SENDMAIL 8.8

ebi@india.hp.com

Hewlett Packard Company,

International Software Operations

30 Cunningham Road,
Bangalore,
India

1.0 Introduction

Sendmail is an implementation of an inter-network mail routing facility. It is the heart of TCP/IP based mail communication systems. Sendmail relays inbound and outbound mails to appropriate programs for delivery or further routing based on information specified in a configuration file

Sendmail has been implemented in public domain. Both old versions based on sendmail 5.65 as well as new versions of sendmail based on sendmail 8.x are currently supported on HP-UX. However, sendmail 8.x supports many feature additions and configuration options that are not present in sendmail 5.65. Hence, there is a growing trend to move from older versions of sendmail to sendmail 8.8. It is the intention of this paper to make the users aware of the important new features of the latest version of sendmail (sendmail 8.8), and help them migrate to the latest version. Some of the salient features of sendmail 8.8 are:

1. Increasing the number of MX hosts for a single hostname to hundred. This feature is primarily meant for large Internet Service Providers.

2. Implementation of the ESMTP ETRN command. This is an SMTP service extension whereby a SMTP client can request a SMTP server to start the processing of its mail queues for messages that are waiting at the server for the client machine. This extension is meant to be used in the start-up of a SMTP session as well as for mail nodes that have transient connections to their service providers.

3. Allowing the specification of the new named sendmail.cf rulesets (check_mail,check_rcpt, check_compat and check_relay) to validate the addresses passed as arguments to SMTP commands. These rulesets can be used to prevent spamming.

4. Implementation of 7bit->8bit MIME conversions. This is implemented by the addition of a new mailer flag and is typically used for the local delivery agent. This feature can be used to decode MIME encoded text attachments, if the mail reader on the recipient machine cannot read MIME-encoded mail**.**

We will see these features in more detail in this paper

2.0 New Features of sendmail 8.8

2.1 ETRN

Extended TURN (ETRN) is an extension to the SMTP service. In this an SMTP client and server may interact to give the server an opportunity to start the processing of its queues for messages to go to a given host. This extension is meant to be used in startup conditions as well as for mail nodes that have transient connections to their service providers.

The previous TURN command was a valid attempt to address the problem of having to start the processing for the mail queue on a remote machine. However, the TURN command presents a large security loophole. As there is no verification of the remote host name, the TURN command could be used by a rogue system to download the mail for a site other than itself.

This has been addressed in the design of the ETRN command. The security loophole is avoided by asking the server to start a new connection aimed at the specified client In this manner, the server has a lot more certainty that it is talking to the correct SMTP client.

2.2 DSN

Delivery Status Notification (DSN) is a function of the Mail Transfer Agent (MTA). There are two different kinds: positive and negative delivery status notifications. Negative delivery status notifications have been available for a long time. Positive delivery status notifications have not been available as a standard. sendmail prior to version 8.7 (and other MTAs) supported Return-Receipt-To: There are several problems with this

1. the recipient's MTA does not support Return-Receipt-To: ;
   This problem is addressed by DSN in the following way: if it delivers an email to an MTA which doesn't support DSNs it will tell the sender so (using a MIME message as defined in RFC 1892):

```
----- The following addresses have delivery notifications -----

RECIPIENT  (relayed to non-DSN-aware mailer)

----- Transcript of session follows -----

RECIPIENT relayed; expect no further notifications
```

2. How to use it with a mailing list.
   This problem is addressed by DSN too: you have to specify for each recipient, whether you want a DSN for a particular recipient or not.

   In sendmail 8.8 we have three new command line flags to pass in DSN parameters:

   -V envid (equivalent to ENVID=envid on the MAIL command),

   -R ret (equivalent to RET=ret on the MAIL command), and

   -Nnotify(equivalent to NOTIFY=notify on the RCPT command).

Note that the -N flag applies to all recipients; there is no way to specify per- address notifications on the command line, nor is there on equivalent for the ORCPT=per-address parameter.

