2-1-2010

A Practical Study of E-mail Communication through SMTP

M. Tariq Banday
University of Kashmir, sgrmtb@yahoo.com

Jameel A. Qadri
British Institute of Technology & Ecommerce, scorpiojameel@yahoo.com

Nisar A. Shah
University of Kashmir, nassgr@yahoo.com

芽|Sprouts Working Papers on Information Systems ISSN 1535-6078

A Practical Study of E-mail Communication through SMTP

M. Tariq Banday
University of Kashmir, India

Jameel A. Qadri
British Institute of Technology & Ecommerce, UK

Nisar A. Shah
University of Kashmir, India

Abstract
Simple Mail Transfer Protocol (SMTP) is an application layer protocol for e-mail communication. It has been adopted as a standard by Internet Engineering Task Force (IETF). SMTP has set conversational and grammatical rules for exchanging messages between connected computers. It has evolved through several revisions and extensions since its formation by Jon Postel in 1981. In SMTP, the sender establishes a full-duplex transmission channel with a receiver. The receiver may be either the ultimate destination or an intermediate forwarding agent. SMTP commands are issued by the sender and are sent to the receiver, which responds to these commands through codes. Each SMTP session between the sender and the receiver consists of three phases namely: connection establishment, mail transactions and connection termination. This paper describes and illustrates the process of e-mail communication through SMTP by issuing the individual SMTP commands directly to transmit e-mail messages. It also describes individual SMTP commands and extensions with practical implementation using a Telnet client.

Keywords: E-mail Communication, SMTP, ESMTP, SMTP Commands, SMTP Reply Codes, SMTP Extensions, SMTP using Telnet, SMTP Authentication.

Permanent URL: http://sprouts.aisnet.org/10-20

Reference: Banday M.T., Qadri J.A., Shah N.A. (2010). "A Practical Study of E-mail Communication through SMTP ," . *Sprouts: Working Papers on Information Systems*, 10(20). http://sprouts.aisnet.org/10-20

Introduction

An E-mail message consists of e-mail Body and e-mail Header. The Body is text which can also include multimedia elements in Hyper Text Markup Language (HTML) and attachments encoded in Multi-Purpose Internet Mail Extensions (MIME) [01]. The Header is a structured set of fields that include 'From', 'To', 'Subject', 'Date', 'CC', 'BCC', 'Return-To', etc. Headers are included in the message by the sender or by a component of the e-mail system. TCP/IP e-mail address consists of username and domain name separated by @ sign e.g. aliace@a.com. Ray Tomlinson [02] first initiated the use of @ sign to separate username from the domain name. An e-mail communication between a sender 'Alice' having e-mail address 'alice@a.com' and recipient 'Bob' having e-mail address 'bob@b.com' is shown in figure 1 below.

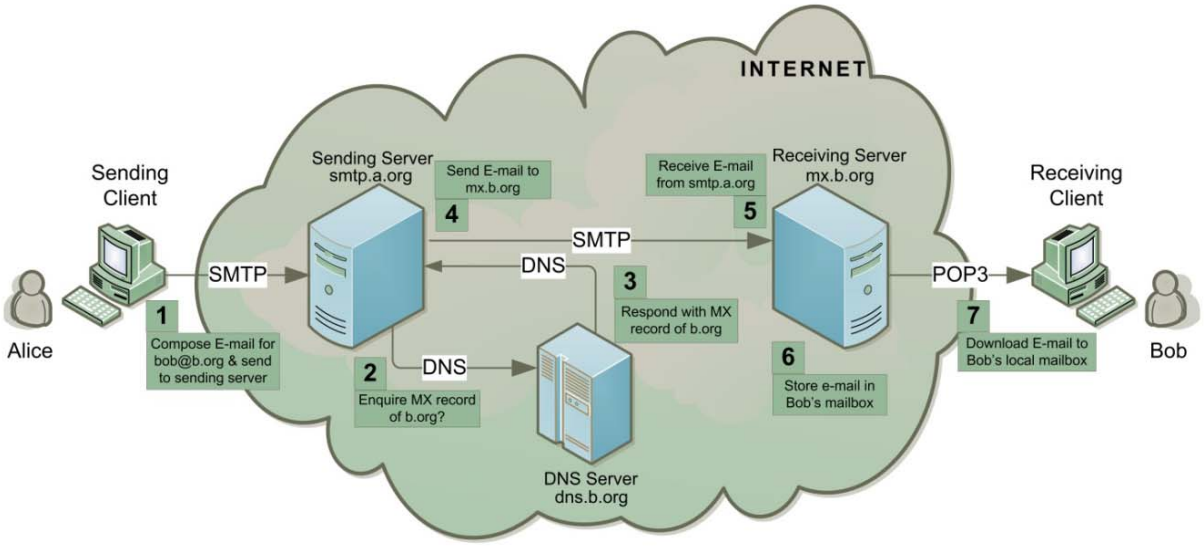

Figure 1: E-mail communication between a sender 'Alice' and recipient 'Bob'

'Alice' composes an e-mail message on her computer called client for 'Bob' and sends it to her sending server 'smtp.a.org' using Simple Mail Transfer Protocol (SMTP) [03, 4]. Sending server performs a lookup for the mail exchange record of receiving server 'b.org' through DNS protocol on DNS server 'dns.b.org'. The DNS server responds with the highest priority mail exchange server 'mx.b.org' for the domain 'b.org'. Sending server establishes SMTP connection with the receiving server and delivers the e-mail message to the mailbox of 'Bob' on the receiving server. 'Bob' downloads the message from his mailbox on receiving server to local mailbox on his client computer using POP3 or IMAP protocols. Optionally, 'Bob' can also read the message stored in his server mailbox without downloading it to the local mailbox by using some Webmail program. This model of electronic communication involves a number of hardware and software components that communicate with each other using some protocols especially SMTP protocol. SMTP protocol has evolved as a complex system since its inception. Its commands have been augmented by inclusion of various extensions which may or may not be adopted by every SMTP client and server.

The remaining paper is organized as follows: Section 2 describes various hardware and software components of e-mail system. Section 3 describes SMTP connection, mail transmission and termination processes. Section 4 presents and demonstrates various SMTP commands and extensions. Section 5 briefly presents SMTP reply codes followed by conclusion.

E-mail Infrastructure and Protocols

E-mail infrastructure comprises of various hardware and software components. It includes sender's client and server computers and receiver's client and server computers with required software and services installed on each. Besides these, it uses various systems and services of Internet. The sending and receiving servers are always connected to the Internet but the sender's and receiver's clients connect to the Internet as and when required. At each component several communicating entities called e-mail nodes are involved in the process of e-mail delivery. The directed graph model of Internet e-mail infrastructure [5] shown in figure 2 can be used to study the e-mail infrastructure and protocols involved in e-mail creation, transmission and delivery process. The vertices in this model represent e-mail infrastructural elements and each edge corresponds to the possible e-mail path and protocol.

