RedinSkala

The SMTP Protocol Fundamentals

Published date: January 2013

Version 1.0

1st. Edition 2013

Table of Contents

I. Purpose and Audience

The goal of this book is to improve the knowledge of the SMTP Protocol to tighten the security of email in your business, locating the security holes in the actual standards, the forms, methods and tools that may be used to mitigate them and therefore improving your business productivity.
This book is intended for technical staff with minimal experience in:

- Email Systems (Exchange, Postfix, Sendmail or other)
- Windows Server Operating Systems (2003/2008)
- Linux Server Operating Systems
- Basics on email flow

When you're finished, you should have enough knowledge to identify if a particular mail or SMTP connection may represent a risk to your Organization, based on a proper interpretation of the SMTP protocol.

Chapter 1. Introduction to Email

IN THIS CHAPTER:

- ✓ Email and standards
- ✓ Email flow components
- ✓ Email Threat Landscape
- ✓ Summary

No matter which MTA is implemented on your Organization, the way they all behave is based on the same mechanisms of the SMTP Protocol and therefore they are all exposed to the same types of attacks.

The email protocol has a close relationship with the usual process used by a post office in real life. Each process, such as determining the route to reach the recipient, the letter format, the situations where the letter cannot be delivered, among others, has its counterpart on the SMTP Protocol. This makes it easier to assimilate how email operates.

However, the SMTP Protocol is not based on a single standard. In order to really understand it you need to understand its relationship and integration with other protocols that may, in some cases, add more security holes and vulnerabilities than those that are native to the email protocol.

There are several threats that email platforms must face every day. By understanding how an attacker uses SMTP principles to compromise an email architecture, you'll be able to act proactively, improving the security of your Organization and our employee's productivity as well.

1.1. Email and Standards

Electronic Mail (or email) is the generic term used to designate a service that allows a user to send and receive information in an automatic way, having an storage mechanism (mailbox) from which this user may check such information.

In order to make this information flow reliable, several International groups generated a universal protocol that can be used under any platform and allows the process of sending and receiving any email.

- **August 1982.** IETF (Internet Engineering Task Force) publishes the RFC 0821 on Simple Mail Transfer Protocol, being this the first document that intended to regulate the transmission of email, standardizing the network packets, instructions' order and type of content used during the communication. This RFC was the first attempt to create a protocol that could work under any platform (Windows, Linux, UNIX, etc.) and was such a success that it lasted almost twenty years until it needed a new revision in order to combat the increasing email threat landscape.
- **April 2001.** RFC 2821 is officially published obsolescing RFC 821. This new document incorporates several enhancements to the protocol and from that moment it would receive the new name of Extended SMTP (ESMTP). The new characteristics allowed for a better control, including receiving a notification when the email has been received, receiving a notification with the original email attached every time the email could not be delivered, add non-standard characters in the email headers, among others. There is also an improvement on the standard that will allow SMTP to communicate among MTAs using encryption methods. Although ESMTP becomes the new standard, it is important to emphasize that it is mandatory that every MTA still allow the use of normal SMTP.
- **October 2008.** RFC 5321 obsoletes RFC 2821 incorporating to ESMTP new functions to handle TLS and encrypted communications besides incorporating new rules on the use of the original SMTP functions.

It's worth mentioning that every new RFC doesn't contain the whole material contained in the previous RFC. This is why it's important to study the three SMTP RFC's in order to fully understand the behavior and rules of email.

In spite of RFC 5321 containing all the information about SMTP/ESMTP implementation, this is not enough to describe the full email flow. It is also necessary to define the conventions used for naming the domains and mailboxes, the DNS records destined to find the final MTA and the type of content allowed for sending. All this information is contained on the appropriate section of every RFC. Below is the official list that complements RFC 5321 for a full email flow.

```
   [1]    Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821,
          August 1982.

   [2]    Mockapetris, P., "Domain names - implementation and
          specification", STD 13, RFC 1035, November 1987.

   [3]    Braden, R., "Requirements for Internet Hosts - Application and
          Support", STD 3, RFC 1123, October 1989.

   [4]    Resnick, P., "Internet Message Format", RFC 5322, October 2008.

   [5]    Bradner, S., "Key words for use in RFCs to Indicate Requirement
          Levels", BCP 14, RFC 2119, March 1997.

   [6]    American National Standards Institute (formerly United States
          of America Standards Institute), "USA Code for Information
          Interchange", ANSI X3.4-1968, 1968.

          ANSI X3.4-1968 has been replaced by newer versions with slight
          modifications, but the 1968 version remains definitive for the
          Internet.

   [7]    Crocker, D. and P. Overell, "Augmented BNF for Syntax
          Specifications: ABNF", STD 68, RFC 5234, January 2008.

   [8]    Hinden, R. and S. Deering, "IP Version 6 Addressing
          Architecture", RFC 4291, February 2006.

   [9]    Newman, C., "ESMTP and LMTP Transmission Types Registration",
          RFC 3848, July 2004.

   [10]   Klensin, J., Freed, N., and K. Moore, "SMTP Service Extension
          for Message Size Declaration", STD 10, RFC 1870, November 1995.
```

The strict implementation of each of these RFCs will assure that any MTA in the world, no matter the platform, will be able to send and receive email successfully. In the following chapters we'll study each of this points to make sure our mail solutions are working according to the international standards in both sending and receiving email. It's worth mentioning that the main purpose of SMTP isn't for the mail flow to be a safer mean for information exchange, its scope is only to make sure that any SMTP Client is able to communicate with any SMTP Server with the right syntax and mechanisms. It's because of this that many hackers or cyber-criminals take advantage of the "security holes" in SMTP Protocol to perform attacks on this kind of infrastructure. By understanding each of the protocol basis we'll be able to make the right adjustments to our solutions in order to cover these vulnerabilities and anticipate to any new attacks.

1.2. Email flow components

In Figure 1 we can see the basic SMTP model used to send an email between two MTA servers:

NOTE: MTA (Message Transfer Agent) is the generic term used to designate any server / application (for example Exchange and Postfix) that are capable of sending, receiving and queuing email. Any entity that is not able to queue is designated as SMTP Proxy only.

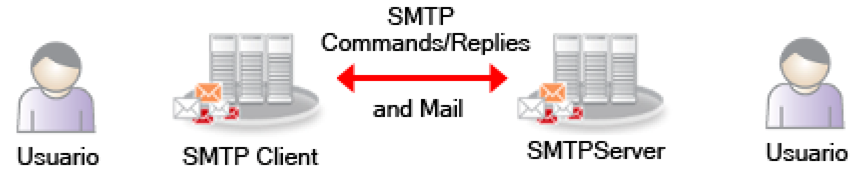


Image 1. Basic components of the email flow

This simplistic model, will help us understand the basics of what happens when we use our email clients to send and receive email. This model defines the main components interacting in an email transfer. These are:

- SMTP Client. Is responsible for starting and closing the email transfer by answering with the proper commands to each of the SMTP Server replies.
- SMTP Server. Is responsible for replying to each command sent by the SMTP Client and when appropriate, receive and accept the full responsibility for the email delivery. It should also respond with a proper code that reflects the reason why a particular email cannot be accepted if for any reason the reception of such mail should be rejected.
- SMTP Commands/Replies. These are a set of ordered and coherent commands and codes used to transfer each piece of information that composes the full email. This information is enough for an MTA to react accordingly in such a case when the email transfer action is unsuccessful.

For a more detailed description about the multiple components we may find in an email transaction we'll study the process described in Figure 2.

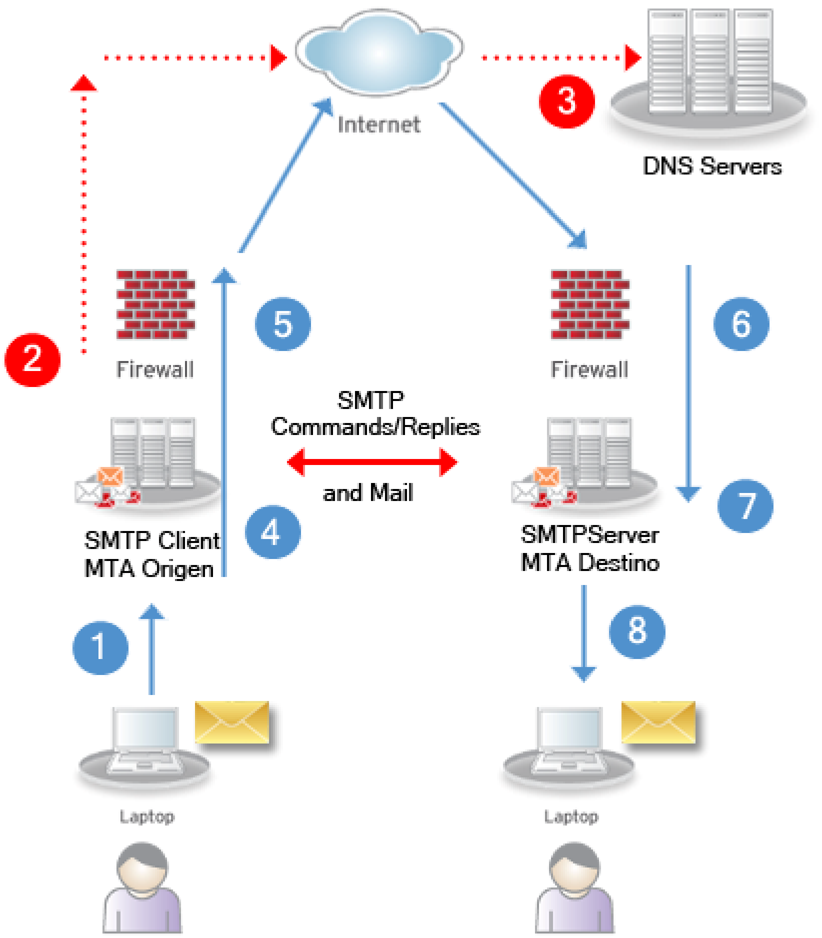


Image 2. Advanced model of the SMTP transaction

Let's analyze now each of the steps involved in the email flow according to Figure 2.

1. **Creating a new mail.** "Creating" here means a user or an automatic notification system generating a completely new email (this may be an email web interface, an application like Outlook or any automatic system that is able to generate an SMTP conversation), responds to a previously received mail or re-sends a previously received mail. The sender generates all the information the email needs to be transferred to the next step, this information includes: sender's name, recipient's name, headers (email clients usually write this information automatically with the information they have at the moment) and finally the body of the email (if there are any attachments they will be included at this point). After it is created, the email is transferred to the MTA which will have the responsibility to deliver it to its final destination.
2. **DNS-MX Query.** The MTA receives the email and marks it with a unique identifier (Message-ID) that may be used for further tracing. Once marked, the email gets queued as the MTA must first identify what the destination domains are. In order

to do this, the MTA must query a DNS server to find the MX (Mail Exchanger) resource type of the destination domain found in the "RCPT TO" command to find the IP addresses it must contact. This request is sent to the Firewall on the port 53 (UDP) which in turn redirects the traffic to Internet in order to contact the DNS server. If the recipient's domain is not public but private, this means it is an internal domain used by company, then this mail will be treated as local and no DNS query is necessary.

3. **Selecting the MX Record.** When the DNS Server receives the MX registry query, it verifies in its own table and resolves the IP of the email server requested. If the registry resolves to a name (FQDN) it then verifies the type "A" registry for that server as well in order to answer to the request with a list that contains both the server name and the server IP. This table also includes the priorities the MTA should verify in order to select the appropriate email server. This is a typical process when there are more than one MX registry for the same domain.
4. **Starting the SMTP Session.** Now that the MTA has the MX registry table, it's time to verify the priority of each server in the DNS response and choose the server with the highest priority. In case there is more than one server with the same priority, choosing the server will be decided randomly using the Round-Robbin algorithm. At this point, the MTA acquires the SMTP Client role. This role will allow it to start an SMTP conversation where it is the one that starts and ends the whole session. With this information, the MTA tries to make the first contact to the destination MTA directing the request through port 25 in the Firewall.
5. **Routing the SMTP Communication.** When the Firewall receives the TCP packets on port 25 from the originating MTA, it directs the traffic to Internet using the most appropriate route. It is worth mentioning that if the communication between two MTA servers is not encrypted, it could be intercepted in the middle very easily given the fact that this kind of communication is not usually peer-to-peer.
6. **TCP Session.** When the destination Firewall receives the originating MTA's request at port 25, a TCP session is established between both MTA servers so the communication may be possible. The traffic received by the Firewall is transferred to internal network until it reaches the destination MTA.
7. **SMTP Session.** The destination MTA receives the request at TCP port 25 from the Firewall. From now on, this MTA server will behave as an SMTP Server. This means the destination MTA server will respond to each request the SMTP Client sends. The SMTP Server first responds to the communication with a 220 code, inviting the SMTP Client to initiate an SMTP conversation. The SMTP Server won't be able to end this conversation until the SMTP Client request for it or until a considerable amount of time has passed in a way it can justify a "forced close". Both MTA servers start transferring the email information in an ordered and coherent way. To each SMTP Client request, the SMTP Server will respond with a code that will let the SMTP Client know if it may continue the conversation or if there has been some communication error it has to correct. Once the email transfer has ended, the session ends. The SMTP Server marks the email with the original Message-ID and queues it until delivery to its final destination.

8. **Email delivery.** Finally the destination MTA frees the email from the queue delivering it to the recipient's mailbox.