2.3 New options

2.3.1 AllowBogusHELO

Prior to V8.7, sendmail would accept without complaint an SMTP HELO command that omitted the hostname. But from V8.7, omitting the hostname will result in the following errors:"501 helo requires domain address". This option can be used to accept connection from sites which do not obey the protocol by not giving the hostname. The AllowBogusHelo option is used like this:

```
O AllowBogusHelo=bool
```

The bool is of type boolean. If bool is absent, the option defaults to true (do allow hostname to be omitted). If the entire declaration is missing, the default is false (require the hostname to be present).

2.3.2 ConnectionRateThrottle

Whenever an outside site connects to sendmail's SMTP port, sendmail forks a copy of itself. The copy processes the incoming mails. If the number of simultaneous connection exceed a particular value then the system is overloaded. To avoid this, an option is given to slow down the acceptance of connections when the number of children becomes too high. This slowing is achieved by ConnectionRateThrottle. The syntax is
O ConnectionRateThrottle=num

The num is of type numeric. If it is present and greater than zero, connections are slowed when more than that number of connections arrive within one second. the number is less than or equal to zero, or absent, no threshold is enforced. If the entire option is missing, then the default becomes zero.

2.3.3 MaxDaemonChildren

As mentioned earlier the sendmail forks to process each incoming connection, and it forks to process its queue. The number of forked process can be limited by defining the MaxDaemonChildren option:

O MaxDaemonChildren=num

The num is of type numeric and specifies the maximum number of forked children that are allowed to exist at any one time. If num is less than or equal to zero, or if this entire option is missing no limit is imposed. If num is greater than zero, connections that cause more than that number of forked children to be created will be rejected. Setting this in the configuration file can lead to a denial of service attack. This option is appropriate for use in certain queue processing situations.

2.3.4 MustQuoteChars

All addresses are composed of address information and non address information. The non address information is a users fullname or something similar. RFC822 requires that certain characters be quoted if they appear in the non address part of an address. @,;:\()[].'These characters are the default for the MustQuoteChars option. Characters can be added to this mandatory list using this option.

O MustQuoteChars=more

Where more is of type string and is the list of additional characters that needs
to be quoted in the nonaddress part of it.

2.3.5 RunAsuser

On firewalls, for reasons of additional security, it is often desirable to run send-mail as a user rather than root.

O RunAsUser=user:group

Here, user is either the uid of the identity you want sendmail to run under or a symbolic name for that identity. has or a symbolic link of the identity. And the group is the gid of the corresponding group.

Note that running as non root can lead to problems, especially on machines that do more than simply relay mail between networks. As non-root sendmail may not be able to read some :include files, will certainly not be able to read protected ~/ .forward files and won't be able to save messages into queue. This option is intended to be used on a firewall machine.

2.3.6 SingleThreadDelivery

In processing a queue,where parallelism is not necessary, sendmail can be set up to be single threaded. This ensures that only a single sendmail will ever be delivering to a given host at a given time .Single-threaded delivery is enabled with SingleThreadDelivery option.

O SingleThreadDelivery=bool

The argument bool is of type Boolean. If the argument is missing the default value is true. If the whole option is missing the default becomes false. This will work only if HostStatusDirectoryoption is also declared. Its not advisable to have this configured in your sendmail.cf file. To understand why, consider an ongoing queue run to a host that is receiving many messages. If inter-active user mail arrives during that run, the sendmail process executed by the users MUA may find that it cannot send the message because it is single threaded and the other sendmail has the host locked. In that case the users message will be queued and will be in the queue until the next queue is run. Even if your site is on the internet, one large message to a slow site can cause interactive mail for that site to be wrongly queued

2.3.7 Timeout.iconnect

When sendmail attempts to establish a network connection to another host, it uses the connect system call. If the connection is fails, that system call will time out after an amount of time that varies with operating system. This time can also be overridden by passing a parameter to the system call.

When outgoing mail is first processed, mail to responsive hosts should precede mails to sluggish hosts. To understand why, consider that all mail is processed serially during each queue run. If a sluggish source precedes all other hosts in the queue, those other hosts will not even be tried until the sluggish hosts finishes or times out.

With this in mind, the very first time sendmail tries to deliver a message it should enforce a shorter connect time-out than it should for later attempts.