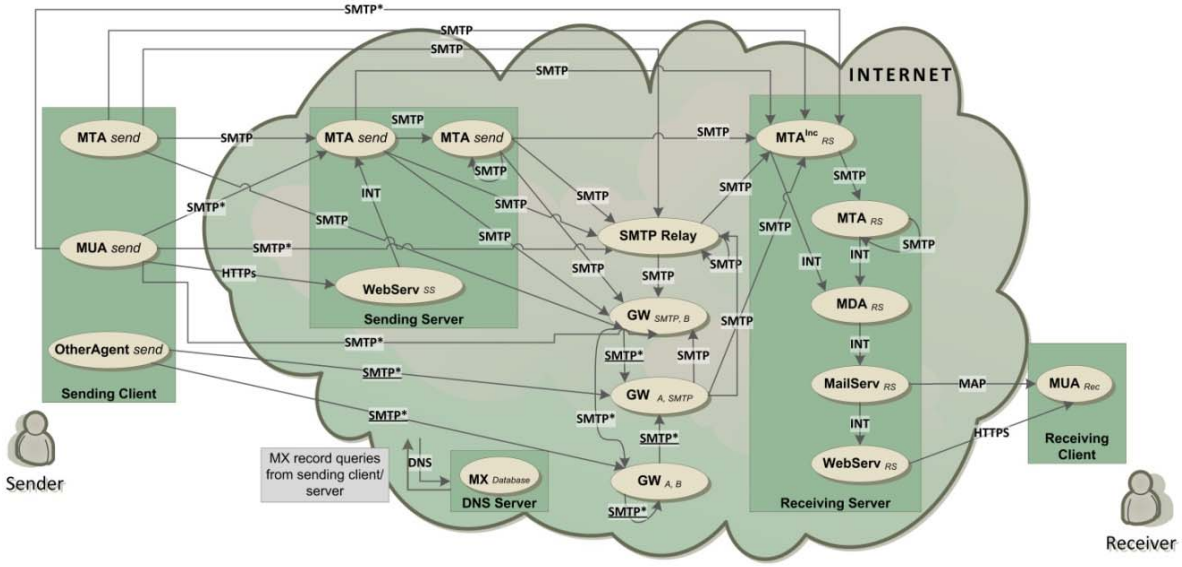

Figure 2: Directed Graph Model of E-mail Infrastructure

Each vertex corresponds to an e-mail node which is essentially a software unit involved in e-mail communication process and works on application layer of TCP/IP model. Nodes working on lower layers such as routers and bridges represent options to send e-mail without using SMTP are not considered in this model as almost all e-mail communication uses SMTP directly or indirectly. Further, proprietary nodes used for internal deliveries at sending and receiving servers are also not considered in this model. All Mail User Agent (***MUA)*** nodes are software packages that run on client computers and allow end users to compose, create or read e-mail. Some ***MUAs*** may be used to send e-mail to the receiving ***MTAs*** directly or indirectly. 'Microsoft Outlook', 'Microsoft Outlook Express', 'Lotus Notes', 'Netscape communicator', 'Qualcomm Eudora', 'KDE KMail', 'Apple Mail', and 'Mozilla Thunderbird' are examples of ***MUAs***. Two or more ***MTAs*** can be used at the sending servers to make e-mail delivery. Several Web-based e-mail programs and services (known as Webmail) such as 'AIM Mail', 'Yahoo Mail', 'Gmail', and 'Hotmail' which integrate e-mail clients and servers behind a Web server are also used as ***MUAs.*** Mail transfer Agent ***(MTA)*** nodes are in effect postal sorting agents that have the responsibility of retrieving the relevant Mail eXchange (***MX***) record from the Domain Name Servers (***DNS***) [6] for each e-mail to be send and thus map the distinct e-mail addressee's domain name with the relevant IP address information. ***DNS*** is a distributed directory database that correlated domain

names to IP addresses. ***MTAs*** can also be used to compose and create e-mail messages. 'Sendmail', 'Postfix', 'Exim', and 'Exchange Server', are examples of ***MTAs***. A receiving MTA can also perform the operation of delivering e-mail message to the respective mailbox of the receiver on the mail server and thus is also called Mail Delivery Agent (***MDA***). Node named ***OtherAgents*** are software packages that send e-mail message through gateways. ***WebServ*** nodes are the e-mail Web servers that provide the Web environment to compose, send and read an e-mail message. SMTP-Relays [7] are the nodes that perform e-mail relaying. Relaying is the process of receiving e-mail message from one SMTP e-mail node and forward it to another one. Gateway nodes are used to convert e-mail messages from one application layer protocol to other. Gateway nodes named $GW_{SMTP,\ B}$ accept SMTP protocol based e-mails and transfer them with protocols other that SMTP and $GW_{A,\ SMTP}$ performs the inverse process at incoming and outgoing interfaces. Gateway nodes $GW_{A,B}$ do not use SMTP either for incoming or outgoing interfaces. A process called Proxy may be done at these nodes when incoming and outgoing interfaces use same protocols. ***MailServ*** node represent e-mail server providing users mail access service using IMAP or POP3 protocols. It also provides an internal interface to a Web server for HTTP based e-mail access.

The e-mail nodes establish connections with one or more nodes. Each edge of the graph connecting two e-mail nodes represents possible e-mail flow between them using a particular set of protocols. Table 1, lists basic protocols used in e-mail flow between two possible e-mail nodes.

Table 1: E-mail Communication Protocols

Protocol Group	Basic Protocols
SMTP	*SMTP protocol (RFC 821), SMTP service extension protocols ESMTP including Service Extension for Authentication (RFC 2554), Delivery by SMTP Service Extension (RFC 2852), SMTP Service Extension for Routing Enhanced error (RFC 2034), and SMTP Service Extension for Secure SMTP over Transport Layer Security (RFC 3207).*
SMTP*	*All protocols in SMTP group and all SMTP extensions for e-mail submission from MUA to e-mail node with SMTP incoming interface. E-mail node can be MTA defined in RFC 2821, MSA defined in RFC 2476. Using MSA various methods can be applied for ensuring authenticating user that include IP address restrictions, secure IP and POP authentication.*
SMTP*	*All Internet application protocols except those specified in SMTP* group, all propraitory application protocols used on the Internet (also used for tunneling), all Internet protocols on the transport and network layers such as TCP/IP as it is possible to send e-mail without the use of application layer protocols.*
HTTP(S)	*HTTP (RFC 2616), HTTP over SSL and HTTP over TLS (RFC 2818).*
INT	*ESP specific protocols and procedures for internal e-mail delivery between e-mail nodes.*
MAP	*All e-mail access protocols used to transfer e-mails from the recipient e-mail server to MUA that include IMAP version 4 (RFC 1730), MAPI and POP version 3 (RFC 1939).*

Some recent standard or experimental extensions to SMTP are extensions pertaining to: support for diverse service environments [8], international delivery status and deposition notifications [9], internationalized e-mail

address [10], submission service extension for future message release [11], content conversion [12] and message tracking [13],

SMTP Connection, Mail Transaction and Termination Process

Each SMTP session between SMTP sender and SMTP receiver consists of three phases namely: connection establishment, mail transactions and connection termination. In the first phase, a session is established through the creation of a TCP connection. In this phase identification information is exchanged between the sender and the receiver using the HELO or EHLO command. In the second phase mail transactions are performed to transfer the mail from the sender to the receiver. After completing the mail transactions, the third phase begins wherein the SMTP sender uses QUIT command to terminate the SMTP session. This section describes phases involved in SMTP mail transfer.