Each of these components may be more complex as we'll see in the following chapters, however, this model shows us the operational model that will work under any email architecture no matter what MTA is used.

1.3. Email Threat Landscape

When we talk about email threats or attacks we usually think about SPAM as the most important but there are other vectors that could compromise your Organization's operation and confidentiality. In the following sections we'll discuss about the most important ones.

1.3.1. SPAM

The main problem Organizations have to face on daily basis is without a doubt SPAM. A SPAM mail is basically any kind of mail that has arrived to your MTA server without being explicitly solicited by the recipient. Under this category there are several types of campaigns you might face.

1. Fake drugs
2. Fake fashion devices (jewelry for example)
3. Pornography and prostitution
4. Stock Information. This type of spam try to increase the stock value deceiving the users to buy them
5. Phishing and other frauds
6. Trojans that infect PC's with other malware programs. In general, they use fake ads, news and fake links to malicious sites
7. Auto-responses and delivery error notifications sent by miss configured or malicious MTA Servers
8. SPAM from other sources like politicians, charity and dishonest business

In order to achieve their goal, spammers use several methods, the same kind that are the result from the ISP's restrictions or Standars incorporated to avoid spam propagation. Some of the most important mechanisms used by spammers are described below:

1. **Botnets.** These are networks of "zombies" (infected PC's with malware) that send mail on behalf of the spammer, without the consent of the owner of the machine. Botnets are controlled by a "botmaster" who sells this kind of service to spammers.
2. **Free mail services.** Free mail public accounts (like Yahoo and Hotmail) may be used to send out SPAM.
3. **Other free services.** Some websites have the functionality to send out an email like "Send to a friend". This kind of functions may be used to send spam as well.
4. **Open Proxies.** These are compromised or miss configured servers that may be used to redirect SPAM to Internet using your Organization's resources. This kind of servers may be sold just like botnets.
5. **Stolen IP's.** Spammers may take control of Internet Addresses by illegal means, using them for criminal purposes.

Some of the most common techniques used by spammers to cheat users and anti-spam filters are the following:

1. **HTML Tricks.** By using HTML code an attacker may manipulate the email making it look in certain way to the user and in another to the anti-spam filter. For example, by using hidden html code like javascript or comments which may be legitimate but invisible. The less sophisticated filters may be confused by the invisible text and therefore fail on the SPAM identification.
2. **Bayesian poisoning.** The email has large blocks of legitimate text trying to fool the anti-spam filter by decreasing the threshold of spam words in the full mail.
3. **Content morphing.** The sender alters the text of the SPAM and the headers to fool the less sophisticated anti-spam filter that look for well known fragments of text.
4. **Attachments and images.** Instead of sending the SPAM as text, the spammers send their content as images attached to mail. By changing only one or a few pixels on every image, the anti-spam engines won't be able to detect this kind of mails all at once, they'll need a pattern to capture them on a one-to-one basis. If the anti-spam filter doesn't count with an OCR technology, this attack will pass through.
5. **Forcing the secondary MX.** There are several mail domains that specify a secondary mail server in case the main server is not available. Some spammers will try this alternative hoping the anti-spam security will be less restrictive than the other servers.
6. **Protecting against IP reputation services.** When a zombie machine, which is part of a botnet, starts sending SPAM to the victim the reputation services may act automatically by blocking those connections that come from dynamic IP's, but this process may be difficult when the SPAM is sent from legitimate mail servers. Some botnets send SPAM through the mail servers of the zombie machine to take advantage of this situation.
7. **Hiding the "action" call.** Every time we click on a URL, we are making use of the "action" function or "<a...>" HTML tag which lets the web browser get to destination web server. Many anti-spam filters look for this kind of tags to intercept the URL the spammer wants to redirect you.

The best way to combat such threats is through anti-spam and content filters that contain one or more of the following mechanisms:

1. *Pattern based SPAM detection.* This is a slow detection but effective on the long term. It consists of a pattern containing signatures that specifically identify a unique mail (or mail content) that has previously been detected as SPAM. It is said to be slow because for every new SPAM sample there is a need to add a new signature. This is can also be considered as a reactive technique because a sample must be feed to be detected.
2. *Heuristic based SPAM detection.* This is a proactive technique because it can combine signatures for specific portions of the mail structure that combined together may detect completely new SPAM samples by determining how much a mail is similar to a SPAM.

3. *Image base SPAM detection.* When the SPAM content is not transferred as text but as an attached image or picture there are at least two ways to determine if such content may be SPAM or not. The first one is to take the HASH of the image and add it to the SPAM pattern, by comparing all images hashes in a mail with the stored hashes the Anti-SPAM engine may determine if one of them correspond to a SPAM. The second one involves OCR technologies which are designed to recognize text inside an image. Hash comparison is not very efficient because for any modification on the picture like a pixel, a new hash will need to be added to the pattern. OCR is more effective but it consumes more CPU, so more resources may be needed for avoiding delivery delays.
4. *Content Filter based SPAM detection.* This technique is a very good ally to any of the other detection methods. It consists in an engine that is capable to find certain words, phrases or string patterns inside the mail body, headers and in some cases in attachments. Its administration may not be a 100% efficient as it needs manual intervention for creating the detection rules, so in some cases it may be considered as reactive. However, it gives you the advantage to react quickly when your organization is being overwhelmed with several SPAM messages that are not included in any other detection technique yet. (like pattern, heuristic or image detection). Some of these filters may include conditions where you can also define the route of the message, this is, you may create rules to block incoming or outgoing mails depending on your needs.
5. *IP reputation based SPAM detection.* This is by far a “must” detection technique for proactive SPAM and other attacks detection. It is fast and reduces the use of resources. It is based on IP lists that have already been identified as SPAM sources. Once it has received a new MTA connection, it makes a query (usually using DNS or HTTP requests) to a local or cloud based service that contains a huge list of malicious IP addresses. If the address is found it automatically drops the connection without the mail having entered to your Organization. This technique is said to be the most effective because it doesn’t really matter if the mail is already detected as SPAM or not, the connection will be dropped saving your Organization a lot of resources.

1.3.2. DHA

Directory Harvest Attack are those kind of threats where a malicious MTA is sending several mails with random mailboxes trying to identify which of them really exist on your Organization. These lists may be sold to other companies for other purposes like SPAM campaigns or malicious activities.

You can detect such an attack because your MTA Server will start receiving mail for inexistent mailboxes such as john.smith@company.com, john_smith@company.com or jsmith@company.com.

The biggest problem with this is not actually for the attacker to know the real mailbox list of your Organization because this can be obtained by means of other SMTP techniques, the biggest problem is for the Organization to spend unnecessary resources in mails that cannot be delivered anyway, instead of using them for valid mail.

A good anti-DHA filter should be based on at least one of the following two characteristics:

1. Your Organization must manually control a list of all the valid mailboxes. The problem here is this is a very time consuming task and an updated list may not always be maintained on time, depending on the size of the company.
2. If your Organization has an LDAP where all valid mailboxes are registered, you can integrate the Anti-DHA filter to this service. This way you'll always have an updated list in real time without any additional manual intervention.

1.3.3. Malware

Using mail for sending malware is not that common as it used to be several years ago, however, it still remains as an important threat in attacks like APT's (Advanced Persistent Threat) where the intention is to target a specific company or employee for a very specific objective.

Even though the probability of being compromised by an attached malware is low, it is still necessary to count with this kind of protection may it be for security or audit reasons. Don't leave your Organization without an Antimalware solution for your mail infrastructure.

The main threats to be detected by this type of engines are: virus, spyware, hacking / cracking tools, key loggers, Trojans, worms and remote access tools amongst others.

1.3.4. Bounced Mail

Mails under this category are those automatically generated when a mail cannot be delivered to its final destination. The SMTP protocol established this mechanism in order for the sender to know if there were any problems with its mail delivery, however, an attacker could send several mails using the company he wants to attack as the sender's mailbox to any other inexistent recipient mailboxes in other MTA Servers over the Internet, these server will react to this situation by responding to each of this mails and sending the corresponding notification (bounced mail) to the attacked domain which at this time will receive hundreds or even thousands of mails in a very short time window. This technique is used mostly for generating DoS attacks.

Some administrators try to combat such attacks by blocking all mails that contain a null sender (from:<>), however, such rules will also block legitimate notification mails. There are more sophisticated filters that can detect this attack by establishing a monitor time window in which they define a threshold of the number of notifications they can receive from a single IP, this way, legitimate notifications can still be accepted.

1.3.5. Spoofing

This is a very common attack nowadays. It consists of a mail sent by an attacker who poses as different person or domain. For example, the attacker may send a mail with a sender like <user@cisco.com> with a mailbox that doesn't actually belong to CISCO in this case. As you'll see in the following chapters, this technique is very common because it is actually very easy to perform.

There are some techniques you can use to protect your Organization against this kind of threats:

1. SPF Filter. Sender Policy Framework is a protocol designed to avoid the reception of mails that illegally use a domain name that doesn't belong to them. These kind of filters first extract the domain part of the mailbox used in the HELO and/or the MAIL commands, then they obtain the list of IP addresses authorized to use that domain by querying the DNS server of the domain. If the connecting IP is not listed in the resultant list, then the connection is dropped. Because this is still an "experimental" protocol, some unexpected results may be found when the comparison is done, in such situations the final action must be decided by the MTA implementing the filter. One of its main advantages is that is easy and economic to implement, Postfix for example only needs to enable certain libraries and parameters to start the IP verification. If the Organization is willing to protect its domain name to prevent its use by unauthorized servers, you just need to publish an SPF record in your public DNS server and you'll be ready to protect your domain usage in all servers that make use of such filters. In Microsoft Exchange Servers this option is available in the Edge Servers under an improved version called Sender-ID that will explain next.
2. Sender-ID Filter. This is a Microsoft technology and is available by default on the Microsoft Exchange servers family. Sender-ID or SID is based on SPF, the main difference between them is an algorithm called PRA (Purported Address) that can identify the last mailbox that really sent the mail to your Organization. Its main disadvantage, which is the same as with SPF, is that this is still and "experimental" protocol with the same limitations already mentioned for SPF. You can participate

on this implementations by just making public a similar TXT record in your public DNS. If you want to validate with SID and you don’t have an Exchange server, there are several modules and plug-ins you can use for several mail servers in Windows and Linux.

3. DKIM Filter. DomainKeys Identified Mail is another protocol that also helps Organizations avoiding the receptions of mails that have senders that illegally use a certain domain name. This is a standard protocol and this fact alone represents a big difference when comparing it against SPF or Sender-ID, this is because almost every possible situation and the corresponding actions are more accurately defined. You can be sure that a mail that has a verified DKIM signature comes from the source domain it is expected to be, however you can never be sure if such mails are not malicious or some kind of SPAM. Its implementation is not that simple and for both, generating and validating the signature you may have to modify or install new modules in your existing infrastructure.
4. Content Filters. If your Organization doesn’t have or doesn’t want to implement any new technology but you have a content filter in place, then you can still protect your users by creating rules that block any incoming mail that uses your Organization’s domain name, because this kind of mails should be processed by your internal mail server and should not arrive from Internet. The main disadvantage of these filters is that you can only create rules that protect your own domain name.

1.4. Summary

In this chapter we have explained the mechanisms used by an MTA server to perform a successful mail delivery by using the SMTP protocol. From the moment a user presses the Send button in its mail client, an ordered and coherent protocol is started that will allow both the sender and recipient MTA servers to transfer the necessary information.

There are several attacks an MTA server may face because of the universal nature of the protocol, and it is because of this fact, that no matter what mail server or operating system you use, they are all vulnerable at some degree. You must take this security vulnerabilities into account in order to implement the best methods for protecting your Organization.

In the following chapters we'll use this information to better understand the environment a mail server is daily exposed to in each of the transactions it performs.

Chapter 2. The SMTP Structure

In order to understand the way email vulnerabilities are used to attack your infrastructure, is necessary to deepen our knowledge on how email is operated and generated.

This chapter is dedicated to get familiar with the structure, protocols and functions used by the mail flow and use this knowledge to transform it in actions you may implement in your Organization.

We'll begin by studying which are the rules stated on the SMTP RFC Standars, these instructions must be obeyed no matter what MTA and platform you use.

Once we are familiar with the main steps in an SMTP transaction, we'll be ready to interpret each of the sections that compose the email. We'll study the structure from the most simple mails in plain text until the formation of more complex mails with attachments and alternative visible parts depending on the email client.

Finally we'll make a summary of the native vulnerabilities for SMTP and some recommendations we may apply to avoid them in your Organization.

IN THIS CHAPTER:

- ✓ The SMTP standards
- ✓ Email Structure
- ✓ Native SMTP Vulnerabilities
- ✓ Summary

2.1. The SMTP Standards

The SMTP Protocol emerged as a necessity to standardize the way different platforms communicated to send information, having this way a universal format any system could process and understand. In the following sections, we'll review what characteristics the mail flow must meet according to the three Standars that have defined SMTP along its history.

2.1.1. RFC 821

In August 1982 RFC 821 gets officially published to standardize the use of a protocol for information exchange which from now on will be known as Simple Mail Transfer Protocol. The concepts for this protocol implementation for any platform are established by this RFC.

This model represents the first definition of an email transfer over any network. The first point defines that the content to be used in the transfer will be composed exclusively of a very specific set called the US-ASCII characters. The basic model of such a transfer includes the following points:

[...]