These values can be set using

O Timeout.iconnect=10s <- time-out for first connection

O Timeout.connect=3m <- time-out all others.
The default value is to have all the values to be same.

2.3.8 Timeout.hoststatus

When processing the queue, sendmail saves the connection status of each host to which it connects and each host to which it fails to connect. It does this because an unsuccessful host should not be tried again during the same queue run. If the queue is very huge, and the processing takes hour then the likelihood of a previously failed connection to succeed increases. In these case sendmail 8.8 has introduced the Time-out.hoststatus option

O Timeout.hoststatus=interval
Here the interval is of type time. If interval is present, it specifies the length of time before the information about a host will be valid. If the queue run finishes faster than interval, then this has no effect. But when queue runs take longer than this interval, a previously down host will be given a second try if it appears in the queue again. The default value is 30 minutes.

2.3.9 UnsafeGroupWrites

Beginning from sendmail 8.8, the decision of whether or not to trust group write permissions is left to the UnsafeGroupWrites option:

O UnsafeGroupWrites=bool

The optional argument bool, when missing defaults to true (check for unsafe group write permission). If this option is entirely missing. it defaults to false With this option set to true, a ~/.forward file or a :include: file with group or world writability will result in error being logged. And any address in the file or a program will result in a bounce and a message being logged.

2.3.10 SingleLineFromHeader

Lotus notes' SMTP mail gateway can generate From: headers that contain new lines and that contain the address on the second line:

From: Full name < address >

Although this is legal per RFC822, many Mail User Agents (MUA) mishandle such headers and are unable to find this address. To overcome this the SingleLineFromHeader option can be defined.

SingleLineFromHeader=bool
bool is of type Boolean. If it is true sendmail will convert all new lines found in a From:header into space characters. If it is false sendmail will leave all From: headers as is.

2.4 Enhancements to existing Flags/Options

2.4.1 F=equate

The flag specified with F= tell sendmail how the delivery agent will behave and what its needs will be. There were two new values added to this flag.

1.F=9
Sendmail 8.8 has the internal ability to convert messages that were converted into either quoted-printable or base64 back into their original 8 bit form. The decision of whether or not to do this conversion is based on this flag.
2.F=0
This flag is set to turn off MX lookups.

2.4.2 QueueSortOrder

Sorting is done based on the Queue sort order option:

QueueSortOrder=how
Where how is of type character. It can be a P or p, which causes sendmail to sort by priority. It can be a H or h, which causes sendmail to do an enhanced sort. Beginning from V8.8 sendmail, it can be t or T which sorts by submission time.This option is present along with the other options.

2.5 New Macros and routines

2.5.1 ${client_addr}

The ${client_addr} macro is assigned its value when a host connects to the running daemon. The value assigned is the IP address of that connecting host and is the same as the IP address stored in the $_ macro, but without the surrounding square bracketsand other non-IP information.
The ${client_addr} macro can be used in the check_rcpt and check_mail rulesets. It can, for example, be used to select whether an external host is trying to send external mail through an outgoing firewall machine.

2.5.2 ${client_name}

The ${client_name} macro is assigned its value when a host connects to the running daemon. This macro holds as its value, the canonical hostname of that connecting host, which is the same as the hostname stored in the $_ macro.

2.5.3 ${client_port}

The ${client_port} macro is assigned its value when a host connects to the running daemon. This macro holds the port from which the connection is established with the SMTP port. This has the same function as that of the previous macros.

2.5.4 check_relay()

A new config file rule check_relay is introduced to check the incoming connection information. This is similar to the check_compat routine. This has host name and host address separated by $| as its arguments and can reject connections based on it.

2.5.5 validate_connection()

This function is given in conf.c and decides whether to accept traffic from a particular host or not. If this returns false, all SMTP commands will return ``550 Access Denied''.

2.6 New Command line options

2.6.1 -U

This command line flag indicates that this is the initial MUA->MTA submission. The flag currently does nothing, but in the future releases (when MUAs start using these flags) it will turn on things like DNS canonifications, etc.