SMTP uses TCP protocol to make mail transfer reliable and efficient. In the first phase, the sender also called client makes a TCP connection with the receiver on an ephemeral TCP port. The receiver also called the server sends connection acceptance reply using a code (220). The reply also includes server information including full server name and the version of the SMTP server software.

The client on receiving the connection ready reply issues HELO command or EHLO in case of ESMTP which also includes the domain name of the client. The SMTP server after receiving the HELO command, responses with service code (250) along with its supported ESMTP extensions. In case, the receiver does not support extensions, it replies with a service code 500. These steps are illustrated in figure 3 below.

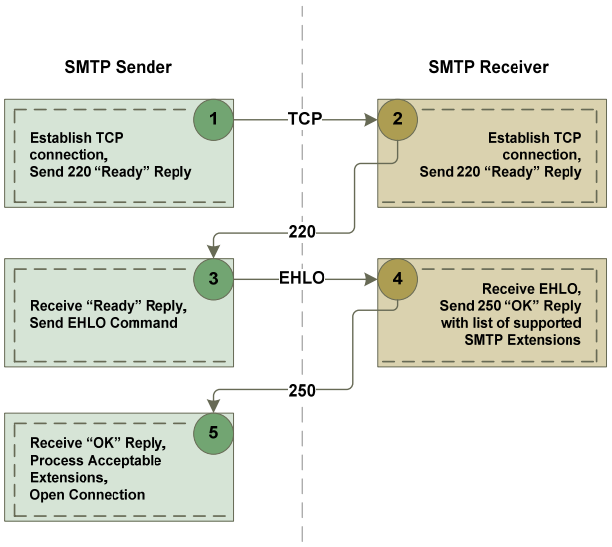

Figure 3: Connection Establishment Process

In the second phase of SMTP session, mail transfer is performed. It involves SMTP commands for sender identification, recipient identification and then mail transfer. The steps involved in this phase are shown in figure 4.

This phase begins with transmission of mail envelope information using MAIL and RCPT TO commands. The MAIL command which includes the sender identification is issued by the sender. The receiver responses with a go ahead service code (250). The receiver may validate the sender and also may reject e-mail reception for security reasons. On receiving the service code 250, the sender specifies recipients using one or more RCPT TO commands. Again the server responds with a go ahead service code (250) or may reject the e-mail reception. After finishing with the envelope transmission, mail is transferred through several transactions using DATA command. The end of the transmission is indicated by transmitting a "." through DATA command. The server stores the e-mail in the mailbox and issues a service reply code (250). The mail transaction is terminated by the sender and the receiver.

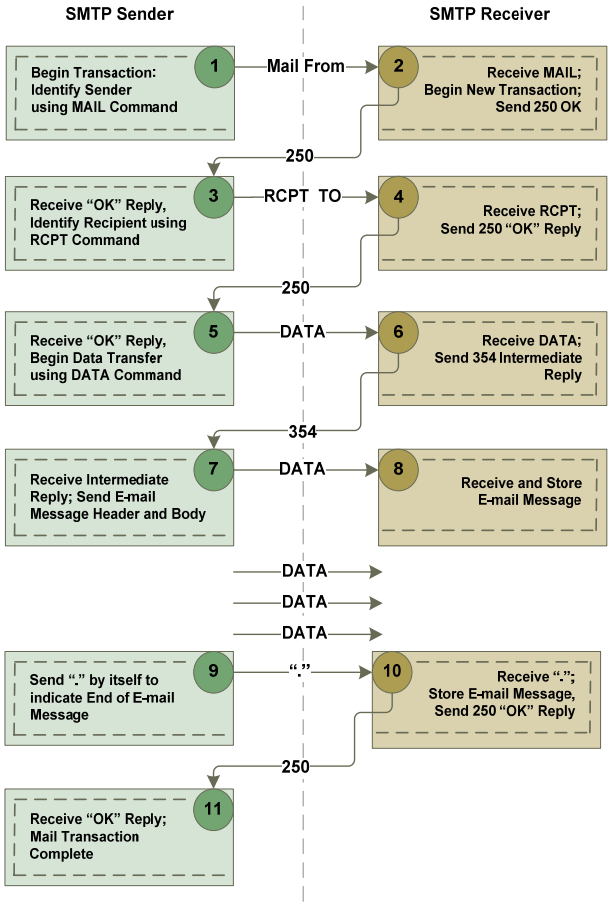

Figure 4: Mail Transaction Process

The sender issues a QUIT command to terminate the session after completing mail transactions as shown in figure 5. The receiver on receiving the QUIT command, issue a service code (221) indicating successful connection termination.

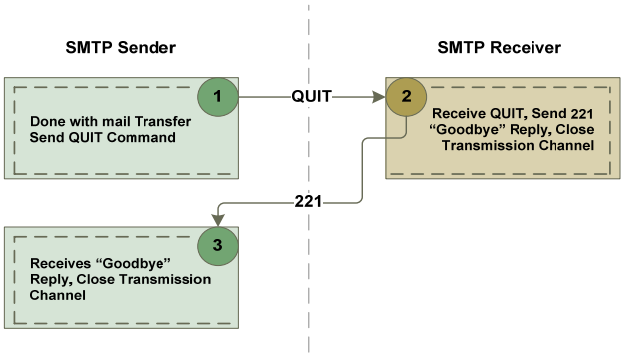

Figure 5: Connection Termination Process

SMTP Commands and Extensions

The SMTP commands (RFC 2821) define the mail transfer or the mail system function requested by the user. SMTP commands are character strings terminated by <SP> if parameters follow or <CRLF> otherwise. The basic syntax of a command is: <Command-code> <parameters>. A mail transaction involves several data objects which are communicated as arguments to different commands. These data objects are transmitted and are held pending until the confirmation is communicated by the end of mail data indication which finalizes the mail transaction. Distinct buffers are provided to hold different types of data objects. Specific commands cause information to be appended to a specific buffer, or cause one or more buffers to be cleared. Several commands require parameters to be specified. Many extensions to the basic operation of SMTP were defined. These are enabled when two SMTP servers supporting the extension set up a session using the EHLO command and appropriate extension response codes. This section briefly describes and demonstrates various SMTP commands and extensions. Telnet Protocol has been used to connect directly to SMTP servers and test SMTP commands (See figure 6). In the SMTP communication examples listed in this paper, the letters ***C*** and ***S*** are used to refer to the commands issued to ***client*** (***sender***) and responses send by the ***server*** (***receiver***) respectively.

HELO and EHLO

The client sends HELO or EHLO command to the SMTP server to identify itself and to initiate the SMTP conversation. The fully-qualified domain name of the SMTP client is sent as an argument to these commands. These commands, and a "250 OK" reply to one of them, confirm that both the SMTP client and the SMTP server are in the initial state, that is, there is no transaction in progress and all state tables and buffers are cleared. The response to EHLO is multiline each containing a

keyword and, optionally, one or more parameters. The syntax of these commands is **HELO** DomainName and **EHLO** DomainName.