3. THE SMTP PROCEDURES

```
[...]
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
<reverse-path> should be the host sending this command.
```

This definition indicates that a new mail transfer should begin with the MAIL command used to indicate its origin. This is still valid and we can confirm it with the following example:

```
220 ESMTP Postfix
MAIL FROM:<asdf@asdf.com>
250 2.1.0 Ok
```

In this example we can confirm the syntax is successfully accepted by the mail server when we use a mailbox as an argument for the MAIL FROM command. However, the definition also states that the mailbox used as argument is not just a mailbox but a “reverse-path”. This means the MAIL command accepts the full path a mail server should use to reach the sender when the mail cannot be successfully delivered.

Because of this definition, it is possible to define the list of servers to be used when returning the delivery failure notification to the sender. This syntax is defined in this RFC as a list of servers in which the first one is the one that originated the mail. This definition is now obsolete because it implies all mail servers should act as Open-Relay but it is still being accepted for compatibility issues even when it is not implemented. We can confirm this with the following example:

```
220 ESMTP Postfix
MAIL FROM:<@hub.rskala.com,@hub.gmail.com:rskala@rskala.com>
250 2.1.0 rskala@rskala.com....Sender OK
```

In this example we can appreciate the Reverse-Path syntax is still valid and the receiving mail server is still able to identify the sender’s mailbox in it. What the example says is that if this mail cannot be delivered, a notification should be sent to “rskala@rskala.com” by first passing through server hub.gmail.com and then to hub.rskala.com. For this process to be executed correctly, hub.gmail.com should be able to receive external mail for a different Internet domain than gmail, this behavior is usually known as Open-Relay because the server is acting just as a intermediate step in the final mail delivery between two external domains. Because this process permits the use of any mail server over the Internet to perform attacks (like sending a bunch of SPAM) with this mail server as the attacker, the new revision of the protocol deprecated this function.

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.
```

This definition indicates that each recipient's mailbox should be explicitly defined by an RCPT command. Let's say for example that you want to send a mail to three different mailboxes, you should send three RCPT commands, one for each of the mailboxes as shown in the following example:

```
MAIL FROM:<user@dominio.com>
250 2.1.0 user@dominio.com....Sender OK
RCPT TO:<user1@rskala.com>
250 2.1.5 user1@rskala.com
RCPT TO:<user2@rskala.com>
250 2.1.5 user2@rskala.com
RCPT TO:<user3@rskala.com>
250 2.1.5 user3@rskala.com
rcpt to:<user4@rskala.com>,<user5@rskala.com>
501 5.5.4 Invalid Address
```

From this example we can see how the first three RCPT commands are successfully accepted, each of this operations is confirmed by the receiving mail server with a numerical status code "250 2.1.5. user1@rskala.com" which indicates the SMTP Server has started the list of recipient mailboxes in its internal buffer. You can also confirm that if you try to send more than one mailbox in the same command, the server responds with a numerical status code "501 5.5.4 Invalid Address" which means the syntax of the mailbox is not the expected, the SMTP Server is not accepting any responsibility for the delivery of this mail to that specific mailbox or mailboxes.

The RCPT command defined by this RFC, just as it happens with the MAIL command, is designed to accept more than a mailbox but a "Forward-Path", this means the command accepts as an argument the full path to be used to reach the final recipient of the mail. Just as it happens with MAIL, this is now deprecated for the same reason but it is still for compatibility issues as shown in the following example. Remember that this behavior implies that all mail servers should act as Open-Relays.

```
rcpt to:<@hub.rskala.com,@hub.gmail.com:user1@rskala.com>
250 2.1.5 user1@rskala.com
```

This examples confirms the old RFC 821 syntax for this command is still accepted even when it is not actually implemented. This syntax indicates the mail should be delivered to user1@rskala.com by first sending it through hub.gmail.com and then to hub.rskala.com where the mailbox actually resides.

```
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 (see Section 4.5.2).

Please note that the mail data includes the memo header items such as
Date, Subject, To, Cc, From [2].


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.

[...]
```

This last section indicates the way the mail content should be transferred, its syntax and the parts the content is composed of. First of all, you can see the transfer is initiated when the DATA command is executed and mail server responds with the following code:

```
MAIL FROM:<user@domain.com>
250 2.1.0 user@domain.com....Sender OK
RCPT TO:<user1@rskala.com>
250 2.1.5 user1@rskala.com
```

DATA
354 Start mail input; end with <CRLF>.<CRLF>

As stated by the definition, after the DATA command the SMTP Server responds with a numerical status code “354” and then just waits for the mail content. Once the full content is transferred, the command is closed by sending a single “dot” in a line alone (this syntax is defined as <CRLF>.<CRLF>) as shown below:

```
DATA
354 End data with <CR><LF>.<CR><LF>
from:user1
to:user2
subject:test

this is the content of the mail
.
250 2.0.0 Ok: queued as F30AFC7040D
```

After closing the command, the SMTP Server returns to the original command/reply state by sending a 250 code. This means the mail transfer has ended and now the SMTP Server has accepted full responsibility for this mail delivery. This section also states that inside the body transferred through this command, all headers should be included here. Headers should be the first lines of the content and should be separated by the body content by a blank line, just as in the previous example where we use the from, to and subject headers.

This RFC also establishes that before initiating any transfer operation the SMTP Client must identify itself by sending its name in the HELO command.

COMMAND SEMANTICS

```
[...]

HELLO (HELO)

This command is used to identify the sender-SMTP to the receiver-SMTP.
The argument field contains the host name of the sender-SMTP.
The receiver-SMTP identifies itself to the sender-SMTP in the connection
greeting reply, and in the response to this command.

This command and an OK reply to it confirm that both the sender-SMTP and
the receiver-SMTP are in the initial state, that is, there is no
transaction in progress and all state tables and buffers are cleared.
```

[...]

This section defines for the first time the use of the HELO command which is used to identify both the SMTP Client and Server with their corresponding names. The definition also states that this must be the first command to send in any SMTP transaction, this must be sent even before the MAIL command. This is still valid and must be respected by all MTA servers. The following examples show how the command is implemented and its order relative to MAIL.

```
220 hub2.rskala.com ESMTP
HELO hub1.rskala.com
250 hub2.rskala.com Hello [192.168.0.89]

220 hub2.rskala.com ESMTP
MAIL FROM:<user@dominio.com>
503 5.5.2 Send hello first
```

With this information we are able to identify that no matter the platform, the following parts are always present:

1. **SMTP Sender.** It has the responsibility to establish the network connection to the SMTP Receiver for which the mail is destined to. Once the connection is established and the SMTP Receiver has greeted with the 220 code, it is its responsibility to initiate the whole conversation until the email transfer has been completed. If the SMTP Sender doesn't initiate the conversation or it stops it at any given time, the SMTP Receiver has the right to end the conversation when a certain time threshold has been exceeded, in any other situation the SMTP Receiver must maintain the conversation open until the SMTP Sender wants to continue.
2. **SMTP Receiver.** Once an SMTP Sender has successfully established a TCP Connection on port 25, the SMTP Receiver must respond with a 220 code indicating it is willing to establish a conversation. Its responsibility is to give an answer to each command sent by the SMTP Sender and if the conversation is satisfactory, accept and deliver the email to the final recipient.
3. **SMTP Conversation.** This is a series of SMTP ordered commands and replies that together establish the rules under the email transfer takes place.

Another key concept in this RFC is the set of components that composes an email.

1. **ENVELOPE.** This section is defined by the MAIL FROM and RCPT TO commands. This information is used by the MTA to identify the route it must follow to find the recipient and in case of a failure, identify the return path to the original sender. The RFC defined a syntax for this command by which the SMTP Sender first provides a list of mail servers (starting by the SMTP Sender itself) through which the email must pass. This feature implied that ALL the MTA server should be open-relays and send email on behalf of any other MTA.
2. **HEADERS.** These are initiated immediately after the DATA command and contain all the visible and invisible information for the recipient. Such headers may be: From, To, BCC, CC, Date, Message-ID, etc. This block ends after sending two consecutive <CRLF> or "ENTER".
3. **BODY.** The mail body begins right after the Headers block has finished. This contains all the readable information for the recipient. If the mail contains any attachments, these must be sent before the body blocks is closed. When the mail body has been successfully transferred, the mail is closed by the following syntax: <CRLF>.<CRLF> or "ENTER".(dot)"ENTER".
4. **ATTACHMENTS.** If the mail is to have any attachments then the headers should indicate the content is compatible with the MIME format and it must identify what is the boundary the MTA should use to separate the different attachments contained in the body. Attachments are transmitted as part of the body after the readable sections had ended. In order to separate the different attachments from one another, the MTA uses the boundary text to identify the beginning and the end of each part. Once the transfer has ended the mail body is closed as described above.

2.1.2. RFC 2821

In April 2001, the RFC 2821 obsoletes 821 in order to counteract the rising of email threats. The main differences are described below.

1. The Extended SMTP.

2.2.1 Background

[...]

> In an effort that started in 1990, approximately a decade after RFC821
> was completed, the protocol was modified with a "service
> extensions" model that permits the client and server to agree to

> utilize shared functionality beyond the original SMTP requirements. The SMTP extension mechanism defines a means whereby an extended SMTP client and server may recognize each other, and the server can inform the client as to the service extensions that it supports.

2. EHLO substitutes HELO in the new ESMTP Framework.

2.2.1 Background

[...]

> Contemporary SMTP implementations MUST support the basic extension mechanisms. For instance, servers MUST support the EHLO command even if they do not implement any specific extensions and clients SHOULD preferentially utilize EHLO rather than HELO. (However, for compatibility with older conforming implementations, SMTP clients and servers MUST support the original HELO mechanisms as a fallback.) Unless the different characteristics of HELO must be identified for interoperability purposes, this document discusses only EHLO.

Sections 1 and 2 indicate the basic mechanisms like MAIL and RCPT are still available but there are new ones incorporated in the new framework. ESMTP now gives the option to incorporate new service extensions . This can be confirmed when you send EHLO instead of HELO when you are initially greeting the MTA server.

```
EHLO hub.rskala.com
250-hub1.rskala.com Hello [192.168.0.89]
250-TURN
250-SIZE
250-ETRN
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-8bitmime
250-BINARYMIME
250-CHUNKING
250-VRFY
250-X-EXPS GSSAPI NTLM LOGIN
250-X-EXPS=LOGIN
250-AUTH GSSAPI NTLM LOGIN
250-AUTH=LOGIN
250-X-LINK2STATE
250-XEXCH50
250 OK
```

The SMTP Client uses this list to know exactly which services are supported by the SMTP Server. The main reason for replacing EHLO with HELO is that with HELO, the SMTP Client will never know which services are available until it tries one of them and the service is actually called or refused. With EHLO, any mail server can know at the moment what services and mechanisms are available for execution. Let’s suppose for example that a mail server wants to establish a TLS session before to securely transmit a message. If it chooses to issue a HELO command, it doesn’t really know if the SMTP Server is really configured to start this kind of communication, so it will just send a “blind” STARTTLS and wait for the response. If it uses EHLO, it can search the STARTTLS command and if it appears listed then the SMTP Client can be sure a TLS conversation can be implemented. Even when HELO is no longer recommended, ALL mail servers should still accepted for compatibility issues.

3. The integration of new extensions that allow the development of new MTA features.

2.2.2 Definition and Registration of Extensions

```
[...]

   In addition, any EHLO keyword value starting with an upper or lower
   case "X" refers to a local SMTP service extension used exclusively
   through bilateral agreement.  Keywords beginning with "X" MUST NOT be
   used in a registered service extension.  Conversely, keyword values
   presented in the EHLO response that do not begin with "X" MUST
   correspond to a standard, standards-track, or IESG-approved
   experimental SMTP service extension registered with IANA.  A
   conforming server MUST NOT offer non-"X"-prefixed keyword values that
   are not described in a registered extension.
[...]
```

This new definition allows each mail server to implement local mechanisms and services without them to be part of an official standard, the only condition is to explicitly present them in the EHLO response with the “X-“ prefix. The following table shows the list of the EHLO responses from two different servers.

Exchange 2003	Postfix
250-TURN 250-SIZE 250-ETRN 250-PIPELINING 250-DSN 250-ENHANCEDSTATUSCODES 250-8bitmime 250-BINARYMIME 250-CHUNKING 250-VRFY 250-X-EXPS GSSAPI NTLM LOGIN	250-PIPELINING 250-SIZE 10240000 250-VRFY 250-ETRN 250-STARTTLS 250-ENHANCEDSTATUSCODES 250-8BITMIME 250 DSN

250-X-EXPS=LOGIN 250-AUTH GSSAPI NTLM LOGIN 250-AUTH=LOGIN 250-X-LINK2STATE 250-XEXCH50 250 OK	

Table 1. EHLO responses in Exchange 2003 and Postfix

4. Several old and problematic commands become obsolete and even forbidden to increase the MTA security.

```
Appendix F. Deprecated Features of RFC 281.

[...]