2.6.2 -bD

The -bD command line switch is almost exactly like the -bd switch. That is it causes sendmail to run as a daemon, but , unlike the -bd switch, it prevents sendmail from performing a fork and there by keeps sendmail in the foreground. This allows sendmail to be run from a wrapper script to detect whether it died or was killed.

2.6.3 -bh

The -bh command-line switch is a synonym for the hoststat command-line. It causes sendmail to print its persistent host status and exist..

2.6.4 -bH

The -bH command-line switch causes sendmail to clear(purge) all the persistent host-status information that was being saved as a result of the HostStatusDirectory option. Not e that the HostStatusDirectory itself is not removed but all the subdirectories under it are. .The purgestat command is synonym for this switch.

2.7 New Named Rule sets

The rapid spread of the internet has led to an increase of mail abuses. Prior to V8.8 sendmail , detecting and rejecting abusive email required you to write C language code for use in the checkcompat() routine. Beginning with V8.8 sendmail important and useful checking and rejecting can be done from within four brand new rulesets:

2.7.1 check_mail

Validate the sender-envelope address given to the SMTP MAIL command.

2.7.2 check_rcpt

Validate the sender-envelope address given to the SMTP RCPT command.

2.7.3 check_relay

Validate the host initiating the SMTP connection.

2.7.4 check_compat

Compare or contrast each envelope sender and envelope recipient pair of addresses just before delivery, and validate based on the result. We will see these rulesets in more detail later in this paper when we take up an example.

2.8 Other features

The ``No ! in UUCP From address!'' message'' is eliminated -- instead, create a virtual UUCP address using either a domain address or the $k macro.

3.0 Using the new check_* rulesets for restricting spamming

To understand these ruleset, we need to understand the steps involved in a SMTP transaction.

3.1 An Introduction to SMTP

This section presents the procedures used in SMTP in several parts. We'll see some basic mail procedure defined as a mail transaction. Throughout this section are examples of partial command and reply sequences

There are three steps to SMTP mail transactions. The transaction is started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction.

The first step in the procedure is the MAIL command. The <reverse-path> contains the source mailbox.

MAIL <SP> FROM:<reverse-path> <CRLF>

This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. It gives the reverse-path which can be used to report errors. If accepted, the receiver-SMTP returns a 250 OK reply.
The <reverse-path> can contain more than just a mailbox. The reverse path> is a reverse source routing list of hosts and source mailbox. The first host in the <reversepath> should be the host sending this command.
The second step in the procedure is the RCPT command.
RCPT <SP> TO:<forward-path> <CRLF>

This command gives a forward-path identifying one recipient. If accepted, the receiver-SMTP returns a 250 OK reply, and stores the forward-path. If the recipient is unknown the receiver-SMTP returns a 550 Failure reply. This second step of the procedure can be repeated any number of times The <forward-path> can contain more than just a mailbox. The <forward-

path> is a source routing list of hosts and the destination mailbox. The first host in the <forward-path> should be the host receiving this command.

The third step in the procedure is the DATA command.

DATA <CRLF>

If accepted, the receiver-SMTP returns a 354 Intermediate reply and considers all succeeding lines to be the message text. When the end of text is received and stored the SMTP-receiver sends a 250 OK reply.

Since the mail data is sent on the transmission channel the end of the mail data must be indicated so that the command and reply dialog can be resumed. SMTP indicates the end of the mail data by sending a line containing only a period. A transparency procedure is used to prevent this from interfering with the user's text

Please note that the mail data includes the memo header items such as Date, Subject, To, Cc, From.
The end of mail data indicator also confirms the mail transaction and tells the receiver-SMTP to now process the stored recipients and mail data. If accepted, the receiver-SMTP returns a 250 OK reply. The DATA command should fail only if the mail transaction was incomplete (for example, no recipients), or if resources are not available.
The above procedure is an example of a mail transaction. These commands must be used only in the order discussed above. Example below illustrates the use of these commands in a mail transaction.
Example of the SMTP Procedure

This SMTP example shows mail sent by Smith at host Alpha.ARPA, to Jones,Green, and Brown at host Beta.ARPA. Here we assume that host Alpha contacts host Beta directly
S: MAIL FROM:<Smith@Alpha.ARPA>

R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>

R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>

R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>

R: 250 OK

S: DATA

R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...