Example 1 (HELO)*:*
C: *HELO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*

Example 2(EHLO)*:*
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S:*250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*

a) Use ***NSLOOKUP*** on computer running Windows OS to determine mail server settings.
Open Command Prompt (Type ***CMD***)
C:\>**NSLOOKUP**
SET Q=MX
RECEIVINGDOMAIN.COM
The mail server settings of **RECEIVINGDOMAIN.COM** will be displayed.

b) Use ***TELNET*** to establish TCP connection with mail server.
Open Command Prompt (Type ***CMD)***
C:\>**TELNET**
C:\>**O RECEIVINGDOMAIN.COM 25**
This will establish a telnet session between the sender (***MYDOMAIN.COM***) and the receiver (***RECEIVINGDOMAIN.COM*****)** on port 25. Connections can be opened on other ports e.g. port 587, 995, etc. if the server supports communication on these ports.

c) Once connection is established, SMTP commands can be issued to establish SMTP session and send mail as shown in the below example.
HELO MYDOMAIN.COM
250- mailboxXXXX.mailhostingXXXX.com **….**
MAIL FROM: alice@a.com
250 2.1.0 OK
RCPT TO: bob@b.com
250 2.1.5 OK
DATA
354 End Data with <CR><LF>.<CR><LF>
Subject: Test Message
Date: 10-01-2010
This is my message body.
.
250 2.0.0 Ok: queued as 85313139007B
QUIT
221 2.0.0 Bye

d) To work with some commands like STARTTLS, ***TELNET*** has to be connected with Transport Layer Security (**TLS)** System. This may be achieved by using some telnet program that supports TLS or by setting up some listener that establishes a secure connection between the SMTP server and the client.

Figure 6: Illustration of the use of Telnet to Establish SMTP Session between Sender and Receiver.

MAIL FROM

This command is used to initiate a mail transaction in which the mail data is delivered to an SMTP server which may, in turn, deliver it to one or more mailboxes or pass it on to another system. This command includes the reverse path as its argument. This is the name of the

sender, but it also can be a list of hosts that were used to relay the mail message from its original Sender-SMTP. In a list of hosts, the first host is the current Receiving-SMTP server. The last is the destination of the e-mail. Syntax: The syntax of this command is ***MAIL FROM:*** *(<>/Reverse-Path) [Mail-parameters]*. Please see figure 6 for a working example of this command.

RCPT TO

This command is used to identify an individual recipient of the mail data; multiple recipients can be specified by multiple use of this command. The argument field contains a forward-path and may contain optional parameters. If service extensions were negotiated, the RCPT command may also carry parameters associated with a particular service extension offered by the server. Syntax: The syntax of this command is ***RCPT TO:*** *(<Postmaster@" domain>/ <Postmaster>/ Forward-Path) [Rcpt-parameters.* Please see figure 6 for a working example of this command.

DATA

This command is sent by a client to initiate the transfer of message. It causes the mail data to be appended to the mail data buffer. It is followed by actual data that makes up the e-mail message. This includes both the body text and headers such as the subject line. The mail data is terminated by a line containing only a period, that is, the character sequence <CRLF>.<CRLF>. Receipt of the end of mail data indication requires the server to process the stored mail transaction information. This processing Consumes the information in the reverse-path buffer, the forward-path buffer, and the mail data buffer, and on the completion of this command these buffers are cleared. If the processing is successful, the receiver MUST send an OK reply. When the SMTP server accepts a message either for relaying or for final delivery, it inserts a trace record also called time stamp line or received line at the top of the mail data. Syntax: The syntax of this command is ***DATA***. Please see figure 6 for a working example of this command.

RSET

This command nullifies the entire message transaction and resets the buffer without closing the connection. The receiver sends a "250 OK" reply to a RSET command with no arguments. Since EHLO implies some additional processing and response by the server, RSET will normally be more efficient than reissuing that command, even though the formal semantics are the same. Syntax: The syntax of this command is ***RSET***.

Example 3:
C: *RSET*
S: *250 2.0.0 Ok*

VRFY

This command will request that the receiving SMTP server verify that a given e-mail username is valid. The SMTP server will reply with the login name of the user. Syntax: The syntax of this command is ***VRFY*** *<username>*.

Example 4:
C: *VRFY bob@b.com*
S: *252 2.0.0bob@b.com*
C: *VRFY bob@c.com*
S: *554 5.7.1 <bob@c.com>:Relay access denied*

EXPN

This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list. A server does not return a 250 code in response to a VRFY or EXPN commands unless it has actually verified the address. In that case, 502 (Command not implemented) or 500 (Syntax error, command unrecognized) is returned. In certain cases when address appears to be valid but cannot be reasonably verified a reply code 252 is returned. Syntax: The syntax of this command is ***EXPN*** *<username>*.

HELP

This command causes the server to send helpful information to the client. The command may take an argument and return more specific information. It has no effect on the reverse-path buffer, the forward path buffer, or the mail data buffer and may be issued at any time. Syntax: The syntax of this command is ***HELP*** *<command name>*.

NOOP

This command does not affect any parameters or previously entered commands. It specifies no action other than that the receiver send an OK reply. This command has no effect on the reverse-path buffer, the forward path buffer, or the mail data buffer and may be issued at any time. If a parameter string is specified, servers SHOULD ignore it. The syntax of this command is ***NOOP.***

Example 5:
C: *NOOP*
S: *250 2.0.0 Ok*

QUIT

This command makes the receiver send an OK reply, and then close the transmission channel. The receiver does intentionally close the transmission channel until it receives and replies to a QUIT command even in case of an error. The sender does not intentionally close the transmission channel until it sends a QUIT command and waits until it receives the reply even if there was an error response to a previous command. The syntax of this command is ***QUIT***. Please see figure 6 for a working example of this command.

SAML

This stands for Send and Mail. Mail is the typical use today with SMTP. The send method is meant to be used when the SMTP server has been implemented to deliver mail directly to a recipient that is actively connected. The argument for this command, again, is a reverse-path showing the path to the destination of the e-mail.

SOML

This stands for Send or Mail. Similar to SAML, this command requests that the mail be "sent" (for example, directly to the actively connected recipient) or mailed. The server tries the Send method first, and if that fails, the server attempts to deliver the message to the destination mailbox.

SEND

This command, not often implemented, specifies that the mail message be delivered directly to the destination, if it's actively connected. If this cannot be done, the server returns a message code of 450 (the mailbox is not available). Similar to the SAML command, the argument for this command is the text FROM: followed by the reverse-path to the destination mailbox.

TURN

This command is used to switch the roles of sender and receiver. It instructs the Receiver to assume the role of the Sender of the mail (in which an OK response is returned). The server can refuse (with a code 502) and remain in the role of Receiver.