F.1 TURN. The SMTP Client cannot force the SMTP Server to change its
role.
F.2 SOURCE ROUTING. The MAIL FROM and RCPT TO feature that allowed MTA
servers to become open-relays is forbidden, the new syntax includes only
the sender and recipient mailboxes.
F.3 HELO. It is recommended to use only EHLO, even every MTA must
continue to respond to the HELO command for legacy implementations.
F.4 #-literals. RFC 821 allowed a mailbox to be described as a sequence
of numbers separated by dots and preceded by the "#" symbol. This syntax
become obsolete by this RFC.
F.5 DATES AND YEARS. Dates must be composed by four digits. Using only
two digits becomes obsolete.
F.6 SENDING VERSUS MAILING. The SEND, SAML and SOML commands become
obsolete and should not be used anymore.
```

5. Terminology. SMTP Client and SMTP Servers will be referenced from now on as MTA (Mail Transfer Agent), given the fact that any of these devices may have both roles. Those devices or applications that allow an end user to create / receive email is known as a Mail User Agent (MUA or just UA).

6. Codification. In spite of the protocol being very strict on the exclusive use of ASCII characters on any email transaction, it is allowed now the character codification in order to send character sets different to the US-ASCII in MIME format.

7. The strict use of FQDN.

```
2.3.5 Domain

...The domain name, as described in this document and in [22], is the
```

```
   entire, fully-qualified name (often referred to as an "FQDN").  A
   domain name that is not in FQDN form is no more than a local alias.
   Local aliases MUST NOT appear in any SMTP transaction....
```

At this point is worth mentioning that the use the use of a domain name in the HELO/EHLO/MAIL/RCPT commands, requires for such name to be a complete FQDN. The following examples shows situations in some Exchange and Postfix versions that do not check this condition by default. This condition may result in problems like mail tracing, attack identification or even unexpected results because some mail servers will try to auto complete these parts with its own.

```
EHLO a
250 hub.rskala.com Hello [192.168.0.89]
MAIL FROM:<a>
250 2.1.0 Sender OK
RCPT TO:<a>
250 2.1.5 a@rskala.com
```

8. Sizes and timeouts. Any MTA must process emails that satisfy at least the following sizes. Any MTA will be able to send email that exceed these limits but it should be prepared to get the mail rejected by the SMTP Server in case this one is not able to process it.

```
 4.5.3.1 Size limits and minimums

local-part
   The maximum total length of a user name or other local-part is 64
characters.


domain
   The maximum total length of a domain name or number is 255 characters.


path
   The maximum total length of a reverse-path or forward-path is 256
characters (including the punctuation and element separators).


command line
   The maximum total length of a command line including the command word
and the <CRLF> is 512 characters. SMTP extensions may be used to increase
this limit.


reply line
   The maximum total length of a reply line including the reply code and
```

the <CRLF> is 512 characters. More information may be conveyed through multiple-line replies.

text line

The maximum total length of a text line including the <CRLF> is 1000 characters (not counting the leading dot duplicated for transparency).

message content

The maximum total length of a message content (including any message headers as well as the message body) MUST BE at least 64K octets. SMTP server systems that must impose restrictions SHOULD implement the "SIZE" service extension [18], and SMTP client systems that will send large messages SHOULD utilize it when possible.

recipients buffer

The minimum total number of recipients that must be buffered is 100 recipients. Rejection of messages (for excessive recipients) with fewer than 100 RCPT commands is a violation of this specification.

The general principle that relaying SMTP servers MUST NOT, and delivery SMTP servers SHOULD NOT, perform validation tests on message headers suggests that rejecting a message based on the total number of recipients shown in header fields is to be discouraged. A server which imposes a limit on the number of recipients MUST behave in an orderly fashion, such as to reject additional addresses over its limit rather than silently discarding addresses previously accepted. A client that needs to deliver a message containing over 100 RCPT commands SHOULD be prepared to transmit in 100-recipient "chunks" if the server declines to accept more than 100 recipients in a single message.

Errors due to exceeding these limits may be reported by using the reply codes. Some examples of reply codes are:

```
   500 Line too long.
or
   501 Path too long
or
   452 Too many recipients  (see below)
or
   552 Too much mail data.
```

RFC 821 [30] incorrectly listed the error where an SMTP server exhausts its implementation limit on the number of RCPT commands ("too many recipients") as having reply code 552. The correct reply code for this condition is 452. Clients SHOULD treat a 552 code in this case as a temporary, rather than permanent, failure so the logic below works.

When a conforming SMTP server encounters this condition, it has at least 100 successful RCPT commands in its recipients buffer. If the server is able to accept the message, then at least these 100 addresses will be removed from the SMTP client's queue. When the client attempts retransmission of those addresses which received 452 responses, at least 100 of these will be able to fit in the SMTP server's recipients buffer. Each retransmission attempt which is

```
   able to deliver anything will be able to dispose of at least 100 of
   these recipients.

   If an SMTP server has an implementation limit on the number of RCPT
   commands and this limit is exhausted, it MUST use a response code of
   452 (but the client SHOULD also be prepared for a 552, as noted
   above).  If the server has a configured site-policy limitation on the
   number of RCPT commands, it MAY instead use a 5XX response code.
   This would be most appropriate if the policy limitation was intended
   to apply if the total recipient count for a particular message body
   were enforced even if that message body was sent in multiple mail
   transactions.
```

This last definition may be a little confusing because it states that all MTA servers should limit the RCPT commands to at least 100 recipients, however, at the end of the text allows the reception of mails that overpass this limit. To make this explanation clearer we'll show here an example of a server that limits the number of recipients to two and we'll send a message to five. As a result you'll see how this text is implemented in real life scenarios.

For this example let's configure a Postfix server to limit the number of recipients to only two by executing the following command:

```
postconf -e smtpd_recipient_limit=2
postfix reload
```

Now let's send a mail with 5 recipients. In order for this rule to apply, the SMTP Client will have re-send the original mail in three copies to maintain the rule that a single mail should only have two recipients. The following lines show the SMTP conversation when the mail is received by Postfix for the first time.

```
MAIL FROM:<user@domain.com> SIZE=2187
250 2.1.0 Ok
RCPT TO:<user1@rskala.com>
250 2.1.5 Ok
RCPT TO:<user2@rskala.com>
250 2.1.5 Ok
RCPT TO:<user3@rskala.com>
452 4.5.3 Error: too many recipients
RCPT TO:<user4@rskala.com>
452 4.5.3 Error: too many recipients
RCPT TO:<user5@rskala.com>
452 4.5.3 Error: too many recipients
DATA
```

From this conversation we can see the first two RCPT commands being successfully accepted with a status code 250 and therefore, this mail will be delivered to the first two recipients. If we continue to see the packet capture we'll see the following lines:

```
250 2.0.0 Ok: queued as 9D579C70453
MAIL FROM:<user@domain.com> SIZE=2187
```

```
250 2.1.0 Ok
RCPT TO:<user3@rskala.com>
250 2.1.5 Ok
RCPT TO:<user4@rskala.com>
250 2.1.5 Ok
RCPT TO:<user5@rskala.com>
452 4.5.3 Error: too many recipients
DATA
```

The first line just confirms that Postfix has accepted responsibility for delivering this mail to the first two recipients. Instead of closing the SMTP session, our SMTP Client starts a new message transfer operation to deliver the same mail for the remaining users 3, 4 and 5. You can see that now user5 gets rejected again with a status code "452 4.5.3 Error: too many recipients". Because of this, user5 will have to wait for a third try as we see in the final section of the capture.

```
250 2.0.0 Ok: queued as A2ACEC7045C
MAIL FROM:<user@domain.com> SIZE=2187
250 2.1.0 Ok
RCPT TO:<user5@rskala.com>
250 2.1.5 Ok
DATA
```

At this point we have confirmed that three exact copies of the same mail have been generated to successfully deliver the mail to the five original recipients. If we check the Message-ID header of all these mails we can confirm that all of them are in effect the same mail. This confirms that in order to comply with this section of the RFC our SMTP Client had to triplicate the same message.

```
User1: Message-ID: <1C6DA6E388B1184BABFC02524413B32037C6>
User2: Message-ID: <1C6DA6E388B1184BABFC02524413B32037C6>
User3: Message-ID: <1C6DA6E388B1184BABFC02524413B32037C6>
User4: Message-ID: <1C6DA6E388B1184BABFC02524413B32037C6>
User5: Message-ID: <1C6DA6E388B1184BABFC02524413B32037C6>
```

Now that we have confirmed that this rule actually takes place when delivering mail in these conditions we'll try to explain why the RFC demands this kind of behavior.

When we talk about the mail responsibility in both RFC 821 and 2821, it is defined that an SMTP Server should start storing all the RCPT commands that have been already successfully accepted in an internal buffer. In our example this implies that the first time when we tried to deliver the mail for all five recipients, at least both of them had already been successfully accepted and therefore there's no reason why this two recipients should not receive the said mail, remember that they are already saved in the internal buffer! The note on this section clearly states that mail servers that either send or receive mail shouldn't reject a mail based on the number of recipients that appear on the header section because by that time the mail has already been accepted for those recipients. Because of this logic, the first two recipients should receive the mail.

The note also states that when a RCPT command is rejected for exceeding the recipient limit, a temporary error code should be used and not a permanent one. This condition causes the SMTP Client to queue the original copy for the remaining recipients until it can successfully send the mail to all recipients.

```
RFC821(Limit to the number of recipients)
4.5.3. SIZES
         Recipients buffer

         The maximum total number of recipients that must be buffered is
         100 recipients. Errors due to exceeding these limits may be
         reported by using the reply codes, for example:

            500 Line too long.

            501 Path too long

            552 Too many recipients.

            552 Too much mail data.
```

This is the original text that was deprecated by RFC 2821. The problem with this definition is that by reporting a permanent error (5XX) the rejected mailboxes would never receive a copy of the mail which is incoherent, because at least the first two recipients (in our example) would be able to receive the original mail. Because of this, RFC 2821 no redefines the response as temporal and not permanent.

As you can see, as an implicit result of all this, one single mail could appear as duplicated several times in the mail server logs, but at least, by knowing this basic principle will help you understand that this is not actually a problem but an expected condition. As an advice, don't impose a recipient limit at the SMTP level, if you must enforce such a policy then do it at a policy level using either your mail server or your anti-spam solution.

```
4.5.3.2 Timeouts

      Initial 220 Message: 5 minutes
      MAIL Command: 5 minutes
      RCPT Command: 5 minutes
      DATA Command: 2 minutes
      DATA Block: 3 minutes
      DATA Termination: 10 minutes.
```

9. Re-Sending the mail. The SMTP Client should implement the proper mechanisms to re-send the mail that could not be delivered because of a 4XX code error. The minimum waiting period for retransmission should be of at least 30 minutes and the MTA should try this operation for a period between 4 to 5 days.

> The sender MUST delay retrying a particular destination after one attempt has failed. In general, the retry interval SHOULD be at least 30 minutes; however, more sophisticated and variable strategies will be beneficial when the SMTP client can determine the reason for non-delivery. Retries continue until the message is transmitted or the sender gives up; the give-up time generally needs to be at least 4-5 days. The parameters to the retry algorithm MUST be configurable.

This definition implies that whenever an SMTP Client is not able to deliver a mail because of transient error 4XX, the server must queue the mail and retry the delivery later for period between 4 to 5 days but this can be modified. According to this definition, when a permanent error code 5XX is received, the mail MUST NOT be queued, instead, a no-delivery notification should be sent using as the recipient the mailbox received in the MAIL command.

10. Address Resolution. Every time the SMTP Client identifies the domain to which it should deliver mail, the first step should be the MX RR resolution in order to obtain the SMTP Server IP (TYPE A). Nevertheless, there might be some situations where the MX record is not present on the DNS definition or it may be wrongly defined. In such situations there are alternative solutions that might be implemented, however these situations may result in other kind of errors, and for this reason it is always recommended that the MX record be always defined following these guidelines:

> The names are expected to be fully-qualified domain names (FQDNs): mechanisms for inferring FQDNs from partial names or local aliases are outside of this specification and, due to a history of problems, are generally discouraged. The lookup first attempts to locate an MX record associated with the name. If a CNAME record is found instead, the resulting name is processed as if it were the initial name. If no MX records are found, but an A RR is found, the A RR is treated as if it was associated with an implicit MX RR, with a preference of 0, pointing to that host. If one or more MX RRs are found for a given name, SMTP systems MUST NOT utilize any A RRs associated with that name unless they are located using the MX RRs; the "implicit MX" rule above applies only if there are no MX records present. If MX records are present, but none of them are usable, this situation MUST be reported as an error.

Once the MTA has the MX record list ordered by priority, the appropriate record should be chosen in order to obtain the corresponding IP Address. Whenever more than one type A record is found for the same MX record, the proper mail server will be chosen randomly. If more than one MX record is found with the same priority, the appropriate record will be chosen randomly using the round-robbin algorithm.

11. Notifications and responsibilities. From the moment the SMTP Server responds with a 250 code to the mail body closure, it has the full responsibility for that mail delivery. If an error has occurred and the SMTP Server responds with an error to the DATA command, the responsibility for delivering that mail remains on the SMTP Client, which will be forced to internally queue the mail and retry the transmission in another moment, or notify the original sender using the return-path mailbox. All notifications must use the NULL mailbox "<>" as the sender, this way the SMTP Server knows this mail is in fact a notification and not a mail sent by a user. Every time an MTA receives a mail with a null sender "<>" it must not respond with a notification in case such mail cannot be delivered as well, this behavior prevents loops between MTA Servers.

2.1.3. RFC 5321

In October 2008 the last and most recent revision to the SMTP Protocol was made. This is still a draft RFC. Even when it doesn’t include any vital modifications, it offers a more comprehensive explanation on some sections like relay and the right use of file transfers using MIME types.

Along with this RFC, the 5322 was also published in order to update the structure of the IMF (Internet Message Format). This revision offers several modification that improve the usage of IMF on other protocols like DKIM.

The RFC documents give us the knowledge we need to interpret and analyze the data transmission through SMTP Protocol, the format they must obey, the order every mail part must be in and the function of all the commands used in the SMTP conversation. Nevertheless, this information cannot be contained in a single RFC document. If we are to study the full operation of the protocol, we must study other RFC documents to complete this knowledge.

2.2. Email Structure

For the digital information contained in any email to be transferred among MTA servers, the rules provided in several standards must be followed. Any non compliant mail is at risk of being rejected by a more strict MTA server.

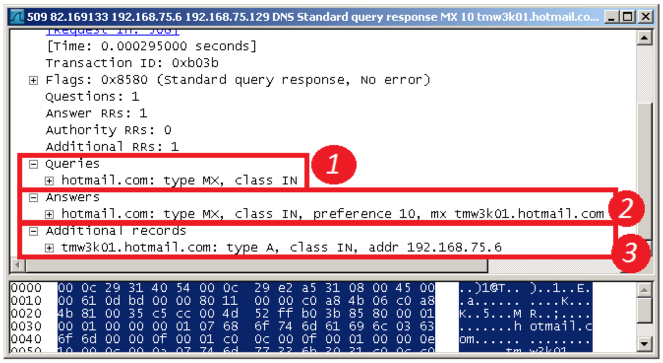
To begin the email analysis, we'll divide this section as follows:

1. Handshake
2. Envelope
3. Headers
4. Body
5. Attachments
6. Reply/Error Codes

2.2.1. Handshake

The first process that takes place when sending an email has to do with the process to establish communication between the SMTP Client and the SMTP Server. This process starts from the moment the SMTP Client has a new mail that has to be transmitted. In order to do this, the SMTP Client first identifies which is the domain the mail should go to and to determine this, it queries its DNS Server for the corresponding MX registries.

You can see this phase in the next picture where an MTA has a new mail and wants to obtain the server information for the hotmail domain.



509 82.169133 192.168.75.6 192.168.75.129 DNS Standard query response MX 10 tmw3k01.hotmail.co...

