you open it.\r\n"
MsgHTML "Embedded HTML was included in the message.\r\n"
MsgHTMLScript \
  "Potentially harmful embedded HTML was included in the message.\r\n"
#
# Spam fast path stuff.
#
whitelist_from myfriend@hometown.org
whitelist_from theboss@workplace.org
blacklist_from *@spammers.com
blacklist_from *@badguys.com
```

4. User Support

MailCorral provides virus and spam filtering at the system level for all mail delivered by sendmail. While some filtering options may be applicable to all email delivered, many users may prefer to set certain options according to their individual tastes. MailCorral allows users to tailor how it filters their mail via individual config files or a configuration database.

4.1 Configuration Methods

Individual configuration file support allows a user to create a config file in their home directory ("~/.sendmailfilter", by default, unless "smfopts.h" is changed prior to compiling the filter) where all of the local options may be specified. These local options are applied to each message, based on the recipient of the message. When a message is received, the recipient's username is determined and then their local options file is looked up and processed to arrive at the options used to filter the message.

In many instances, MailCorral is employed by organizations that have a large number of users (e.g. ISPs) but where the users are essentially virtual users. That is to say, the users don't have traditional accounts in the sense that they have home directories or other accoutrements normally associated with real users. More importantly from our point of view, they have no place to conveniently store a user options file.

If your system has virtual users, individual user support is still possible through MailCorral's DBM database configuration file. Using this option, all of the user settable parameters are stored in a single configuration database, which may be located anywhere. The virtual user's name is used to store and lookup the parameters and everything else works as if they were a real user. This feature is enabled via the "-c" command line option or the "ConfigDB" config file parameter.

Note that, when you are using virtual user support, you might also want to use fully qualified names for storing configuration information in the configuration database. Since it is possible for the same username to exist in two domains (e.g. "custserv"), domain information must be used to qualify the username and ensure that the name used to store configuration information is unique. This is done by specifying the "-q" command line parameter or "QualifyNames" in the global configuration file.

In the case of the individual configuration file, the user may utilize a text editor to edit the parameters within it. However, in the case of the DBM database, this isn't possible. Either way, a nice user interface, that allows a user to easily edit their configuration options, would be better.

The purpose of MailCorral's configuration editor ("ConfigEdit.cgi") is to provide individual users with a simple, easy to use method of editing configuration parameters. It is a Perl program that runs as a CGI script under a Web server. The user visits a configuration form, with their Web browser, where they see all of their configuration parameters and where they can edit them. Once they have made changes to the form, the results are posted back to the same CGI script which then updates the local config file or DBM database.

The configuration editor is actually a generic config file and/or DBM database editor that is driven by an HTML template file. It can be set up to edit one or more config files and/or DBM databases in a single form. The form, since it is governed by an HTML template, can be tailored to have whatever look and feel a site desires, probably that of their standard configuration pages.

A sample HTML template file ("MCEdit.html") is shipped with MailCorral that implements a simple form with all of the configuration parameters that a user would want to edit. This form defaults to edit "~/.sendmailfilter". A comment immediately after this file name, in the template, shows how to edit the DBM

database "/etc/mail/MCUsers". You can use the template pretty much as is or alter it to get something more to your liking.

Note to RPM installers. The configuration editor and sample template are installed in the "/etc/mail" directory, for want of a better place to put them, not knowing how you have your Web server set up or even if one exists at all. You can link to them there or move them to wherever else you want.

4.2 Configuration Editor

The configuration editor ("ConfigEdit.cgi") should be moved to the appropriate CGI directory for your Web server and given execute permissions, etc. You may have to alter the path to your Perl interpreter in the first line of the script. The editor is invoked by placing a link to it in a HTML document. The URL of the template to be loaded (minus the address of the server) is given by the "TplURL" parameter. Hence, the link should look something like:

 zzz

You can debug this program and your templates by adding a value of "Debug" to the URL. For example:

 zzz

The purpose of the template is to describe a form which displays and allows the user to edit the configuration values in a Unix-style configuration file. This program is also reinvoked by the form in the template to process the edits to the form data and update the configuration file. This reinvocation of the program should look like:

```
<form name="Config" action="/cgi-bin/ConfigEdit.cgi?TplURL=MCEdit.html"\
      method="post">
```

Any parameters that are to be replaced in the template are given by names which must exactly match the parameter names found in the config file. You can have as many parameters as necessary. Parameters will be replaced by blocks of HTML so, if you want the output to look presentable, each parameter should occur on a line by itself.

In addition to the HTML necessary to display the parameters, the template file must also contain three special comment blocks which describe how the parameter replacements should be done. These can appear anywhere in the template and will be removed before transmission.

The first block, named "configfile", simply gives the name of the config file who's parameters are to be displayed. Multiple config file names can be supplied, separated from each other by commas, in which case the field types (see below) need to be qualified with a number, indicating which config file the field comes from (the default, if omitted, is 1, the first config file).

All of the usual pathname expansion features are available. If the path begins with "~/", the parameter UserName must be supplied as one of the values when the script is invoked and this user name will be used to look up the home directory for the user. For example:

<!-- configfile ~/.sendmailfilter -->

requires

```
<a href=/cgi-bin/ConfigEdit.cgi?UserName=JoeBlow&\
    TplURL=MCEdit.html> zzz </a>
```

Note that there is a potential security hole caused by this feature. A malicious person could cause a parameter file for a particular user to be updated by figuring out which parameters to pass to this program. As such, the program should be put in a password protected directory where it can only be executed by legitimate users who have previously supplied the correct password to the Web server.

Also note that, in order for the form to be able to recall this script with the user name needed to update the config file, the "$ConfigEditUserName$" parameter in the template file will be updated with the username passed on the command line. Although this precludes having a parameter in the config file with this name, it does allow the username to be passed along from invocation to invocation. You should probably use this feature as follows:

```
<form name="Config" action="cgi-bin/ConfigEdit.cgi?\
    UserName=$ConfigEditUserName$&\
    TplURL=MCEdit.html" method="post">
```

Finally, if you want to be able to access user files from Apache and to be able to create user files with the correct mode and permissions, you will need to run this script as setuid root. This shouldn't pose a problem because the only I/O that is done is to the config file, which is specified in the HTML template, although it is possible to supply a username and have a file created in that user's directories, wherever the template says it should be put. Consequently, I reiterate that this program should be put in a password protected directory where it can only be executed by legitimate users who have previously supplied the correct password to the Web server.

An alternate form of the user config file is allowed to support DBM-based user config files. If the file name starts with a pair of slashes (i.e. "//"), it is assumed to be a DBM database which contains the parameters as key/value pairs. In this case, the parameter UserName must also be supplied as one of the values when the script is invoked and this user name will be used to create unique keys for each user. For example:

```
<!-- configfile //global/userparms -->
```

requires

```
<a href=/cgi-bin/ConfigEdit.cgi?UserName=JoeBlow&TplURL=MCEdit.html> zzz </a>
```

All of the discussion above, about the updating of the $ConfigEditUserName$ parameter, security and permissions applies equally to this database. The keys in the database will have their names composed of "username|parmname" while parameters that can have multiple values will have a numeric counter added to their keys to make them unique.

The second template comment block, named "fieldnames" is a list of field names plus information about them. Each field name must appear on a line by itself. The line contains the name, a type (including an optional config file index), a template name plus some additional information. The field types can be "btn" (button), "chk" (checkbox), "txt" (text) or "rpt" (repeating text). If the optional config file index is supplied (when the "configfile" section names more than one config file), it follows the field type immediately with no spaces in between (e.g. "chk2"). For each of the field types, here's how their entry in the template file must look:

```
fieldname|btn|tpltname|values|...
fieldname|chk|tpltname|checkedvalue|uncheckedvalue
```

fieldname|txt|tpltname|value
fieldname|rpt|tpltname1|tpltname2

The templates referred to above are small chunks of HTML that describe how to display the parameter as a field in the Web page (presumably a form). The fieldnames must match the parameter names in the config file. If a field takes values, they follow the template names.

The values given for each field are the default values, set before the config file is read. In the case of the checkbox, the unchecked value is set as the default value of the field but, when the substitution is made into the template, if the value being substituted matches the checked value, that value is substituted into the template followed by the tag "checked". Regardless of which value the parameter is set to, the checked value is always substituted with the only difference being whether the "checked" tag is added or not. Repeating fields have no default value but a single, extra, empty field will be created to allow for additions. The second template in a repeating field is used to provide a delete checkbox. The checkbox is left off for the generated, empty field.

The third special comment block, named "fieldtemplates" must contain the named field templates referred to in the first comment block. These also must occur one per line and look like:

tpltname|replacment HTML

In each template, the name of the field (i.e. the parameter name) replaces any occurrences of "@@" and the values replace any occurrences of "%%" and/or "^^", in the order in which they are seen and in the order of the values. The "@@" are substituted first. Then the "%%" are substituted next. Finally the list of values is re-substituted to replace the "^^", if any, mainly for the purpose of labeling radio buttons.

The HTML template that is given will be read and searched for all occurrences of the parameters in the config file that match field names in the fieldnames section of the template. These parameters must be surrounded by two dollar signs in the template (e.g. "$ParmName$") and must match the parameter names found in the config file exactly (except for case insensitive). Each occurrence of a parameter will be replaced in situ by its field template, after substitutions are made. Here is an example:

```
<html>
<head>
<!-- configfile ~/.sendmailfilter,~/.otherparms -->
<!-- fieldtemplates
Button1|<input type=radio name=@@ value=%%> ^^ \
        <br><input type=radio name=@@ value=%%> ^^ \
        <br><input type=radio name=@@ value=%%> ^^
Button2|<input type=radio name=@@ value=%%> ^^ \
        <input type=radio name=@@ value=%%> ^^
Check1|<input type=checkbox name=@@ value=%%>
Text1|<input type=text name=@@ value=%% size=20 maxlength=50>
Delete1|  <input type=checkbox name=@@ value=%%>
-->
<!-- fieldnames
ParmBtn1|btn|Button1|Top|Middle|Bottom
ParmBtn2|btn|Button1|Fast|Normal|Slow
ParmBtn3|btn|Button2|High|Low
ParmCheck|chk|Check1|No|Yes
ParmText|txt|Text1|Some text
ParmList|rpt|Text1|Delete1
OParmTxt|txt2|Text1|Other text
OParmChk|chk2|Check1|Maybe Not|Maybe
-->
```

```
</head>
<body>
<form name="Config" action="cgi-bin/ConfigEdit.cgi?\
        UserName=$ConfigEditUserName$&\
        TplURL=MCEdit.html" method="post">
<table align=left cols=2 cellpadding=0 cellspacing=0 border=0>
<tr><td width=300>
ParmBtn1 - The button 1 parameter ...
</td><td>
$ParmBtn1$
</td></tr>
     .
     .
     .
</table>
<hr>
<table align=left cols=2 cellpadding=0 cellspacing=0 border=0>
<tr><td width=300>
OParmTxt - The other config file text 1 parameter ...
</td><td>
$OParmTxt$
</td></tr>
     .
     .
     .
</table></form>
</body></html>
```

The config file itself follows the normal rules for Unix-style config files. However, for the sake of making this program simpler, the updates are only done to the tail end of the config file. Also for the sake of simplicity, the parameter names must not end with one or more numbers nor include the string "__delete__" in them.

Only the parameters named in the "fieldnames" section of the template file are written to the form and updated in the config file. All other parameters are ignored. If you are smart, you'll put the parameters that are to be ignored at the start of the file and those that are to be updated at the end. This program makes no attempt to preserve the order of comments or the insertion order of updated parameters. Comments on the same lines as updated parameters are lost.

4.3 Web Page Template

The sample HTML template file ("MCEdit.html") should be copied to the appropriate HTML directory for your Web server. As supplied, the template implements a simple form with all of the configuration parameters that a user would want to edit. This form defaults to edit "~/.sendmailfilter". A comment immediately after this file name, in the template, shows how to edit the DBM database "/etc/mail/MCUsers".

Since the template is editing a config file that needs a user name to locate it, you will need to invoke the configuration editor with the "UserName" parameter. Probably the easiest way to do this is to create a simple, top-level form that contains a single text field which asks for the user's name. The submit button of the form should invoke the editor as follows:

<a href=/cgi-bin/ConfeigEdit.cgi?UserName=JoeBlow&TplURL=MCEdit.html>

Such a simple form might look like:

```
<form name="User" action="/cgi-bin/ConfigEdit.cgi" method="get">
```

```
  <input type="hidden" name="TplURL" value="TplURL=MCEdit.html">
  <input type=text name="UserName" value="" size=20 maxlength=32>
</form>
```

The configuration parameters that can be updated by the sample template are: AddXTags; AutoRemail; SpamAlways; SpamDel; SpamFast; SpamLevel; blacklist_from; and whitelist_from. Each of these parameters can be found documented in the Configuration section.

The parameters SpamDel and SpamLevel are implemented as radio buttons, in this case with three values, via the field template "Button1". The three values that follow the field name (e.g. "Corral", "Deliver" and "Trash") correspond to the three buttons in the field template. As an example, the first button in the field template for the SpamDel button would get expanded to read:

<input type=radio name=SpamDel value="Corral"> Corral

A second radio button field template, Button2, is included in the sample template as an example but it isn't used. It would be used for a parameter that had four values, like the DebugLev global parameter (it is possible to have a form that edits the global parameters too). You can leave it there (its harmless) or remove it, if you want.

The AutoRemail, SpamAlways and SpamFast parameters are implemented as checkboxes, via the Check1 field template. Note that the first value following the field template name, in the field, is the value that is chosen when the box is checked, while the second value is the unchecked value. In the sample template file, the operation of SpamFast is backwards, with "No" being the value chosen when the box is checked.

The AddXTags, blacklist_from and whitelist_from parameters are all text fields with identical characteristics, implemented by the field template Text1. The AddXTags field has no default value so it will appear empty when the form is brought up, unless the config file has some setting for it.

The blacklist_from and whitelist_from parameters, in addition to being text fields, are repeating fields. That means that they can have multiple values in the config file or database and that each value will result in a separate field in the generated form. On top of this, the Delete1 field template will be added to each occurrence of the text field, except the last one, which will be given an empty value. This allows additions of new values and updates or deletions of existing values.

The actual form itself is contained in a table with two columns. On the left is a paragraph explaining what each option is. On the right is the form field where the parameter value can by typed in, the checkbox checked or the radio buttons clicked. The repeating fields have two items on the right, a text field and a checkbox. There is a column heading above them so that the user can tell what's what.

Note that the Language parameter was not included in the sample template file, since most installations of MailCorral will not require language support. If you should require it, add a multi-button radio field that has the languages you need and create a set of language configuration files in a convenient directory. Set the values of the radio buttons to the names of the language configuration files and you should be in business.

5. Interoperability

5.1 Test Suite

Most email viruses employ similar propagation techniques and are, consequently, very similar in external appearance to one another. That being the case, it is possible to design an email virus filter that removes viruses based on outward appearance. This is highly desirable, since it will ensure that, even when a new virus comes along, the filter will be able to screen for it.

On the other hand, the penalty for failing to detect a virus in an email message is high. We like to know if we've covered all the bases in MailCorral by being sure that we detect all of the important propagation techniques?

The Email Filter Validation Suite consists of generic email messages which can be passed through an email filter to test all of the popular methods of virus propagation. If the filter detects each message and handles it correctly, it is probably ready for prime time.

Each time we make any changes to MailCorral, we pass all of the email messages from the validation suite through it and examine the results. Each time we discover a new type of virus that must be specifically scanned for, we construct a test message and add it to the validation suite. In this manner, we ensure that no new bugs are introduced nor are any reintroduced.

5.2 Working With SpamCorral

MailCorral can redirect all received spam to the spam corral, instead of delivering it to the spammer's intended victim. All in all, not a bad plan but it is possible that a piece of spam could prove valuable (stranger things have happened). It is even possible that a non-spam message might be misidentified as spam and rustled into the corral by mistake.

The ultimate decision, about whether a piece of spam is valuable or not, is best left up to the intended recipient. After all, they are really the only ones who know whether they actually want so see the spammer's message or not. But, if the recipient is shown every piece of spam and asked to make a decision about whether they want to see it or not, the solution is no better than the problem. The compromise solution is to only ask them once or twice a day, in a single message, and to provide a summary that contains sufficient information to allow them to decide, quickly and easily, whether they want to see the spam or not.

To accomplish this goal, a spam handling package, called SpamCorral, works hand in glove with MailCorral to provides spam notifications, through a program which can be run periodically by a cron job. When this program is run, it will send email notification messages to all of the recipients of spam. It summarizes each of the messages received since the last time it was run, giving the sender's address, the subject, the delivery date/time and the associated spam statistics (as a percentage, with 100% being the threshold for classification of a message as spam). When a user receives the notification, they can optionally reply to the message (using their mailer's reply function), retaining the description of any pieces of spam which they wish to see and deleting the description of those which they don't. The action is simple and natural, in that it is just like replying to any other piece of email that they receive.

A second program in the spam handling package, the spam handling robot, listens for messages sent to it, as replies to notification messages, requesting that spam be extracted from the mail corral and remailed to the original recipient. Upon verification of the sender's right to remail the spam, the corralled messages will be

remailed to them but, this time, they will pass directly through the sendmail filter unscathed. The operation of the spam handlers is automatic and unattended, simply responding to requests from the recipients to remail all of the spam that they ask to see. No intervention by administrative personnel is required. Furthermore, no important or interesting messages are ever dropped by accident. The recipient has final say in all decisions.

As was mentioned above, MailCorral works very closely with SpamCorral, putting the spam into the corral, where it awaits its final disposition at the user's behest. Furthermore, when the spam handler remails spam that has been released by the recipient, MailCorral does some special processing (very minimal in nature) to carry out delivery of the spam with a minimum of fuss.

5.3 Creating Your Own Spam Handler

If MailCorral is asked to redirect received spam to the spam corral, you can choose to write your own spam handling programs to process the spam that is placed therein. Here are some notes on how to go about this.

Corralled spam is completely formatted and ready for remailing. If any viruses or other obnoxious entities were found within the message, in addition to its being spam, these have already been removed. Spam headers have been inserted as have any descriptive messages. To remail the message for delivery, all that need be done is to invoke sendmail and pass it the message exactly as it is stored.

When spam is remailed, be aware that there is a header in the message (SPAMHDRBYPASS in smfopts.h) that contains a key which will instruct the filter to bypass processing of the message the second time around. This header contains a copy of the message date/time, encrypted using the bypass tag key (SPAMKEY in smfopts.h). For the message to be successfully remailed, without additional processing, this header must be kept intact, the date/time must not be changed from when the message was originally sent and two other criteria must be met. The mailer that remails the spam must be "local" and the name of the remailer must not contain an '@' (i.e. the message must be from the local domain). All remailed spam will be tagged by the filter as "[SPAM]" in the subject line.

The name of the original recipient of a message, from the envelope, is included in the actual name under which the spam is stored in the corral. Remailing should be done with this name, not the to name in the message headers. Note that, if there were multiple recipients on a piece of spam's envelope, one copy of the message is stored without a recipient name. A symbolic link is then made to the stored message for each of the envelope recipients. Thus, any program that processes corralled spam must consider symbolic links as well as files and must disregard files that have no recipeint included in their name. Here are a couple of examples:

```
-rw-------   1     1650 Aug  3 14:14 spam_to_jblow_3D4C1D90
lrwxrwxrwx   1       48 Aug  3 14:21 spam_to_jblow_3D4C1F0D
                                      -> spam_to__3D4C1FD0
lrwxrwxrwx   1       48 Aug  3 14:21 spam_to_jdoe_3D4C1F0D
                                      -> spam_to__3D4C1FD0
-rw-------   1     1644 Aug  3 14:21 spam_to__3D4C1F0D
```

To process the above files, send out three pieces of remailed spam, two to "jblow" and one to "jdoe". A single copy of the first message ("spam_to_jblow_3D4C1D90") is sent to "jblow". Multiple copies of the second message ("spam_to__3D4C1F0D") are sent to "jblow" and "jdoe". If you want to figure out when to delete the messages, any message with a recipient name in its file name can be deleted immediately, as can any link. File names with no recipient name in them can only be deleted when no links pointing to them remain.

5.4 Using SpamAssassin To Classify Spam

If you wish to use SpamAssassin as your spam arbitron, to classify messages that are spam, you need to first install the package (either by selecting the version that comes with your OS distro or by downloading and installing the source from: https://spamassassin.apache.org/downloads.cgi). Follow the instructions for installing SpamAssassin, including the spamd daemon. Make sure to install the system startup script in the appropriate place (e.g. /etc/init.d) and verify that spamd is set to automatically start at boot time, in sequence before sendmail (and terminate after sendmail at system shutdown). Turn the startup script on so that spamd starts automatically at boot time.

The spamd options that you should consider using are:

-c Create user preferences files (e.g. in the ~/.spamassassin directory) where auto whitelist and Bayesian statistics will be kept for each, individual user.

-d Daemonize, that is to say run as a daemon, waiting for work (in this case from MailCorral). You must specify this option.

-L Use local tests only. Do not look up addresses, etc. in DNS. This makes spamd be a lot quicker to judge, although it may result in more false negatives.

-p *portnum* The port number to listen on. By default, spamd listens on port 783. If you'd rather use some other port number, specify it here. The rest of the MailCorral documentation makes mention of port 2527 so you might wish to make use of that number.

Earlier versions of SpamAssassin required that the "-a" parameter be used with spamd to cause auto whitelisting to be enabled. With the latest versions of SpamAssassin, auto whitelisting is turned on by default, and is either enabled or disabled with the "use_auto_whitelist" option in the configuration file. The "-a" parameter is deprecated and should be removed when running the later versions of SpamAssassin. On the other hand, if you are still using an earlier version, by all means, turn on auto whitelisting with "-a".

A system-wide auto-whitelist can be used, by setting the auto_whitelist_path and auto_whitelist_file_mode configuration commands as you like to define where the whitelists will be stored:

```
auto_whitelist_path   /var/spool/spamassassin/auto-whitelist
auto_whitelist_file_mode   0666
```

Note that, if SpamAssassin is used as your spam arbitron, MailCorral uses the SpamAssassin whitelist and local configuration files to implement a fast path lookup for whitelisted and blacklisted mail senders. This eliminates the need to send received mail messages to SpamAssassin purely to find out whether they are white or blacklisted, which can result in a considerable speedup in handling white and blacklisted messages.

To implement this feature, MailCorral must know where SpamAssassin is installed. Normally, it is installed in a well-known location and this feature should just work by default. However, if you install SpamAssassin somewhere other than the usual spot, you will probably need to change the path names found in the module spamfilter.c and recompile MailCorral. Here is the list of path/file names that MailCorral usually looks for:

```
/usr/local/share/spamassassin/60_whitelist.cf
/usr/share/spamassassin/60_whitelist.cf
/var/lib/spamassassin/*/60_whitelist.cf
/usr/local/etc/spamassassin/local.cf
/usr/pkg/etc/spamassassin/local.cf
/usr/etc/spamassassin/local.cf
```

/etc/mail/spamassassin/local.cf
/etc/spamassassin/local.cf
~/.spamassassin/user_prefs
The way that MailCorral searches the list is intuitive, except for the path/file names that have an asterisk in them. In those cases, the entire directory path at and below the point at which the asterisk occurs is searched for the file named. This supports SpamAssassin's use of versioned white and blacklists.

5.5 Using ClamAV To Detect Viruses

If you wish to use ClamAV as your virus arbitron, to detect messages that contain viruses, you should first install it on your system (either by selecting the version that comes with your OS distro or by downloading and installing the source from: https://www.clamav.net/downloads). Follow the instructions for installing ClamAV, including the clamd daemon. Make sure to install the system startup script in the appropriate place (e.g. /etc/init.d) and verify that clamd is set to automatically start at boot time, in sequence before sendmail (and terminate after sendmail at system shutdown). Turn the startup script on so that clamd starts automatically at boot time.

The clamd options (which can only be set in the configuration file "/etc/clam/clamav.conf") are pretty much OK, by default. The only options that we suggest that you should consider using are:

PidFile *</var/run/clamav/clamd.pid>* ClamAV may put its PID file somewhere where traditional startup scripts can't find it. We suggest that you force it to use "var/run/clamav/clamd.pid" as this is where traditional startup scripts expect PID files to be.

TCPSocket *<socknum>* The TCP socket number to listen on. By default, clamd doesn't listen on any TCP sockets so you must specify this parameter along with a socket number. The rest of the MailCorral documentation makes mention of port 2528 so you might wish to make use of that number.

Make sure that you keep ClamAV up to date, as viruses are constantly evolving. Also, be sure to run freshclam at regular intervals (at least once or twice daily) to keep the virus signatures file up to date. This is your first line of defence against new viruses. ClamAV will notify you, in your log file, about updates. If you run a log watcher program like "logwatch", you should receive email notification whenever a new version of ClamAV is available. Once again, keep it up to date!

GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for

drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.