ATRN

This command is used for authenticated TURN. After a client has been authenticated to the SMTP server, this command instructs the Receiver to assume the charge of sender. The receiver can return an OK response if it is ready to assume charge of sender or otherwise, returns a Bad Gateway message (reply code 502) and remain in the role as receiver.

SIZE [14]

The SIZE command, allows the receiving host to tell the sending host the maximum message size before the

message is transmitted. This is supported only by SMTP implementations that use the SMTP Service Extensions. Many organizations implement a maximum size limit for inbound e-mail messages. Imposing such a limit prevents senders from running a server out of disk space or launching a denial-of-service attack by e-mailing a message with extremely large attachments. For Example please see ***EHLO*** command.

DSN [15]

This command enables delivery status notifications. It notifies the host of a delivery failure, and is considered an improvement over simple non-delivery reports. An extended SMTP server which implements this service extension will accept an optional NOTIFY parameter with the RCPT command. If present, the NOTIFY parameter alters the default conditions for generation of Delivery Status Notifications only on failure. The ESMTP client may also request (via the RET parameter) whether the entire contents of the original message should be returned as opposed to just the headers of that message, along with the DSN. A RCPT command issued by a client may contain the optional ESMTP-keyword "NOTIFY", to specify the conditions under which the SMTP server should generate DSNs for that recipient.

Example 6:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C: *MAIL FROM: alice@a.com RET=HDRS ENVID=QQ314159*
S: *250 2.1.0 OK*
C*: RCPT TO: bob@b.com NOTIFY=SUCCESS RCPT=rfc822;bob@b.com*
S: *250 recipient ok*
C*: RCPT TO: bob2@b.com NOTIFY=FAILURE RCPT=rfc822;bob2@b.com*
S: *250 recipient ok*
C*: RCPT TO: bob3@b.com NOTIFY=SUCCESS, FAILURE RCPT=rfc822;bob3@b.com*
S: *250 recipient ok*

The receiving domains mail server mailboxXXXX.mailhostingXXXX.com sends DSN in form of e-mail(s) for each recipient where DSN was requested through the NOTIFY parameter of RCPT command.

ETRN

ETRN define extensions to the SMTP service whereby a client ("sender-SMTP") may request that the server ("receiver-SMTP") start the processing of its mail queues for messages that are waiting at the server for the client machine. If any messages are at the server for the client, then the server creates a new SMTP session and sends the messages at that time. The extended ETRN command is issued by the client host when it wishes to start the SMTP queue processing of a given server host. Syntax: The syntax of this command is: ***ETRN*** *[<option character>] <node name>*.

Example 7:
C: ETRN
S: *500 Syntax Error*
C: ETRN localname
S: *501 Syntax Error in Parameter*
C: ETRN b.com
S: *458 Unable to queue messages for node b.com*

C: ETRN allowed.com
S: *250 OK, queuing for node allowed.com started*

C: ETRN allowed.com
S: *251 OK, no messages waiting for node allowed.com*

C: ETRN allowed.com
S: *252 OK, pending messages for node allowed.com started*

CHUNKING

The CHUNKING command indicates that the recipient supports the use of the BDAT command. The BDAT command is used as an alternative to the DATA command. It replaces the DATA command so that the

SMTP host does not have to continuously scan for the end of the data; this command sends a BDAT command with an argument that contains the total number of bytes in a message. The receiving server counts the bytes in the message and, when the message size equals the value sent by the BDAT command, the server assumes it has received all of the message data. For example of Chunking, please see the BDAT command.

PIPELINING [16]

Command pipelining is batching up of multiple commands into a single TCP send operation. It provides the ability to send a stream of commands without waiting for a response after each command which is inefficient way to communicate with a host. If a client wishes to employ command pipelining, it first determines its support by the server by issuing the EHLO command. If the server responds with code 250 and the response includes the EHLO keyword value PIPELINING, then the server supports command pipelining. The client may transmit groups of SMTP commands in batches without waiting for a response to each individual command. In particular, the commands RSET, MAIL FROM, SEND FROM, SOML FROM, SAML FROM, and RCPT TO can all appear anywhere in a pipelined command group. The EHLO, DATA, VRFY, EXPN, TURN, QUIT, and NOOP commands can only appear as the last command in a group since their success or failure produces a change of state which the client must accommodate.

Because current networks are reliable enough, the transmit-and-wait nature of SMTP is no longer necessary. As an alternative, ESMTP supports pipelining. Pipelining refers to the host's ability to send commands in batches, without having to wait for a response after each command.

Example 8:
C*: HELO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
C: *MAIL FROM: alice@a.com*
S: *250 sender <allice@a.com> OK*
C: *RCPT TO: bob@b.com>*
S: *250 recipient <bob@b.com> OK*
C: *RCPT TO: <tariq@b.com>*
S: *250 recipient <tariq@b.com> OK*
C: *RCPT TO:<taheem@b.com>*
S: *250 recipient <taheem@innosoft.com> OK*
C: *DATA*
S: 354 End Data with <CR><LF>.<CR><LF>
C: *Subject: Test Message*
C: *Date: 10-01-2010*
C: *This is my message body.*
C: **.**
C: 250 2.0.0 Ok: queued as 85313139007B
C: QUIT
S: 221 2.0.0 Bye

In the above simple example the client waits for a server response 8 times. But if pipelining is employed (as shown in example below) it is possible to reduce the waiting time.

Example 9:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C: *MAIL FROM: alice@a.com*
C: *RCPT TO: bob@b.com>*
C: *RCPT TO: <tariq@b.com>*
C: *RCPT TO :<taheem@b.com>*
C: *DATA*
S: *250 sender <allice@a.com> OK*
S: *250 recipient <bob@b.com> OK*
S: *250 recipient <tariq@b.com> OK*
S: *250 recipient <taheem@innosoft.com> OK*
S: 354 End Data with <CR><LF>.<CR><LF>
C: *Subject: Test Message*
C: *Date: 10-01-2010*
C: *This is my message body.*
C: **.**
C: QUIT
C: 250 2.0.0 Ok: queued as 85313139007B
S: 221 2.0.0 Bye

In the above example using pipelining, the client waits for a server response 3 times only.

BDAT [17]

This command replaces the DATA command in situations where e-mail message contains large data. The BDAT [17] command sends the message size and the message itself. When the receiving host has received the number of bytes specified by the BDAT command, it assumes that it has reached the end of the message. Using this technique frees the server from having to scan every packet for an end-of-message flag. The following example demonstrates the usage of BDAT and Chunking commands.