```
    [Time: 0.000295000 seconds]
    Transaction ID: 0xb03b
 ⊞  Flags: 0x8580 (Standard query response, No error)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 1
 ⊟  Queries
     ⊞ hotmail.com: type MX, class IN
 ⊟  Answers
     ⊞ hotmail.com: type MX, class IN, preference 10, mx tmw3k01.hotmail.com
 ⊟  Additional records
     ⊞ tmw3k01.hotmail.com: type A, class IN, addr 192.168.75.6

0000  00 0c 29 31 40 54 00 0c  29 e2 a5 31 08 00 45 00   ..)1@T.. )..1..E.
0010  00 61 0d bd 00 00 80 11  00 00 c0 a8 4b 06 c0 a8   .a...... ....K...
0020  4b 81 00 35 c5 cc 00 4d  52 ff b0 3b 85 80 00 01   K..5...M R..;....
0030  00 01 00 00 00 01 07 68  6f 74 6d 61 69 6c 03 63   .......h otmail.c
0040  6f 6d 00 00 0f 00 01 c0  0c 00 0f 00 01 00 00 0e   om...... ........
```

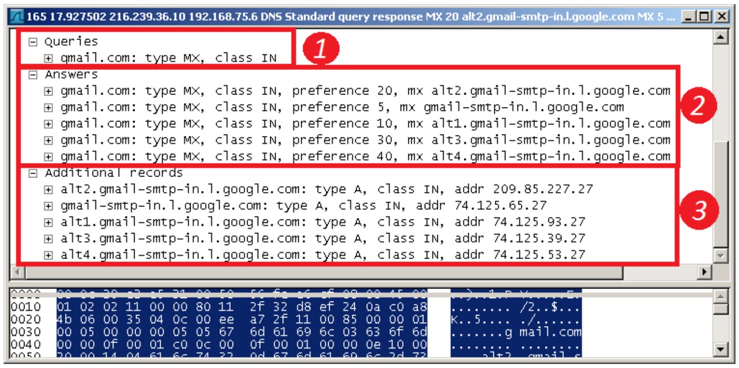
Image 3. Single MX record resolution

In this capture we can identify the following main items:

1. A DNS query has been made for the MX resource belonging to the hotmail.com domain.

2. The DNS server searches its own entries until it finds the requested resource and answers with resource found for the hotmail.com domain. Note that if the hotmail.com were not to be local to the DNS, then it should forward the query to another DNS server until it finds the corresponding or declare it as non-existent.
3. As an additional record, the DNS also sends the A record for the MX server found in the previous step, this way the MTA will have both records at once. Note that this is not always the case as some DNS responses include on the MX record and a second one must be made to obtain the A record.

In this case, it is very easy to manage this information because there is only one MX record and there is only one IP related to it. But in cases where the mail infrastructure is much bigger, this resolution is not that simple. Let's suppose now that we have more than one MX records for the same domain as shown in the following image.



```
165 17.927502 216.239.36.10 192.168.75.6 DNS Standard query response MX 20 alt2.gmail-smtp-in.l.google.com MX 5 ...
⊟ Queries
  ⊞ gmail.com: type MX, class IN
⊟ Answers
  ⊞ gmail.com: type MX, class IN, preference 20, mx alt2.gmail-smtp-in.l.google.com
  ⊞ gmail.com: type MX, class IN, preference 5, mx gmail-smtp-in.l.google.com
  ⊞ gmail.com: type MX, class IN, preference 10, mx alt1.gmail-smtp-in.l.google.com
  ⊞ gmail.com: type MX, class IN, preference 30, mx alt3.gmail-smtp-in.l.google.com
  ⊞ gmail.com: type MX, class IN, preference 40, mx alt4.gmail-smtp-in.l.google.com
⊟ Additional records
  ⊞ alt2.gmail-smtp-in.l.google.com: type A, class IN, addr 209.85.227.27
  ⊞ gmail-smtp-in.l.google.com: type A, class IN, addr 74.125.65.27
  ⊞ alt1.gmail-smtp-in.l.google.com: type A, class IN, addr 74.125.93.27
  ⊞ alt3.gmail-smtp-in.l.google.com: type A, class IN, addr 74.125.39.27
  ⊞ alt4.gmail-smtp-in.l.google.com: type A, class IN, addr 74.125.53.27

0010  01 02 02 11 00 00 80 11  2f 32 d8 ef 24 0a c0 a8   ........ /2..$...
0020  4b 06 00 35 04 0c 00 ee  a7 2f 11 00 85 00 00 01   K..5.... ./......
0030  00 05 00 00 00 05 05 67  6d 61 69 6c 03 63 6f 6d   .......g mail.com
0040  00 00 0f 00 01 c0 0c 00  0f 00 01 00 00 0e 10 00   ........ ........
```

Image 4. Multiple MX records resolution

In this case, we can see the number of answers and additional records grows up to 4 on steps 2 and 3. This situation occurs when the organization that owns the domain has more than one single server to receive mail from the Internet, which gives them more bandwidth to receive more mails, or will allow them to have High Availability infrastructure where more than one server are available to keep receiving mail when another fails.

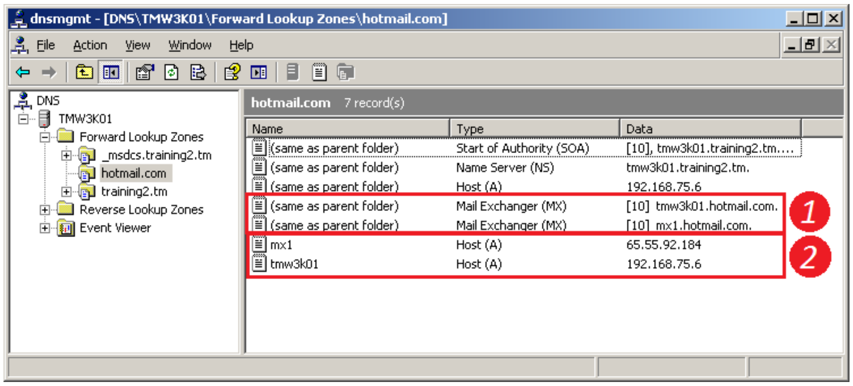
For this scenarios to work correctly, it is necessary to define the MX resolution priority. From the las answer we can see that there are 5 mail servers and each of them has different priorities (5, 10, 20, 30 and 40) being the highest the one with the lowest number (5 in this example). This means that whenever we want to send mail for the domain gmail.com, the mail server that will receive our mail will be:

gmail-smtp-in.l.google.com *with a priority of 5*

If this server fails, then the next server to contact will be:

alt1.gmail-smtp-in.l.google.com *with a priority of 10*

And so on until there are no more servers left, in which case the process will start with the highest priority again. Now, what happens when there are more than one record with the same priority? The following image shows the zone for the hotmail.com domain where you can see two records with the same priority.



dnsmgmt - [DNS\TMW3K01\Forward Lookup Zones\hotmail.com]

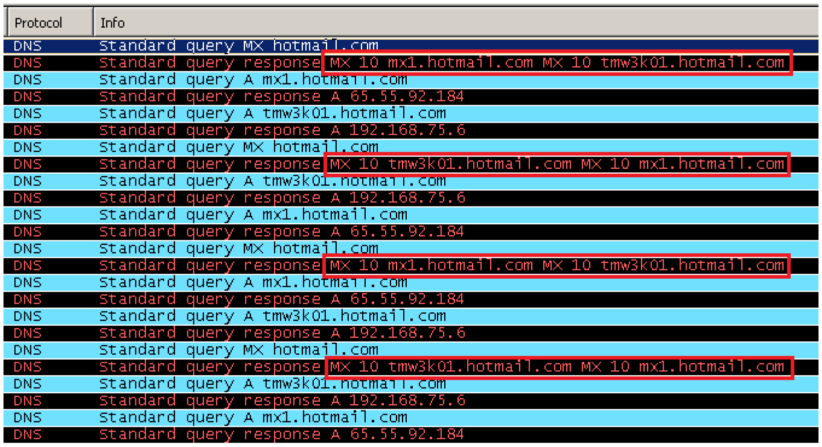
hotmail.com 7 record(s)

Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[10], tmw3k01.training2.tm....
(same as parent folder)	Name Server (NS)	tmw3k01.training2.tm.
(same as parent folder)	Host (A)	192.168.75.6
(same as parent folder)	Mail Exchanger (MX)	[10] tmw3k01.hotmail.com.
(same as parent folder)	Mail Exchanger (MX)	[10] mx1.hotmail.com.
mx1	Host (A)	65.55.92.184
tmw3k01	Host (A)	192.168.75.6

Image 5. Definition for MX records with the same priority

Here you can see both MX records with the same priority of [10] (tmw3k01.hotmail.com and mx1.hotmail.com). For each MX record there must be an associated A record.

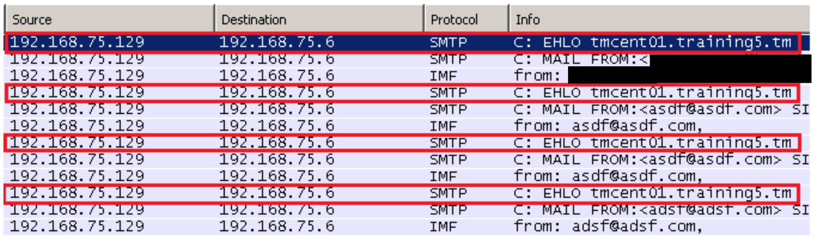
Now let’s see how the resolution takes place when you try to send a mail for this domain:



Protocol	Info
DNS	Standard query MX hotmail.com
DNS	Standard query response MX 10 mx1.hotmail.com MX 10 tmw3k01.hotmail.com
DNS	Standard query A mx1.hotmail.com
DNS	Standard query response A 65.55.92.184
DNS	Standard query A tmw3k01.hotmail.com
DNS	Standard query response A 192.168.75.6
DNS	Standard query MX hotmail.com
DNS	Standard query response MX 10 tmw3k01.hotmail.com MX 10 mx1.hotmail.com
DNS	Standard query A tmw3k01.hotmail.com
DNS	Standard query response A 192.168.75.6
DNS	Standard query A mx1.hotmail.com
DNS	Standard query response A 65.55.92.184
DNS	Standard query MX hotmail.com
DNS	Standard query response MX 10 mx1.hotmail.com MX 10 tmw3k01.hotmail.com
DNS	Standard query A mx1.hotmail.com
DNS	Standard query response A 65.55.92.184
DNS	Standard query A tmw3k01.hotmail.com
DNS	Standard query response A 192.168.75.6
DNS	Standard query MX hotmail.com
DNS	Standard query response MX 10 tmw3k01.hotmail.com MX 10 mx1.hotmail.com
DNS	Standard query A tmw3k01.hotmail.com
DNS	Standard query response A 192.168.75.6
DNS	Standard query A mx1.hotmail.com
DNS	Standard query response A 65.55.92.184

Image 6. Resolution of two MX records with the same priority

In the DNS resolution process we can see that both MX records alternate on every resolution that takes place on the DNS, however this doesn’t mean that the MTA will follow this order just as we can appreciate on the next image.

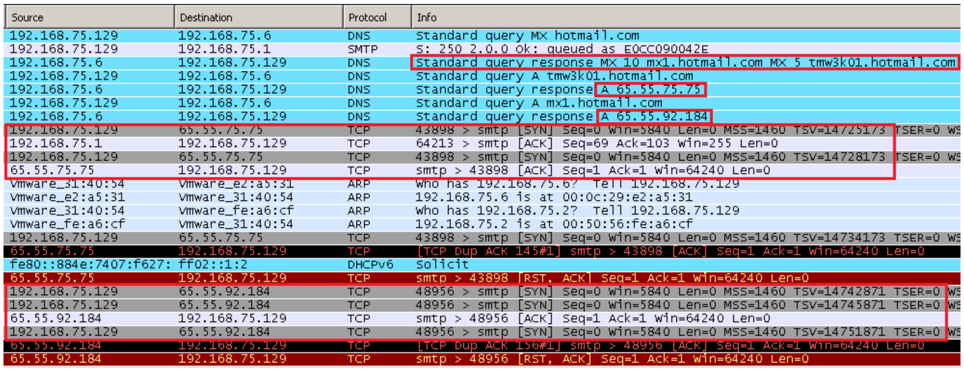


Source	Destination	Protocol	Info
192.168.75.129	192.168.75.6	SMTP	C: EHLO tmcent01.training5.tm
192.168.75.129	192.168.75.6	SMTP	C: MAIL FROM:<[redacted]
192.168.75.129	192.168.75.6	IMF	from: [redacted]
192.168.75.129	192.168.75.6	SMTP	C: EHLO tmcent01.training5.tm
192.168.75.129	192.168.75.6	SMTP	C: MAIL FROM:<asdf@asdf.com> SI
192.168.75.129	192.168.75.6	IMF	from: asdf@asdf.com,
192.168.75.129	192.168.75.6	SMTP	C: EHLO tmcent01.training5.tm
192.168.75.129	192.168.75.6	SMTP	C: MAIL FROM:<asdf@asdf.com> SI
192.168.75.129	192.168.75.6	IMF	from: asdf@asdf.com,
192.168.75.129	192.168.75.6	SMTP	C: EHLO tmcent01.training5.tm
192.168.75.129	192.168.75.6	SMTP	C: MAIL FROM:<adsf@adsf.com> SI
192.168.75.129	192.168.75.6	IMF	from: adsf@adsf.com,

Image 7. Email delivery for two MX records with the same priority

As we can see, the four mails are sent to the same MTA (tmw3k01.hotmail.com). This means that when you use the DNS as your balancer, the balance itself doesn’t take place as one would assume (one mail for one record, and the next mail to the next record). What actually happens is that the DNS is trying to advice the MTA to choose the records alternatively but the MTA will have the final word when choosing the next record. If you really want a load balancing to take place on your organization you should not use DNS balancing, use a load balancer instead.

Now, for our next example, what happens when the server with the highest priority is down? Let’s check out the next TCP capture so we can identify the way an MTA reacts under this circumstances.