S: ...etc. etc. etc.

S: <CRLF>.<CRLF>

R: 250 OK
The mail has now been accepted for Jones and Brown. Green did not have a mailbox at host Beta.

3.2 check_mail

The MAIL command in the SMTP dialog is used to specify the envelope-sender address:
MAIL From: <sender@host.domain>

If the check_mail rule set exists, it is called immediately after the MAIL command is read. The work passed to check_mail is the address following the colon in the MAIL command. The envelope sender address may or may not be surrounded by
angular braces.
The address supplied through the MAIL command can be checked against the check_mail ruleset. So you can use this to prevent known spammers from sending you e-mail. First, you may have a list of Domains which you want to ban completely. You may have this in an external file:

```
F{SpamDomains} /etc/mail/SpamDomains
```

e.g., cyberpromo.com quantcom.com
Next, you may have a list of users which you want to ban too:

```
F{Spammer} /etc/mail/Spammer
```

e.g.,
mailer@aol.com
Now you can use this as follows:

```
Scheck_mail

R<$={Spammer}>             $#error $@ 5.7.1 $: ``571 We don't accept junk mail''
R<$={Spammer}.>            $#error $@ 5.7.1 $: ``571 We don't accept junk mail''
R$*                          $: $>3 $1
R$*<@$={SpamDomains}.>$*   $#error $@ 5.7.1 $: ``571 We don't accept junk mail
from your domain''
R$*<@$={SpamDomains}>$*    $#error $@ 5.7.1 $: ``571 We don't accept junk
mail from your domain''
```

In addition, you may want to act on broken mailers which don't use quotes around
addresses:

```
R$={Spammer}               $#error $@ 5.7.1 $: ``571 We don't accept junk mail''

R$={Spammer}.              $#error $@ 5.7.1 $: ``571 We don't accept junk mail''
```

If you want to stop receiving mails from subdomains of well known spammers,
you can modify the last two rules a bit:

```
R$*<@$*$={SpamDomains}.>$*  $#error $@ 5.7.1 $: ``571 We don't accept junk
mail from your domain''
R$*<@$*$={SpamDomains}>$*  $#error $@ 5.7.1 $: ``571 We don't accept junk
mail
from your domain''
```

Next step could be the following: you want also to reject mail from those domains,
which are not registered with DNS. However, this may also be a temporary fault, so you should give back a temporary failure.

```
# if you enable the last rule, you can disable this one. # host without a . in
the Fully Qualified Host Name ?
R$*<@$->$*      $#error $@ 4.1.8 $: ``418 invalid host name'' no real name
# lookup IP address (reverse mapping available?)
# R$*<@[$-.$-.$-.$-]>$* $: $1  < @ $[ [ $2.$3.$4.$5 ] $] > $6


# no DNS entry? this is dangerous!
# R$*<@$*$~P>$* $#error $@ 4.1.8 $: ``418 unresolvable host name, check your
configuration.''                               no real name
```

3.3 check_rcpt

The RCPT command in the SMTP dialogue specifies an envelope recipient's address:

RCPT To: <recipient@host.domain>

If the check_rcpt rule set exists, it is called immediately after the RCPT command is read. The workspace that is passed to check_rcpt is the address following the colon. The envelope recipient address may or may not be surrounded by angle brackets and may or may not have other RFC822 associated with it.

The address supplied through the RCPT command can be checked against the check_rcpt ruleset. On first look, this ruleset doesn't make much sense. Why check the recipient? sendmail does this anyway when trying to deliver, esp. for local recipients. However, this ruleset can be used to check whether your system is (mis)used as a gateway. The check_compat ruleset, which seems to be better suited for this purpose, since it gets both addresses (sender and recipient) as parameters, is called too late. To reject a misuse at the earliest moment (and save your bandwidth etc.), you can refer to the address of the sending system, which is available in the macro ${client_addr} . However, to use it in a rule, you have to refer to it as: $(dequote ```` $&{client_addr} $) so sendmail defers evaluation and tokenizes it. But since there is a problem with these rules, here is a new solution. First, we check whether it is a local client: it can do whatever it want. Next, we remove the local part, maybe repeatedly. If it still has routing information in it, it seems to be a relay attempt. So list in the class

```
 F{LocalIP} /etc/mail/LocalIP
```

