# Sending Mail the Net::SMTP Way

Brad Marshall <brad.marshall@member.sage-au.org.au>

December 13, 2003

## 1 Introduction

Perl is a very common language for writing administrative scripts in, and as such, it is useful to be able to send email reports using it. Additionally you may not want to have to run a mail server on each and every server you have and it would be nice to be able to generate emails from these boxes using SMTP. Perl's answer for this is the Net::SMTP module, and this article will cover basic usage on sending emails using it.

## 2 Sending Email using Net::SMTP

### 2.1 Modules

First off, you need to import the required modules into your script, as follows:

```
use Net::SMTP;
```

Net::SMTP is included with the libnet series of perl modules, which includes modules for talking FTP, DNS, POP3 and a range of others. Later versions of perl (5.8.0 and above) include it by default, otherwise you can get it from your local CPAN mirror.

### 2.2 Opening Connection

The next step is to open a connection to your desired mailhost. This is done via the following and returns a Net::SMTP object.

```
$smtp = Net::SMTP->new("mailhost.example.com",
                    Hello => 'my.host.com',
                    Timeout => 60);
```

The first argument is obviously the host that you wish to connect to, and is the only required option. The next one is the string to include in the HELO command - if it is not specified it will take a good guess based on hostname information. The timeout is how long to wait on the connection to the mailhost - it defaults to 120. There are many more options you can pass, for more information see the Net::SMTP manpage.

### 2.3 Defining Addresses

Defining the addresses that you are sending from and to is simply done via:

```
$smtp->mail("from@example.com");
$smtp->recipient("to@domain.com", \
     "user@example2.com");
```

This simply sets the MAIL FROM in the SMTP conversation to from@example.com, and the RCPT TO to the listed addresses.

You can also use the following commands to set the receipients:

```
$smtp->to("user1@domain.com");
$smtp->cc("foo@example.com");
$smtp->bcc("bar@blah.net");
```

### 2.4 Building the Message

To actually start building the message, you use the following command. This is equivalent to DATA in the SMTP conversation.

```
$smtp->data;
```

The next set of commands are the headers that are used in the email message. Note that the from and to headers don't need to match the headers given earlier in the SMTP conversation.

```
$smtp->datasend("From: me\@example.com");
$smtp->datasend("To: to\@domain.com");
$smtp->datasend("Subject: This is a test");
$smtp->datasend("\n");
```

The blank line signifys the end of the headers, and the start of the body, which you fill out using the following command:

```
$smtp->datasend("blahblah");
```

To finish sending data and close the connection is simply a matter of:

```
$smtp->dataend;
$smtp->quit;
```

## 3 Other Neat Stuff with Net::SMTP

### 3.1 Return Receipts

Sometimes it is useful to get notification of when someone reads an email, assuming their mail user agent (or MUA) supports return receipts. To do this, put the following in the header section of the above script:

```
$smtp->datasend("Disposition-Notification-To: \
          to.me\@domain.com");
```

When a user reads their mail with a MUA that understands return receipts, a message will be sent back to the specified address displaying what time the message was read.

To set something at a high urgency, you simply insert the following into the headers:

```
$smtp->datasend("Priority: Urgent\n");
$smtp->datasend("Importance: high\n");
```

What this actually does will depend on the MUA at the other end, but there should be some indication that this is at a high priority.

## 3.2 Verifying Addresses

It is useful to check if email address are valid occasionally - unfortunately, the EXPN and VRFY SMTP commands are often turned off due to (very valid) spam concerns. However, you may still have reason and a server that supports them - to utilize this, do the following:

```
$expn = $smtp->expand($address);
$vrfy = $smtp->verify($address);
```

Expand and verify will return null if the address doesn't exist. Expand will return the expanded address if it does exist, and verify will return one.

For domains without a full time internet connection, mail is often spooled at a host ISP. It is common to use the ETRN command to tell the host server that the machine is up and ready to accept mail - the way to do this in Perl is:

```
$smtp->etrn($domain);
```

## 3.3 Getting Information

To pull the banner information from the mailhost that a connection is open to, the following is used:

```
$banner = $smtp->banner;
```

To find out what hostname and domain it returns, do the following:

```
$domain = $smtp->domain;
```

To get generic help, just set the subject you want, then call the help routine, like so:

```
$subject = "RCPT";
$help = $smtp->help($subject);
```

# 4 Conclusion

As you have seen, it is simple to send emails using Perl - this can be used in many situations in scripts that are used for systems administration for notification of events. It is also very easy to set return receipts and urgency to ensure that emails are read.