Example 10:
C: EHLO MYDOMAIN.COM
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-SIZE 20971520*
S: *250-PIPELINING*
S*: 250-BINARYMIME*
S*: 250 CHUNKING*
C: MAIL FROM: alice@a.com BODY=BINARYMIME
C: RCPT TO: tariq@b.com
C: RCPT TO: taheem@b.com
S: *250 <alice@a.com>... Sender and BINARYMIME ok*
S: *250 <tariq@b.com ... Recipient ok*
S: *250 <taheem@b.com>... Recipient ok*
C: BDAT 100000
C: (First 10000 octets of canonical MIME message data)
C: BDAT 324
C: (Remaining 324 octets of canonical MIME message data)
C: BDAT 0 LAST
S: *250 100000 octets received*
S: *250 324 octets received*
S: *250 Message OK, 100324 octets received*
C: QUIT
S: *221 Goodbye*

AUTH

The **AUTH** command is a SMTP service extension by which an SMTP client may indicate an authentication mechanism to the server, perform an authentication protocol exchange, optionally negotiate a security layer for subsequent protocol interactions during this session and, during a mail transaction, optionally specify a mailbox associated with the identity that submitted the message to the mail delivery system. This extension includes a profile of the Simple Authentication and Security Layer (SASL) for SMTP. The AUTH EHLO keyword contains as a parameter a space-separated list of the names of available [SASL] mechanisms. The list of available mechanisms MAY change after a successful STARTTLS command [SMTP-TLS]. Syntax: The syntax of this command is: ***AUTH*** mechanism [initial-response]. The mechanism is a string identifying a [SASL] authentication mechanism which may be ***PLAIN, LOGIN, CRAM-MD5***, ***DIGEST-MD5***, *etc.* The authentication mechanism chooses how to login and which level of security that should be used. The initial-response is an optional initial client response which must be encoded in BASE64 or contain a single character "=". The example shown below demonstrates ***AUTH*** command with ***PLAIN*** SASL mechanism making use of the initial client response:

Example 11:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C: AUTH PLAIN
S: *334*
C: dEBuaWN0c29mdC5jb20gdGVzdA==
S: 235 2.7.0 Authentication successful

In the above example although the keyword PLAIN is used, the username and password (in this example: *t@nictsoft.com test*) are not sent as plain text but are encoded using BASE64 encoding to form *dEBuaWN0c29mdC5jb20gdGVzdA==*. It is also possible to send the username and password, together with the AUTH PLAIN command, as a single line as shown below:

Example 12:
C: AUTH PLAIN
dEBuaWN0c29mdC5jb20gdGVzdA==
S: 235 2.7.0 Authentication successful

The LOGIN mechanism is another common method to login to an SMTP server. The SMTP communication example below shows how AUTH LOGIN can be used to make an authenticated login to server:

Example 13:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C: AUTH LOGIN
S: *334 VXNlcm5hbWU6*
C: dEBuaWN0c29mdC5jb20=
S: *334 UGFzc3dvcmQ6*
C: dGVzdA==
S: 235 2.7.0 Authentication successful

In the above example the response to ***AUTH LOGIN*** is ***334 VXNlcm5hbWU6*** which is 334 status code with BASE64 coded message ***Username:.*** The BASE64 encoded username ***dEBuaWN0c29mdC5jb20=*** of text username ***t@nictsoft.com*** is given in the next response from the client. The server responds with ***334 UGFzc3dvcmQ6*** which is 334 status code with BASE64 coded message ***Password:.*** The BASE64 encoded password ***dGVzdA==*** of text password ***test*** is given in the next response from the client and the authentication is successful.

The drawback using the PLAIN and LOGIN authentication mechanisms is that the username and password can be decoded easily. To obtain higher security, ***CRAM-MD5*** authentication mechanism can be used. CRAM-MD5 combines a challenge-response authentication mechanism to exchange information and a cryptographic Message Digest 5 algorithm to encrypt important information. In this mechanism the client does not begin the authentication exchange and instead includes a server challenge. The following example demonstrates this mechanism:

Example 14:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C*: AUTH CRAM-MD5*
S*: 334 PDQxOTI5NDIzNDEuMTI4Mjg0NzJAc2*
91cmNlZm91ci5hbmRyZXcuY211LmVkdT4=
C: cmpzMyBlYzNhNTlmZWQzOTVhYmExZW
M2MzY3YzRmNGI0MWFjMA==
S: 235 2.7.0 Authentication successful

The response to ***AUTH CRAM-MD5*** from the servers is a one-time BASE64 encoded challenge to the client. The client responds by sending a BASE64 encoded string to the server that contains a username and a 16-byte digest in hexadecimal notation. The digest in the reply string is the output of a Hash-based Message Authentication Code calculation with the password as the secret key and the SMTP server's original challenge as the message. The SMTP server also calculates its own digest with its notion of the user's password. The authentication is successful if the client's digest and the server's digest match.

STARTTLS [18]

The STARTTLS command allows a client to initiate a TLS-based Secure Socket Layer (SSL) connection between itself and the SMTP server. The STARTTLS keyword is used to tell the SMTP client that the SMTP server allows the use of TLS.

E-mail plain text communicate using SMTP protocol over the Internet is insecure because the message travels through one or more routers that are not trusted. To improve security, an encrypted TLS (Transport Layer Security) connection can be used when communicating between the e-mail server and the client. TLS is most useful when a login username and password (using AUTH command) are encrypted. TLS can be used to encrypt the whole e-mail message, but the command does not guarantee that the whole message will stay encrypted the whole way to the receiver; some e-mail servers can decide to send the e-mail message with no encryption. But at least the username and password used with the AUTH command will stay encrypted. Use of STARTTLS command together with the AUTH command is a very secure way to authenticate users. The following example demonstrates the STARTTLS.

Example 15:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250- STARTTLS*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C*: **STARTTLS***
S: *220 Ready to start TLS*
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*
S: *250-VRFY*
S: *250-ETRN*
S: *250-AUTH PLAIN LOGIN*
S: *250-AUTH=PLAIN LOGIN*
S: *250- STARTTLS*
S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
S: *250-DNS*
C: AUTH LOGIN
S: *334 VXNlcm5hbWU6*
C: dEBuaWN0c29mdC5jb20=
S: *334 UGFzc3dvcmQ6*
C: dGVzdA==
S: 235 2.7.0 Authentication successful

In the above example the client first establishes a connection with the server to know its supported extensions. The client requests the server to start Transport Layer Security. Once status code 220 is received by the client, it issues EHLO command again to start a new SMTP session under TLS. The AUTH command is next used to authenticate the sender under a TLS cover.

8BITMIME [19]

Initially, e-mail messages consisted of ASCII text only. As such, e-mail messages typically used a 7-bit encoding scheme, which was ideally suited for messages consisting of letters, numbers and some special symbols. Currently, e-mail messages now include HTML-elements, large documents of different formats as attachments and Unicode characters that are not a part of the ASCII character set but SMTP still encodes messages in a 7-bit format. To get around the limitations of 7-bit encoding, SMTP messages typically are encoded in 8bitmime, and then encapsulated in a 7-bit packet for transmission. Upon receipt, the 7-bit capsule is stripped away, and the message is converted back to its original 8bitmime format. This encapsulation puts additional burden on the mail server thus reducing its performance. To avoid this, ESMTP protocol supports 8bitmime natively wherein the e-mail can be transmitted and received in this format without the need for 7-bit encapsulation. A mail client that wants to transmit 8bitmime data, first enquires the receiver about its support through EHLO command. The 8bitmime complaint server in response to EHLO command besides other replies also replies with 250-*8BITMIME.* Following example demonstrates the usage of *8BITMIME.*

