

MuntsOS Embedded Linux

Application Note #16: Sending Email

**Revision 3
22 March 2025**

**by Philip Munts
dba Munts Technologies
<http://tech.munts.com>**

Introduction

This application note describes how send email messages from a **MuntsOS Embedded Linux** (hereafter just **MuntsOS**) target computer.

Prerequisites

MuntsOS must be installed on the target computer ([AppNote #3](#) or [AppNote #15](#)).

Test Platform Hardware

The test platform for the purposes of this application note consists of any **MuntsOS** target board with Internet connectivity.

Mail Transfer Agent (MTA)

A **Mail Transfer Agent** is a system program that accepts an email message text conforming to [RFC5322](#) and then dispatches it to one or more recipients.

Examples of full service MTA's include [Sendmail](#) and [Postfix](#). Full service MTA's are often large and difficult to configure and may not suitable for small embedded systems.

Examples of lightweight MTA's more suitable for small embedded systems include [DragonFly Mail Agent](#), and [E-MailRelay](#).

By tradition every MTA for Unix systems provides a command line program named `/usr/sbin/sendmail` that reads an email message text conforming to RFC5322 from standard input and then dispatches the message to its recipient(s).

Given a file `message.txt` containing an email message text conforming to RFC5322, the following command sends an email:

```
/usr/sbin/sendmail -t <message.txt
```

The `sendmail` command included in **MuntsOS** is provided by BusyBox. It is not really an MTA *per se*, as it just dispatches emails to an SMTP (**Simple Mail Transfer Protocol**) server listening on `localhost:25` that must do all the work of dispatching emails to recipients.

MuntsOS does not include an SMTP server, so in order to send email from a **MuntsOS** target computer, you must configure and/or install either an MTA or an SMTP server or both. Three options are described later in this document.

Junk mail countermeasures have made it increasingly difficult for an [Internet of Things](#) device like a **MuntsOS** target computer to send email successfully, even with a robust MTA. You will most likely need to dispatch emails to an authenticating intermediate SMTP relay (*aka* smart host) instead of directly to the recipient's domain SMTP server(s).

Mail User Agent (MUA)

A **Mail User Agent** is a user program that generates an email message text conforming to RFC5322 and then passes it to an MTA for dispatching. On Unix and Linux, the MUA often runs `/usr/sbin/sendmail` using the `popen()` Standard I/O Library function and writes the generated email message text to the file object returned by `popen()`.

MuntsOS includes the MUA `/usr/bin/mail` from the [GNU Mailutils](#) package. The `mail` program reads a message payload from standard input, generates an email message text conforming to RFC5322, and then passes the result to `/usr/sbin/sendmail` for dispatching.

The following command sends an email with the subject `Test1` to recipient `you@me.com`:

```
echo "This is a test" | mail -s "Test1" you@me.com
```

An application program can run `/usr/bin/mail` with `popen()` to send an email message.

An application program can also act as its *own* MUA, dispatching RFC5322 conforming email message texts to `/usr/sbin/sendmail` via `popen()` or to an SMTP server via TCP at `localhost:25`. The [Ada Web Server](#) library provides email client services for the Ada programming language. Similarly, the [System.Net.Mail](#) namespace provides email client services for .Net Core programs.

E-MailRelay

The `emailrelay` extension package provides an SMTP server at `localhost:25`. It does not replace `/usr/sbin/sendmail`.

E-MailRelay must be configured by editing two configuration files:

`/usr/local/etc/emailrelay/auth.conf` -- Replace `username` and `password` with the login credentials for your SMTP relay.

`/usr/local/etc/emailrelay/emailrelay.conf` -- Replace "`servername:port`" with your SMTP relay settings.

DragonFly Mail Agent

The `dma` extension package replaces `/usr/sbin/sendmail`. It does not provide an SMTP server at `localhost:25`, making it somewhat less flexible than `emailrelay`.

DragonFly Mail Agent must be configured by editing two configuration files:

`/usr/local/etc/dma/auth.conf` -- Replace `username`, `smarthost`, and `password` with the login credentials for your SMTP relay.

`/usr/local/etc/dma/dma.conf` -- Replace the values for `SMARTHOST`, `PORT`, and `MASQUERADE` with values specific to your SMTP relay settings.

On Demand SSH Tunnel to a Remote Computer

You can configure a **MuntsOS** target computer to SSH `mailtunnel@foo.bar` whenever `sendmail` or another process opens a TCP connection to `localhost:25`.

If you have administrative access to some other Unix (FreeBSD, OpenBSD, Linux, etc.) computer `foo.bar` that runs an SMTP server listening on *its* `localhost:25`, that is permitted to send email, and that you can log into with `ssh`, then you can create a user `mailtunnel` on `foo.bar` that connects to `foo.bar localhost:25` whenever you SSH `mailtunnel@foo.bar`.

Now when `sendmail` on the target computer connects to `localhost:25`, it actually connects to `foo.bar localhost:25` by way of a temporary SSH tunnel. When `sendmail` exits, the SSH tunnel is torn down.

Target Computer Setup

1. Create or modify `/etc/inetd.conf` and `/etc/ssh/ssh_known_hosts` with the following commands, replacing `foo.bar` with the domain name of your remote computer:

```
cat <<EOD >>/etc/inetd.conf
# Mail relay over SSH tunnel
127.0.0.1:25 stream tcp nowait root /usr/bin/ssh -q -T mailtunnel@foo.bar
EOD
ssh-keyscan foo.bar >>/etc/ssh/ssh_known_hosts
```

2. Create `.ssh/id_rsa` and `.ssh/id_rsa.pub` using `sysconfig` option **Regenerate superuser id_rsa**.
3. Enable `inetd` by setting bit 10 in the `OPTIONS` word in `/boot/cmdline.txt` (Raspberry Pi) or `/boot.config.txt` (Orange Pi Zero 2W or BeaglePlay) using `sysconfig` option **Edit cmdline.txt** or **Edit config.txt**.
4. Copy `.ssh/id_rsa.pub` to the remote computer superuser and then reboot.

Remote Computer Setup

1. Become super user with `sudo su -`
2. Modify `/etc/ssh/sshd_config`:

```
cat <<EOD >>/etc/ssh/sshd_config

Match User mailtunnel
    AllowTcpForwarding no
    ForceCommand ncat -4 127.0.0.1 25
EOD

killall -HUP /usr/sbin/sshd
```

3. Create user **mailtunnel** with commands similar to the following:

```
groupadd -g 997 mailtunnel
useradd -c "Mail Tunnel" -m -g 997 -u 997 -s /bin/sh mailtunnel
rm -rf /home/mailtunnel/.*
```

4. Create **/home/mailtunnel/.ssh/authorized_keys** on the remote computer, using commands similar to the following:

```
sudo su -
mkdir -p /home/mailtunnel/.ssh
cat id_rsa.pub >>/home/mailtunnel/.ssh/authorized_keys
chown -R mailtunnel.mailtunnel /home/mailtunnel
chmod 444 /home/mailtunnel/.ssh/authorized_keys
chmod 500 /home/mailtunnel/.ssh
chmod 500 /home/mailtunnel
```

Testing

1. Try to log in from the **MuntsOS** target computer to the remote computer:

```
ssh mailtunnel@foo.bar
quit
```

You should get responses similar to the following:

```
220 bethel.munts.net ESMTP OpenSMTPD
221 2.0.0 Bye
```

2. Try to send yourself an email with a command similar to the following:

```
echo "This is a test" | mail -s Test1 me@you.com
```