```
Source                  Destination       Protocol  Info
192.168.75.129          192.168.75.6      DNS       Standard query MX hotmail.com
192.168.75.129          192.168.75.1      SMTP      S: 250 2.0.0 Ok: queued as E0CC090042E
192.168.75.6            192.168.75.129    DNS       Standard query response MX 10 mx1.hotmail.com MX 5 tmw3k01.hotmail.com
192.168.75.129          192.168.75.6      DNS       Standard query A tmw3k01.hotmail.com
192.168.75.6            192.168.75.129    DNS       Standard query response A 65.55.75.75
192.168.75.129          192.168.75.6      DNS       Standard query A mx1.hotmail.com
192.168.75.6            192.168.75.129    DNS       Standard query response A 65.55.92.184
192.168.75.129          65.55.75.75       TCP       43898 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14725173 TSER=0 WS
192.168.75.1            192.168.75.129    TCP       64213 > smtp [ACK] Seq=69 Ack=103 Win=255 Len=0
192.168.75.129          65.55.75.75       TCP       43898 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14728173 TSER=0 WS
65.55.75.75             192.168.75.129    TCP       smtp > 43898 [ACK] Seq=1 Ack=1 Win=64240 Len=0
Vmware_31:40:54         Vmware_e2:a5:31   ARP       Who has 192.168.75.6?  Tell 192.168.75.129
Vmware_e2:a5:31         Vmware_31:40:54   ARP       192.168.75.6 is at 00:0c:29:e2:a5:31
Vmware_31:40:54         Vmware_fe:a6:cf   ARP       Who has 192.168.75.2?  Tell 192.168.75.129
Vmware_fe:a6:cf         Vmware_31:40:54   ARP       192.168.75.2 is at 00:50:56:fe:a6:cf
192.168.75.129          65.55.75.75       TCP       43898 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14734173 TSER=0 WS
65.55.75.75             192.168.75.129    TCP       [TCP Dup ACK 145#1] smtp > 43898 [ACK] Seq=1 Ack=1 Win=64240 Len=0
fe80::884e:7407:f627:   ff02::1:2         DHCPv6    Solicit
65.55.75.75             192.168.75.129    TCP       smtp > 43898 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.75.129          65.55.92.184      TCP       48956 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14742871 TSER=0 WS
192.168.75.129          65.55.92.184      TCP       48956 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14745871 TSER=0 WS
65.55.92.184            192.168.75.129    TCP       smtp > 48956 [ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.75.129          65.55.92.184      TCP       48956 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=14751871 TSER=0 WS
65.55.92.184            192.168.75.129    TCP       [TCP Dup ACK 156#1] smtp > 48956 [ACK] Seq=1 Ack=1 Win=64240 Len=0
65.55.92.184            192.168.75.129    TCP       smtp > 48956 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
```

Image 8. MX record resolution when the highest priority is down

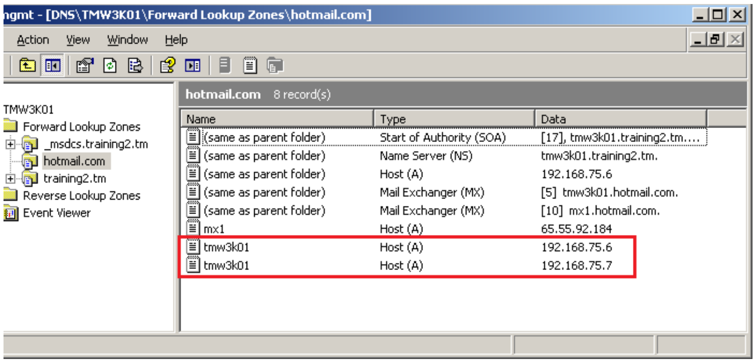
In this capture we can see there are two different priorities:

MX 5 tmw3k01.hotmail.com
MX 10 mx.hotmail.com

According to the SMTP protocol rules, the mail should be delivered to the server with the highest priority (in this example [5]) but from the capture we can see the mail is actually going to the second MX record 65.55.92.184. This is because the IP 192.168.75.76 is not responding to the TCP handshake as you can see in the gray row of the capture. Because the IP is unresponsive and the MTA cannot start a TCP connection to port 25, the MTA now reuses the information obtained previously from the DNS Server and changes to the next priority [10].

It is worth mentioning that the way an MTA chooses to change between servers of different priorities occurs only if the actual MX server is not responsive on the TCP port 25. If the port is up but the MTA application or the server itself is unresponsive, then the change won't take place and the SMTP Client will queue the mail to try and deliver it later.

Another way to have a High Availability infrastructure when you have more than one mail server is to associate several A records to the same host. This will force the MTA to choose one using the Round-Robbin algorithm. If the chosen IP is responsive on TCP Port 25, then the connection will take place, otherwise, the MTA will again choose another IP using the same algorithm until it finds one that responds to the TCP handshake on port 25.

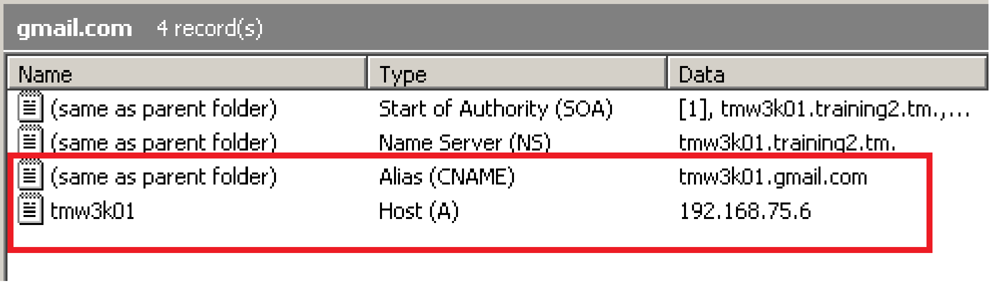


hotmail.com 8 record(s)

Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[17], tmw3k01.training2.tm.,...
(same as parent folder)	Name Server (NS)	tmw3k01.training2.tm.
(same as parent folder)	Host (A)	192.168.75.6
(same as parent folder)	Mail Exchanger (MX)	[5] tmw3k01.hotmail.com.
(same as parent folder)	Mail Exchanger (MX)	[10] mx1.hotmail.com.
mx1	Host (A)	65.55.92.184
tmw3k01	Host (A)	192.168.75.6
tmw3k01	Host (A)	192.168.75.7

Image 9. MX record with two different IP associated addresses

Another scenario included in RFC 2821 to establish a connection between MTA servers is the absence of the MX record but the existence of a CNAME or A record.

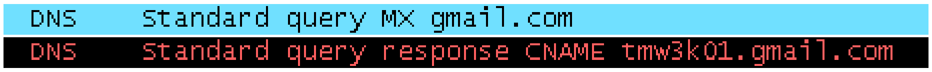


gmail.com 4 record(s)

Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[1], tmw3k01.training2.tm.,...
(same as parent folder)	Name Server (NS)	tmw3k01.training2.tm.
(same as parent folder)	Alias (CNAME)	tmw3k01.gmail.com
tmw3k01	Host (A)	192.168.75.6

Image 10. Definition of a CNAME record

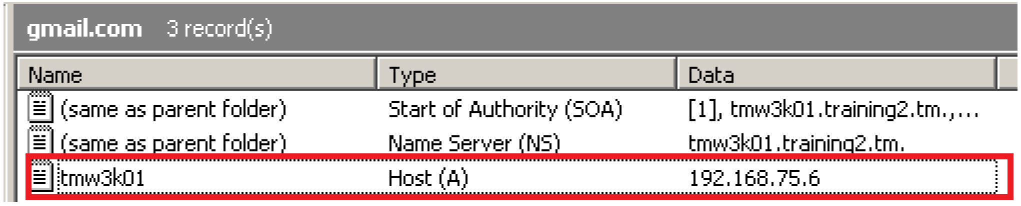
When we try to send a mail to our private gmail.com zone we see the DNS resolution responds now with the CNAME record just as expected by the RFC definition.



```
DNS    Standard query MX gmail.com
DNS    Standard query response CNAME tmw3k01.gmail.com
```

Image 11. CNAME resolution when no MX record is associated with the domain

The last condition is the absence of both the MX and CNAME records, in such a case the last try is to obtain an A record. Our test zone has now only this record to confirm this behavior.

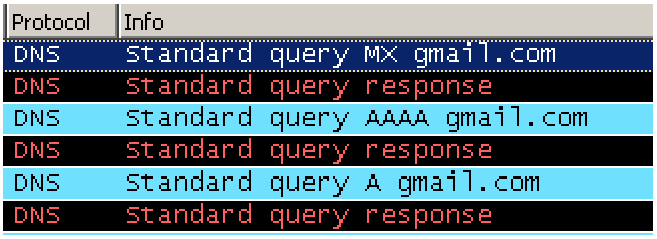


gmail.com 3 record(s)

Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[1], tmw3k01.training2.tm.,...
(same as parent folder)	Name Server (NS)	tmw3k01.training2.tm.
tmw3k01	Host (A)	192.168.75.6

Image 12. Domain definition with only an A record

When the resolution is performed, the result is as expected. In this case our MTA will try to deliver mail to the A record. You must note here, that in real life there are some domains that no longer host an MX record but they do have an A record for their website, in these cases mail will remain queued trying to deliver mail to this server.



Protocol	Info
DNS	Standard query MX gmail.com
DNS	Standard query response
DNS	Standard query AAAA gmail.com
DNS	Standard query response
DNS	Standard query A gmail.com
DNS	Standard query response

Image 13. A record resolution when no MX record is associated with the domain

At this point, the MTA should be ready to open and receive connections on port 25 and accept mails from other SMTP Clients. The next step will involve the handshake between both MTA servers at an SMTP level.

This Handshake is composed from both servers' hostnames (FQDN). The SMTP Server is the first to start the conversation by presenting the following line:

220 [SMTP_Server_FQDN] [SMTP_Version_And_Other_Info]

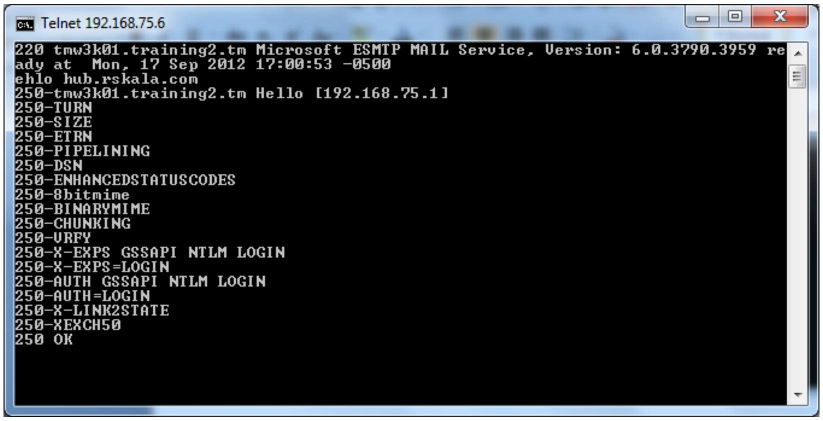
[220] This is the SMTP code that tells the SMTP Client that it may now start sending commands. The description that follows the numeric code is just for human reading and is never interpreted by the machines. It is worth mentioning that even the SMTP RFC states that at this point the SMTP Server should identify itself with both its FQDN and SMTP version, it is now allowed to skip this information if it involves a security issue on your organization.

The SMTP Client should always start the communication by introducing itself with one of the following commands:

HELO: This command is only used when the SMTP Client has only SMTP capabilities and is defined by the RFC 821 document. This is now deprecated and should not be used on actual conversations.

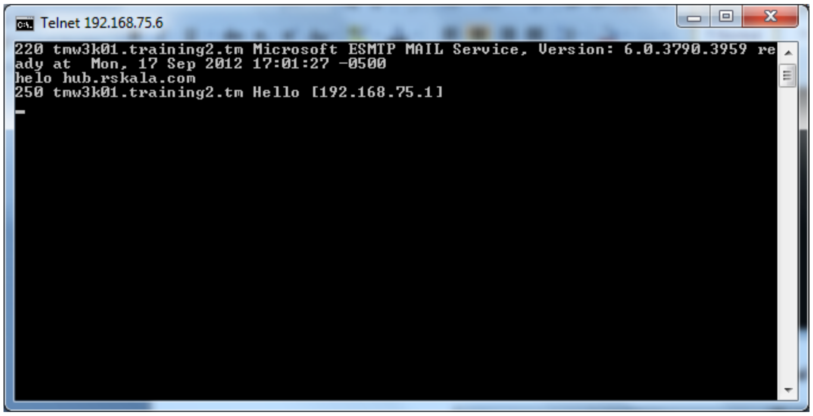
EHLO: This is the documented command on RFC 2821 and 5321 for ESMTP in order to take advantage of the new characteristics of the protocol and is the one that should be used in all SMTP communications.

The following screenshots show the difference between both commands.