Example 16:
C*: EHLO MYDOMAIN.COM*
S: *250- mailboxXXXX.mailhostingXXXX.com*
S: *250-PIPELINING*
S: *250-SIZE 20971520*

S: *250-ENHANCEDSTATUSCODES*
S: *250-8BITMIME*
C: MAIL FROM: alice@a.com BODY=8BITMIME
S: *250 Sender and 8BITMIME ok*
C: RCPT TO: bob@b.com
S: *250 recipient ok*
C: DATA
S: *354 Send 8BITMIME message, ending in CRLF.CRLF.*
C: ….
C: ….
C: .
S: *250 ok*
C: QUIT
S: *221 Goodbye*

It is essential to understand the differentiate between *8BITMIME* and *BINARYMIME* responses of server. *8BITMIME* server response to EHLO command indicates that the local SMTP virtual server supports eight-bit Multipurpose Internet Mail Extensions (MIME) messages. The *BINARYMIME* server response to EHLO command indicates that the SMTP virtual server accepts a message that contains binary material without transport encoding by using a BODY parameter with a value of "BINARYMIME" in the MAIL command. When the SMTP server accepts a MAIL command with a BODY parameter of BINARYMIME, the server agrees to preserve all bits in each octet passed using the BDAT command. The BINARYMIME SMTP extension can only be used with CHUNKING.

ENHANCEDSTATUSCODES

Request for Comments RFC3463 [19] provides an enhanced set of status codes for Delivery Status Notification (DSN) messages. The Enhanced Status Codes provide a standard mechanism for reporting mail system errors, and provide more meaningful information than the standard error codes defined in the SMTP RFC (821). The server response to client ***EHLO*** command includes *250-ENHANCEDSTATUSCODES* if the server supports enhanced delivery status notification messages. Recently, IANA has created the registry "SMTP Enhanced Status Codes" [20]. The SMTP Enhanced Status Codes registry has three tables namely Class Sub-Codes, Subject Sub-Codes, and Enumerated Status Codes. Status codes consist of three numerical fields separated by ".". The first sub-code indicates whether the delivery attempt was successful. The second sub-code indicates the probable source of any delivery anomalies, and the third sub-code indicates a precise error condition. The syntax of the new status codes is defined as:

status-code = class "." subject "." detail
class = "2"/"4"/"5"
*subject = 1*3digit*
*detail = 1*3digit*

SMTP Reply Codes

SMTP and ESMTP [21, 22] reply codes are of three digits. The first reply code digit indicates the success or failure of the command in general terms. This digit is interpreted in exactly the same way as it is in FTP. The second reply code digit is used to categorize messages into functional groups. This digit is used in the same general way as in FTP, but some of the functional groups are different in SMTP. The third reply code digit indicates a specific type of message within each functional groups described by the second digit. The third digit allows each functional group to have ten different reply codes for each reply type given by the first code digit that e.g. preliminary success and transient failure. SMTP reply codes with a short description of each are listed in table 2. The support of ENHANCEDSTATUSCODES SMTP extension supplements reply codes and thus the receiving server can issue them in response to each command. These ESMTP codes are similar to the standard reply codes and are also of three digits, but the digits are separated by periods. These enhanced codes provide more information to the sending clients about the results of operations, especially errors.

Table 2: **SMTP reply codes**

Reply Code	Description
101	*This error code is send when SMTP server or e-mail program is unable to start an SMTP session. Various possible reasons may be incorrectly spelt SMTP server, invalid IP address and invalid or busy SMTP port. Typical example of error messages with this error code are: "SMTP Error 101, Error opening connection" or "SMTP Error 101, cannot open SMTP stream".*
211	*This is a reply code that includes message about the mail server status. The server responds with this code in reply to some clients commands e.g. the client may issue a command to the mail server to display a list of commands it supports and the server responds with a reply 211 followed by the requested list.*
214	*This is a response to the "HELP" command. It displays information about the server, usually a URL to the FAQ page of the SMTP software running on the server.*
220	*This code indicates the mail service is running. It will normally contain a welcome message and/or the title of the SMTP software and, sometimes, the version number of the mail server software.*
221	*This reply indicates that the server is ending the mail session, i.e. it is closing the conversation as it has no more mail to send in this sending session. Typical examples are: "221 Closing connection" -or- "221 Goodbye".*
250	*This status code indicates successful execution of a variety of commands.*
251	*This code indicates that the e-mail account is not local to the mail server but the server will accept the e-mail and will forward it.*
252	*This response code means that the user account appears to be valid but could not be verified, however the server will try to deliver the message.*
354	*This is normally in response to the DATA command. It indicates that the server has received the mail envelope and is ready to accept mail body.*
421	*This reply code indicates that the mail transfer service is unavailable because of some transient event. It generally indicates that the mail server is currently unavailable but may be available later.*
422	*It indicates that either the recipient's mailbox or the message delivery folder on the recipient's mail server has crossed its storage limits.*
431	*This reply code is an indication of disk full error or out of memory error on the recipient mail server.*
432	*This is a status response specific to Exchange Server. It indicates that the recipient's mail queue on their Exchange Server has been stopped.*
441	*This is generated by sending client and indicates that the recipient's server is not responding.*
442	*This error code indicates that the connection was dropped during transmission which may be an unusual transient error.*
446	*This code indicates that the maximum hop count has exceeded for the message.*
447	*It indicates that the outgoing message has timed out because of some problems with the receiving server which objected to the message.*
449	*This response is specific to Exchange Server. It indicates that an SMTP connector is configured to use DNS without a smart host and also uses a non-SMTP address space.*
450	*This response indicates that the mailbox is unavailable at the receiver. It is a transient error at the receiver and the sender can retry after some time. Typical examples are: "450 Please try again later" -or- "SMTP Error 450 5.2.3 Msg Size greater than that allowed by Remote Host"*
451	*This code indicates that the action has been aborted locally. This is usually due to overloading at the server as a result of many messages or transient failures. Typical examples are: "SMTP error 451 Unable to complete command, DNS not available or timed out" -or- "451 Domain of sender address does not resolve".*

Reply Code	Description
452	*It indicates that the server's disk system has run out of storage space. Occasionally this error may be raised if the receiving mail server is overloaded by messages. Typical example is: "452 Out of memory".*
465	*This response is specific to Exchange Server returned by the recipient's server in case incoming e-mail specifies a Code Page that is not installed on the recipient's server.*
471	*This is a local error on the sending server and is often followed with "Please try again later" message. The error may be caused due to some problem with anti-spam or anti-virus software.*
500	*It indicates syntax error which may be generated due to an invalid SMTP command.*
501	*It indicates syntax error in parameters or arguments of a SMTP command.*
502	*This error code indicates that the command or function issued by the sending mail server is valid but has not been activated.*
503	*This code indicates a bad sequence of SMTP commands.*
504	*It indicates that the command and parameters of the issued SMTP command are valid, but some parameter is not implemented on the receiving server, or some additional parameter or action is missing.*
510	*This code indicates that the e-mail address specified in the SMTP command is invalid.*
511	*This code indicates that the e-mail address specified in the SMTP command is invalid.*
512	*This response indicates that the host server for the recipient's domain cannot be found through DNS. This response code is received when one intermediate servers is unable to resolve the domain name of a recipient e-mail address.*
513	*This status code is received in case the e-mail addresses are not defined correctly by the sender's mail server.*
523	*This error is received in case the total size of the sent message exceeds the limits on the recipient's server.*
550	*This error code indicates that the requested action has not been performed because the mailbox is not available. Typical examples are: "550 Invalid recipient", "550 User account is unavailable" and "550 No such user".*
551	*This response indicates a denied relay access.*
552	*It indicates that the recipient's mailbox has reached its maximum allowed storage limit.*
553	*It indicates an invalid e-mail address in "To", "CC", or "BCC" field of the message.*
554	*This is a permanent error indicating the transactions failure that cannot be resolved by resending the message in its current form.*