the IP addresses of the hosts you will allow to relay through your mail

server, for example

```
134.245

127.0.0.1

Scheck_rcpt # first: get client addr

R$+       $: $(dequote ```` $&{client_addr} $) $| $1

R0 $| $*    $@ ok                                  no client addr: directly invoked

R$={LocalIP}$* $| $*                  $@ ok                  from here
# not local, check rcpt

R$* $| $*    $: $>3 $2 # remove local part, maybe repeatedly R$*<@$=w.>$*
$>3 $1 $3
# still something left?
R$*<@$+>$*         $#error $@ 5.7.1 $: 571 we do not relay
```

The trailing $* after $={locally} matches incompletely specified IP addresses on octet boundaries, as can be seen by 134.245 which matches a whole class B sublet.

If you relay mail for other systems, use also:

```
F{realty} /etc/mail/RelayTo
```

to list all hosts you relay mail to or accept mail for. For example, we put uni-kiel.de in RelayTo . Then change the line

```
R$*<@$=w.>$*     $>3 $1 $3
```

to

```
R$*<@$*$={RelayTo}.>$*         $>3 $1 $4
```

(or just add the latter).

3.4 check_relay

Sendmail 8.8 supports a mechanism for screening incoming SMTP connections.

The check_relay ruleset is used to screen incoming network connections and accept or reject them based on hostname, domain or IP number. check_relay gets the host name and host address of the client separated by $| as parameters. An example is

```
F{DeniedIP} /etc/mail/DeniedIP F{DeniedNames} /etc/mail/DeniedNames
```
where these files contain a list of IP addresses and hostnames which are not allowed to access your mailserver.

```
Scheck_relay
R$+ $| $={DeniedIP}$*                    $#error $@ 5.7.1 $: ``no access From your IP address''
R$*$={DeniedNames} $| $*       $#error $@ 5.7.1 $: ``no access from your host''
```

(note the trailing/leading $* to match with incompletely specified IP addresses/names).

Access will be refused with the error message:

550 Access denied
and the error string will be logged.

3.5 check_compat

Although check_compat gets both addresses (sender and recipient) as parameters to check whether your machine is used as a gateway, it's too late. check_compat is called after the whole message has been transmitted. You could do something like this:

```
Scheck_compat
R$+ $| $+                    $: $2 $| $>3 $1 canonicalize sender
R$+ $| $+                    $: $2 $| $>3 $1 canonicalize recipient

 R$- $| $+                    $@ok          from here

R$+ $| $-                    $@ok          to here

R$+<@$=w.> $| $+             $@ok          from here
R$+ $| $*<@$=w.>             $@ok          to here

R$*                           $#error $@ 5.7.1 $: ``571 we do not support relaying''
```

to prevent (mis)use of your machine as a mail gateway by other people. May be you have to use some other class than w . However, this ruleset has a problem with forwarding. That's one of the reasons why you should use the check_rcpt solution .

4.0 Conclusions

The sendmail has got lot of new features which increase its performance and has made it more secure . This version is much superior to the previous versions of sendmail.

5.0 Acknowledgments

The author would like to thank his management for giving him the opportunity to present this paper. The author would also like to thank Abhijit Khot, Arundhati, Madhu Thiyakkat, Lori Dutra and Beena Pillai for their valuable assistance.

6.0 References

- ``sendmail''by Bryan Costales with Eric Allman and Neil Rickert
- ``Sendmail Installation and Operation Guide''by Eric Allman
- ``Installing and Administering Internet Services'' HP manual
- RFC821 SMTP protocol
- RFC822 Mail header format
- RFC974 MX routing
- RFC1425 SMTP Service Extensions
- RFC1426 SMTP Service Extension for 8bit-MIMEtransport
- RFC1521 MIME: Multipurpose Internet Mail Extensions
- RFC1985 SMTP Service Extension for Remote Message Queue Starting