```
220 tmw3k01.training2.tm Microsoft ESMTP MAIL Service, Version: 6.0.3790.3959 re
ady at  Mon, 17 Sep 2012 17:00:53 -0500
ehlo hub.rskala.com
250-tmw3k01.training2.tm Hello [192.168.75.1]
250-TURN
250-SIZE
250-ETRN
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-8bitmime
250-BINARYMIME
250-CHUNKING
250-VRFY
250-X-EXPS GSSAPI NTLM LOGIN
250-X-EXPS=LOGIN
250-AUTH GSSAPI NTLM LOGIN
250-AUTH=LOGIN
250-X-LINK2STATE
250-XEXCH50
250 OK
```

Image 14. HELO Response



```
220 tmw3k01.training2.tm Microsoft ESMTP MAIL Service, Version: 6.0.3790.3959 re
ady at  Mon, 17 Sep 2012 17:01:27 -0500
helo hub.rskala.com
250 tmw3k01.training2.tm Hello [192.168.75.1]
_
```

Image 15. EHLO Response

After accepting the greeting, the SMTP Server responds with a 250 status code in which it identifies itself with its own FQDN. The line usually contains “Hello [IP] where the IP is the one from the SMTP Client.

The main objective of these operations is to allow both SMTP Server and Client to have tracing info from the mail transaction and to perform troubleshooting when needed. The IP and name from the SMTP Client will be registered in the mail headers under the ***Received*** header which should be added at the beginning of any other existing ***Received*** header.

Received: from **hub.rskala.com** (unknown [**192.168.0.89**])
  by *hub2.rskala.com* (Postfix) with *ESMTP* id 9E08E900418
  for <user@rskala.com>; Mon, 27 Sep 2010 21:50:26 -0500 (CDT)

If you wish to verify the IP reputation of the SMTP Client you should implement such methods at this point.

2.2.2. Envelope

Now that the Handshake has been established and both SMTP Client and Server had agreed to start an SMTP session, the responsibility to send the transmission commands belongs to the SMTP Client while the SMTP Server will be responsible of answering to each of them with the corresponding status codes.

The next section to transmit is the Envelope, which is used by SMTP Server to have enough information about to whom the message should be delivered to and to know to whom a notification should be sent in case the transmission fails. The fields that define the header are called:

1. *Forward-Path:* This is the address or addresses intended for delivery. These are actually the recipients' mailboxes.
2. *Reverse-Path:* This is the email address that can be used to send a notification in case the mail transmission fails. This is actually the sender's mailbox.

RFC 821 defined these fields for the first time. The associated commands had the capacity to define the complete route the mail should follow to reach its final destination. These commands are:

- **MAIL:** Establishes the return path in case the mail could not be delivered to the intended recipients. It accepts the FROM word as its only parameter. The complete syntax is defined in RFC 821 as: MAIL FROM:<@relay,@relay2:user@domain>. Where the first parameters are the MTA servers the mail should pass through to reach its destination. This syntax is now obsolete and even prohibited by RFC 2821 and 5321 because it opens a security hole given the fact that all MTA servers should be configured as open relays, this means, accepting mail on behalf of another server. Even though this syntax is still accepted by an MTA it is not implemented when the mail is sent. The right syntax is just MAIL FROM:<user@domain>.

- **RCPT:** Establishes the path that should be used to deliver the mail. Here you should put all the recipient addresses. It accepts the word "TO" as its only parameter. Its syntax was defined by RFC 821 as: RCPT TO:<@relay1,@relay2:user@domain>. Where the first two parameters, as in the MAIL command, are now obsolete for the same reason. RFC 2821 defines the official syntax as RCPT TO:<user@domain>, and this line should be repeated for every recipient. It is not allowed to put more than one recipient on the same line. The domains that appear on these lines are the ones used by the SMTP Client to resolve the MX records of the corresponding domains.

After the HELO/EHLO has been accepted, the SMTP Server waits for the MAIL command to start opening a buffer for the new envelope. An envelope can only contain one MAIL command, if this is repeated again, the SMTP Server should answer with an error code, telling the SMTP Client that it is nesting mails.

```
MAIL FROM:<user@rskala.com>
250 2.1.0 user@rskala.com....Sender OK
MAIL FROM:<user2@rskala.com>
503 5.5.2 Sender already specified
```

If you are to validate or create rules about the sender information, this is the right point to implement them. If there was any mistake when sending the mail commands you can always clear the buffer by sending the EHLO/HELO command again or by sending the RSET command.

After the MAIL command you should send the RCPT command. This can be repeated as many times as needed by the number of recipients, always sending one mailbox on each line. At this point a new Internal ID is generated specific for this mail. The envelope will continue to store the RCPT fields until you are finished with your recipient list. This happens when you send the DATA command. When a SMTP Client sends the DATA command, the envelope is closed and a new buffer is opened to received the mail content identified by the Internal ID. If you are to validate or create rules for your recipients, this is the point where you should implement them.

The information of the MAIL command is transferred by the MTA to the headers using the header name:

Return-Path

The information of the RCPT command is not stored on any header so this info will be lost once the mail has been processed. To make a Reply / Forward of the mail, the information of the To and CC headers is used to fill the corresponding mailbox addresses.

2.2.2.1. SIZE

As part of the SMTP extension, it is now allowed to use a new parameter that helps to identify if the mail meets the size requirements accepted by your SMTP Server. This parameter is SIZE and its syntax is described below:

```
MAIL FROM:<user@domain> SIZE=size
Where SIZE is the approximate mail size in bytes
```

This parameter helps MTA servers to make previous validation of the mail size. Its implementation is described in RFC 1870 that establishes the following conditions:

```
6.2  Client action on receiving response to extended MAIL command
The client, upon receiving the server's response to the extended
MAIL command, acts as follows:

   (1) If the code "452 insufficient system storage" is returned, the
       client should next send either a RSET command (if it wishes to
       attempt to send other messages) or a QUIT command. The client
       should then repeat the attempt to send the message to the server
       at a later time.

   (2) If the code "552 message exceeds fixed maximum message size" is
       received, the client should immediately send either a RSET command
       (if it wishes to attempt to send additional messages), or a QUIT
       command.  The client should then declare the message undeliverable
       and return appropriate notification to the sender (if a sender
       address was present in the MAIL command).
```

```
A successful (250) reply code in response to the extended MAIL
command does not constitute an absolute guarantee that the
message transfer will succeed.  SMTP clients using the extended
MAIL command must still be prepared to handle both temporary and
permanent error reply codes (including codes 452 and 552),
either immediately after issuing the DATA command, or after
transfer of the message.
```

When this method is implemented by the SMTP Server, the EHLO greeting should specify the acceptable mail size. This helps the SMTP Client to decide whether to continue or abort the transmission.

```
220 tmcent01.training5.tm ESMTP Postfix
ehlo me.com
250-tmcent01.training5.tm
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

Because this parameter is optional, the MTA servers may choose to not implement it by leaving the SIZE parameter blank or not to present it at all in the greeting. This size is only an approach as the final size should also consider the size of the commands like ".", <CRLF> and any other character involved in the mail transmission. Any MTA server that accepts a mail because the SIZE of the final mail was lower than its threshold can still reject the mail because of other reasons like hard disk space.

To confirm this process you can connect to any mail server and send the EHLO command, if it has a limit you'll see it in the mechanisms list. You should note that you MUST use EHLO, otherwise you'll not be able to know what the SIZE limit is.

```
250-SIZE 10240000
```

This response indicates this server can only accept mails with a size of 10MB or less. Any bigger mail will get rejected as shown in the following example:

```
MAIL FROM:<user@domain.com> SIZE=888888888888888
552 5.3.4 Message size exceeds file system imposed limit
```

If the mail has an acceptable size then the SMTP conversation can continue.

```
MAIL FROM:<user@dominio.com> size=888
250 2.1.0 Ok
```

If there’s no specific number for the SIZE parameter as shown in the following line, it means that server hasn’t impose any limit on the mail sizes it can accept.

```
250-SIZE
```

In such case, even the following MAIL will be accepted:

```
MAIL FROM:<user@domain.com> SIZE=9999999999999999999
250 2.1.0 user@domain.com....Sender OK
```

A normal MTA server will react according to this definition, however, any attacker can take advantage of this behavior and use a script or manually send a mail to bypass this kind of restrictions by sending a SIZE parameter with a smaller size. It is advisable to configure this parameter on all MTA servers for the SMTP Clients to validate if they can send specific mails or abort the operation on time without wasting resources on both sides. For those cases when the mail comes from an attacker, it is advisable to configure a content filter rule that calculates the actual size of the mail and apply the corresponding actions before delivering the mail. The disadvantage of this kind of filters is that you need to receive the whole mail but it does really worth it if you want to control mail sizes.

2.2.2.2. DSN-RCPT-NOTIFY

RFC 3461 gives the possibility to extend SMTP protocol by adding the NOTIFY method in order to establish the conditions for a DSN (Delivery Status Notification) to be sent. This makes the protocol more flexible because it can now control when and what to notify the sender when a certain condition is met.

It was a definition from RFC 821 that when a mail cannot be delivered to final recipient, a notification should be sent to the original sender to let him know an error had occurred. However this kind of notifications didn’t specify the reason that caused the DSN to be sent. With the NOTIFY method we can request a notification for the following conditions:

- RCPT TO:<user@domain> *NOTIFY=NEVER*. Establishes that under no circumstance a notification should be returned to the sender, not even when the mail delivery was unsuccessful.

- RCPT TO:<user@domain> *NOTIFY=SUCCESS*. Establishes that a notification must be sent to the sender when the delivery is successful.
- RCPT TO:<user@domain> *NOTIFY=FAILURE.* Establishes that a notification must be sent to the sender when the delivery has failed.
- RCPT TO:<user@domain> *NOTIFY=DELAY.* Establishes that a notification must sent to the sender when the delivery of the mail is delayed. It is recommended not to use this parameter because there might be several reasons for a delay to occur and the sender might misunderstand this kind of notifications as an actual error.

The NEVER parameter, if used, must appear as the only parameter of the NOTIFY method. The rest might appear in a list format separated by comma. This is why this method gives complete control about the state of the mail on any condition. You just have to be careful in not overloading your network with unnecessary mails.

It's worth noting that this parameters can be defined for one or all the recipients, and you can even alternate different types of notifications for each recipient on the same mail.

NOTE: You have to understand the delivery of the mail as the process when the mail is successfully delivered to the recipient mailbox in the domain that corresponds, or when the mail has been delivered directly to the User Agent (MUA of the recipient).

To confirm how this works, let's send a mail using the command line with the required parameters as shown in the following example.

```
MAIL FROM:<user@domain.com>
250 2.1.0 Sender OK
RCPT TO:<user@rskala.com> NOTIFY=SUCCESS
250 2.1.5 Recipient OK
DATA
354 Start mail input; end with <CRLF>.<CRLF>
.
250 2.6.0 <8891b431-21ec-4a6c-96a5-6c0203e21f0e> Queued mail for delivery
```

As soon as the mail is accepted by the MTA server you obtain an answer in your mailbox indicating the SUCCESS operation.