Conclusion

Simple Mail Transfer Protocol is the primary and most deployed protocol for e-mail communication. It is being continuously revised by the inclusion of new commands, security mechanisms, message formats and efficiency procedures. It has been found that all sending and receiving SMTP servers do not support all extensions of SMTP. Most SMTP servers are implemented using some library that needs to be constantly upgraded to take advantage of improvements in SMTP. The study of SMTP involves use of utilities permitting issuance of individual SMTP commands and studying of their responses directly. In this study, Telnet program has been used to study various SMTP commands and their responses. However, certain commands that require establishing Transport Layer Security (TLS) System,

simple Telnet client has not proved sufficient, and a secure connection between the SMTP server and the client was established before using Telnet. For study and development of SMTP extensions it is desired to build SMTP-Email utility that besides permitting to issue SMTP commands directly can also perform various other functions required during development and testing. These include format conversions, support for cryptography, handling of security certificates, support batch submission of SMTP commands, establishment of TCP and UDP sessions with other servers without need to disconnect the current session and to support various file operations. It would be our endeavor to build such a system in near future.

Biographies

M. Tariq Banday was born in 1969. He did his M. Sc. and M. Phil. Degrees from the Department of Electronics, University of Kashmir, Srinagar, India in 1996 and 2008 respectively. He did advanced diploma course in computers and qualified UGC NET examination in 1997 and 1998. At present he is working as Assistant Professor in the Department of Electronics & Instrumentation Technology, University of Kashmir, Srinagar, India. He has to his credit several research publications in reputed journals and conference proceedings. He is a member of Computer Society of India, International Association of Engineers and ACM. His current research interests include Network Security, Internet Protocols and Network Architecture.

Jameel A. Qadri was born in 1972. He did his M.Sc. degree in Electronics and Post Graduate Diploma in Computer Applications from Department of Electronics, University of Kashmir, Srinagar, India in 1998 and 2000 respectively. He received M. Sc. Degree in Electronic Commerce from Middlesex University, Hendon Campus, London, UK in 2008. At present he is Lecturer British Institute of Technology & Ecommerce, London, United Kingdom. His research articles have been published in journals and conference proceedings. His Current research interests are Internet Security, Knowledge Management and Web Accessibility.

Nisar A. Shah was born in 1953. He did his M. Sc. and Ph. D. Degrees from the department of Physics, University of Kashmir, Srinagar, India in 1976 and 1981 respectively. At present he is working as Professor in the Department of Electronics & Instrumentation Technology, University of Kashmir. He has to his credit about 150 research publications which have been published in national and international journals of repute. He has supervised several research scholars in M. Phil. and Ph. D. programs. His current research interests include Digital Signal Processing and Network Security.

References

[1] Markus Jakobsson (Ed.) and Steven Myers (Ed.), '***Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft***", Adobe E-Book, ISBN: 978-0-470-08609-4, Dec 2006.

[2] Resnick, P. Ed., '***Internet Message Format***', IETF RFC 2822, Apr 2001.

[3] Klensin '***Simple Mail Transfer Protocol***' IETF RFC 2821, Apr 2001.

[4] R. Siemborski, Ed., and A. Melnikov, Ed., '***SMTP Service Extension for Authentication***', IETF RFC 4954, Jul 2007.

[5] G. Schryen, '***A Formal Approach towards Assessing the Effectiveness of Anti-spam Procedures***', In proceedings of 39th Hawaii International Conference on System Science, vol. 6 pp. 129a-129a, May 2006.

[6] D. Atkins and R. Austein, '***Threat analysis of the Domain Name System (DNS)***', IETF RFC 3833, Aug 2004.

[7] P.J. Sandford, J. M. Sandford, and D. J. Parish, '***Analysis of SMTP Connection Characteristics for Detecting Spam Relays***', International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06), pp.68, 2006.

[8] D. Cridland, Ed., A. Melnikov, Ed., and S. Maes, Ed., '***The Internet Email to Support Diverse Service Environments (Lemonade) Profile***' IETF RFC 5550, Aug 2009.

[9] C. Newman, and A. Melnikov, Ed., '***Internationalized Delivery Status and Disposition Notifications***', IETF RFC 5337, Sep 2008.

[10] J. Yao, Ed., and W. Mao, Ed., '***SMTP Extension for Internationalized Email Addresses***', IETF RFC 5336, Sep 2008.

[11] G. White, and G. Vaudreuil, '***SMTP Submission Service Extension for Future Message Release***', IETF RFC 4865, May 2007.

[12] K. Toyoda, and D. Crocker, '***SMTP and MIME Extensions for Content Conversion***', IETF RFC 4141, Nov, 2005.

[13] E. Allman, and T. Hansen, '***SMTP Service Extension for Message Tracking***', IETF RFC 3885, Sep 2004.

[14] J. Klensin, N. Freed, and K. Moore, '***SMTP Service Extension for Message Size Declaration***', IETF RFC 1870, Nov 1995.

[15] K. Moore, '***Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)***', IETF RFC 3461, Jan 2003.

[16] N. Freed, '***SMTP Service Extension for Command Pipelining***' IETF RFC 2920, Sep 2000.

[17] G. Vaudreuil, '***SMTP Service Extensions for Transmission of Large and Binary MIME Messages***', IETF RFC 3030, Dec 2000.

[18] P. Hoffman, '***SMTP Service Extension for Secure SMTP over Transport Layer Security***', IETF RFC 3207, Feb 2002.

[19] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker, '***SMTP Service Extension for 8bit-MIMEtransport***', IETF RFC 1652 , July 1994.

[20] T. Hansen, and J. Klensin, '***A Registry for SMTP Enhanced Mail System Status Codes***' IETF RFC 5248, Jun 2008.

[21] G. Vaudreuil, '***Enhanced Mail System Status Codes***', IETF RFC 3463, Jan 2003.

[22] N. Freed, '***SMTP Service Extension for Returning Enhanced Error Codes***', IETF RFC 2034, Oct 1996.

芽|Sprouts

Working Papers on Information Systems | ISSN 1535-6078

Office:
Sprouts
University of Amsterdam
Roetersstraat 11, Room E 2.74
1018 WB Amsterdam, Netherlands
Email: admin@sprouts.aisnet.org

芽|Sprouts