Your message

To: Undisclosed recipients
Subject:
Sent: 9/17/2013 8:34 PM

was delivered to the following recipient(s):

user@rskala.com on 9/17/2013 8:34 PM

The following example shows how to use the FAILURE notification option:

```
MAIL FROM:<user@domain.com>
250 2.1.0 Sender OK
RCPT TO:<asdfasfd@rskala.com> NOTIFY=FAILURE
250 2.1.5 Recipient OK
data
354 Start mail input; end with <CRLF>.<CRLF>
.
250 2.6.0 <ef6cf1a7-79a0-487e-96da-1d8f78e067b2> Queued mail for delivery
```

Because this mailbox doesn't really exist, a failure notification will be sent as shown below:

Your message did not reach some or all of the intended recipients.

Subject:
Sent: 9/17/2013 8:40 PM

The following recipient(s) could not be reached:

asdfasfd@rskala.com on 9/17/2012 8:40 PM
The e-mail account does not exist at the organization this message was sent to. Check the e-mail address, or contact the recipient directly to find out the correct address.
<rskala.com #5.1.1 smtp;550 5.1.1 RESOLVER.ADR.RecipNotFound; not found>

When using the NEVER parameter you're indicating that no matter the delivery result (success or failure) a notification must not be sent. DELAY only means you want to receive a notification whenever a situation exists on any of the MTA servers that will not allow the mail to be received in the first delivery try.

2.2.2.3. DSN-RCPT-ORCPT

This method, specified by RFC 3461, allows the DSN to have the original mailbox of the recipient (OriginalRecCiPienT) in those circumstances where the mailbox address in the RCPT TO command should be modified for any reason. Some reasons where this field can be used is when you receive mails from systems that use different standards like X.400 where the codification is not the same as defined by RFC 5321. Other situation is when there is a distribution list in the RCPT TO command and this must be modified to generate a RCPT TO that matches every mailbox address that compose the list, in this case the ORCPT field should match the new mailbox address as well.

The ORCPT field SHOULD ALWAYS match the mailbox specified by the RCPT command and in such a case that it has been received by an SMTP Relay, it should transfer the same field as received from the previous SMTP Client. In case this field doesn’t exist, any MTA server is allowed to modify the mail transfer to incorporate it by copying the RCPT address and code it with the RFC 822 standard (xtext).

The right syntax is:

RCPT TO:<user@domain> *ORCPT=rfc822;user@domain*

When a DSN is generated and if the ORCPT field exists, this mailbox address should be used to let the sender know what notification it is referred to.

2.2.2.4. DSN-MAIL-RET

This method is used with the MAIL command and it is used to specify how the original mail should return to the sender in case of failure. The MTA servers that implement this method and those that were RFC 821 compliant would only sent the headers of the original mail, but this method allows the complete message to be sent. Its implementation is done by using the following syntax:

- MAIL FROM:<user@domain> RET=FULL. With this parameter we indicate the SMTP Server that in case a DSN is to be sent, this must include the complete mail in the notification.
- MAIL FROM:<user@domain> RET=HDRS. With this parameter we indicate the SMTP Server that in case a DSN is to be sent, this must include only the headers of the original mail.

If the RET parameter doesn’t exists, the MTA server is allowed to return only the headers of the original mail, although this can be configured in the MTA.

2.2.2.5. DSN-MAIL-ENVID

The ENVID (Envelope ID) method is used to add a Message-ID to the DSN notification. This will help in tracing the mail that originated the DSN. In those cases where you don’t want to receive the headers or the full mail, this information will be helpful because the DSN will return the original Message-ID of the original mail. This helps to trace a mail in an easy and fast way.

If it is not implemented, no information about the ID will be returned by the MTA that originates the DSN.

The syntax for this mechanism is:

```
MAIL FROM:<user@domain> ENVID=<Envelope_ID>
Where Envelope_ID is any alphanumeric ID
```

When used, you'll receive a line called "Original-Envelope-Id" that contains the Envelope-ID value in the ENVID method as shown below:

```
Original-Envelope-Id: 12345
Reporting-MTA: dns;hub.rskala.com
Received-From-MTA: dns;rskala.com
Arrival-Date: Tue, 18 Sep 2013 14:59:38 -0500

Final-Recipient: rfc822;user@dominio.com
Action: failed
Status: 5.1.1
```

This information can be used to trace a mail when the delivery operation has failed.

2.2.2.6. MDN

RFC 3798 establishes the mechanisms for the sender to receive a notification when the mail has been read. Because this notification requires the user interaction, the method is NOT IMPLEMENTED IN THE MAIL ENVELOPE. This method is implemented directly in the mail headers because it requires a header that keeps this information when the mail transfer has finished.

The right syntax is:

```
DATA
354 End data with <CR><LF>.<CR><LF>
subject: hello
Disposition-Notification-To:<root@training5.tm>

this is the mail content.
.
250 2.0.0 Ok: queued as 70ACF90040A
```

This method requires the mailbox address specified in the Disposition-Notification-To: header be the same of the MAIL command, if they are different the MTA may decline to send the notification because an attacker may use this method to bombard not involved users.

The mail rules to follow when using this method are described below:

2.1. The Disposition-Notification-To Header

The presence of a Disposition-Notification-To header in a message is merely a request for an MDN. The recipients’ user agents are always free to silently ignore such a request. Alternatively, an explicit denial of the request for information about the disposition of the message may be sent using the "denied" disposition in an MDN.

An MDN MUST NOT itself have a Disposition-Notification-To header. An MDN MUST NOT be generated in response to an MDN.

A user agent MUST NOT issue more than one MDN on behalf of each particular recipient. That is, once an MDN has been issued on behalf of a recipient, no further MDNs may be issued on behalf of that recipient, even if another disposition is performed on the message. However, if a message is forwarded, an MDN may have been issued for the recipient doing the forwarding and the recipient of the forwarded message may also cause an MDN to be generated.

While Internet standards normally do not specify the behavior of user interfaces, it is strongly recommended that the user agent obtain the user's consent before sending an MDN. This consent could be obtained for each message through some sort of prompt or dialog box, or globally through the user's setting of a preference. The user might also indicate globally that MDNs are to never be sent or that a "denied" MDN is always sent in response to a request for an MDN.

MDNs SHOULD NOT be sent automatically if the address in the Disposition-Notification-To header differs from the address in the Return-Path header (see [RFC-MSGFMT]). In this case, confirmation from the user SHOULD be obtained, if possible. If obtaining consent is not possible (e.g., because the user is not online at the time), then an MDN SHOULD NOT be sent.

Confirmation from the user SHOULD be obtained (or no MDN sent) if there is no Return-Path header in the message, or if there is more than one distinct address in the Disposition-Notification-To header.

The comparison of the addresses should be done using only the addr-spec (local-part "@" domain) portion, excluding any phrase and route. The comparison MUST be case-sensitive for the local-part and case-insensitive for the domain part.

If the message contains more than one Return-Path header, the implementation may pick one to use for the comparison, or treat the

```
   situation as a failure of the comparison.

   The reason for not automatically sending an MDN if the comparison
   fails or more than one address is specified is to reduce the
   possibility of mail loops and of MDNs being used for mail bombing.

   A message that contains a Disposition-Notification-To header SHOULD
   also contain a Message-ID header as specified in [RFC-MSGFMT].  This
   will permit automatic correlation of MDNs with their original
   messages by user agents.

   If the request for message disposition notifications for some
   recipients and not others is desired, two copies of the message
   should be sent, one with a Disposition-Notification-To header and one
   without.  Many of the other headers of the message (e.g., To, Cc)
   will be the same in both copies.  The recipients in the respective
   message envelopes determine for whom message disposition
   notifications are requested and for whom they are not.  If desired,
   the Message-ID header may be the same in both copies of the message.
   Note that there are other situations (e.g., Bcc) in which it is
   necessary to send multiple copies of a message with slightly
   different headers.  The combination of such situations and the need
   to request MDNs for a subset of all recipients may result in more
   than two copies of a message being sent, some with a Disposition-
   Notification-To header and some without.

   Messages posted to newsgroups SHOULD NOT have a Disposition-
   Notification-To header.
```

An additional way to achieve an MDN is by using the following header:

Return-Receipt-To:<user@domain>

This method is outside the SMTP standards and any MTA server is free to decline to send an MDN notification.

2.2.3. Headers

When you close the envelope with the DATA command the header section begins. This section is used to identify relevant information about this mail. This information can be used to add other data like the date, the mail ID, the MTA servers the mail has passed through, additional identifiers and others that may just be informative.

According to RFC 5322 about the mail format, there are only two headers that are mandatory: Date and From. If these do not exist, the MTA server is free to generate them and integrate them to the final mail. The headers of any mail can be divided in the following categories.

2.2.3.1. General rules for headers

In this section we'll see the general rules that apply to any header used in any mail that is RFC 5321 and 5322 compliant whether or not the headers are SMTP standard or extended.

1. According to RFC 5322 there are only two mandatory headers in any mail: **From** and **Date**. This will assure the mail has enough identification information in order to know who is originating the mail and the date and time in which it was generated. If these fields are not present, any MTA is free to add them using the information in the MAIL command for the From header and its local time for the Date header.

   To confirm this definition you can send a mail with no headers and then check what headers are automatically added by the MTA to be compliant with this rule. The following mail doesn't have any headers:

   ```
   MAIL FROM:<>
   RCPT TO:<user@rskala.com>
   DATA
   .
   ```

   As a result, we can see an Exchange server delivers this mail with the following headers:

   ```
   Microsoft Mail Internet Headers Version 2.0
   Received: from hub.rskala.com ([192.168.0.89]) by hub1.rskala.com
   with Microsoft SMTPSVC(6.0.3790.3959);
          Tue, 18 Sep 2013 19:07:10 -0500
   From: <>
   Bcc:
   Return-Path: <>
   Message-ID: <TMW3K01s43ZRYH6GXDJ00000001@tmw3k01.training2.tm>
   X-OriginalArrivalTime: 19 Sep 2013 00:07:12.0073 (UTC)
   FILETIME=[B7BF1B90:01CD95FA]
   Date: 18 Sep 2012 19:07:13 -0500
   ```

   If you perform the same test by sending the mail to Postfix, you can see the following headers:

```
X-Original-To: user1
Delivered-To: user1@rskala.com
Date: Tue, 18 Sep 2013 19:24:11 -0500 (CDT)
From: MAILER-DAEMON
To: undisclosed-recipients:;
```

With this test we verified that even when not added, an MTA will try to add any required header and of course, both From and Date will always be added no matter what MTA server you use.

2. You can use comments at any part of the headers using the syntax "(" comment ")". This information is informative only and should not be interpreted by the MTA server. If any MTA receives comments in this format it SHOULD NOT delete them or modify them.

   The following is an example of a comment:

   ```
   Received: from mail-pb0-f52.google.com ([209.85.160.52]) by In-MTA
   ```

3. The character-set used in all headers is always US-ASCII. RFC 2047 (about the non-ASCII characters in a MIME mail) allows the use of additional charsets. When there is no explicit definition about the charset in use, it should be assumed that the charset is US-ASCII. Any character that doesn't belong to the US-ASCII charset and that is not specified by a different charset can be ignored by the MTA server.

   This definition implies that any non US-ASCII character can be omitted or its presentation may be modified by the SMTP Server when performing the final delivery. To confirm how this mechanism works let's send the following mail and then let's check how it is being showed to the user:

   ```
   MAIL FROM:<user@domain.com>
   RCPT TO:<user@rskala.com>
   DATA
   From:user1
   To:user2
   Subject: nos vemos mañana

   saludos nos vemos mañana aquí.
   .
   ```

   When you open this mail in an Exchange server, you'll see something like this:

```
from:user1
to:user2
subject: nos vemos ma¤ana

saludos nos vemos ma¤ana aqu¡.
```

Postfix will show the following:

```
from:user1
to:user2
subject:nos vemos ma¤ana

saludos nos vemos ma¤ana aqu¡.
```

With these tests it should be clear by now that any non US-ASCII character will not get properly interpreted when the mail is presented to final recipient. In these cases some strange characters will be shown according to the actual ASCII value the character corresponds to.

4. If non US-ASCII characters are to be used, you have to follow the rules defined by RFC 2047 that indicates there are two different types of codification: Q (for Quoted-Printable) and B (for Base64). For example: the sentence “this is the text” may be coded as “=?iso-8859-1?Q?this is the text?=” which is the same as “=?iso-8859-1?B? dGhpcyBpcyB0aGUgdGV4dA==?=”.

   Some users may think the mail may be malicious when they see this kind of coding. As we can see here, this is not the case as this method is perfectly legal and its rules are defined in the RFC. The purpose of the method is to cover the problem implied in the previous point that forbids the use of non US-ASCII characters when your language is not included in this charset. If you were to send the last mail you would have to send something like this.

```
MAIL FROM:<user@domain.com>
RCPT TO:<user@rskala.com>
DATA
From:user1
To:user2
Subject: =?iso-8859-1?Q?nos_vemos_ma=F1ana?=
MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

saludos nos vemos ma=F1ana aqu=ED.
.
```

You'll note here that when you code text inside the body you have to use the rules defined by the MIME format, this will be covered in the following sections. For the header you can use de "Q" coding as in this example or the "B" (Base64) coding as shown in the following example:

```
MAIL FROM:<user@domain.com>
RCPT TO:<user@rskala.com>
DATA
From:user1
To:user2
Subject: =?iso-8859-1?B?bm9zIHZlbW9zIG1h8WFuYQ==?=
MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

saludos nos vemos ma=F1ana aqu=ED.
.
```

In both examples the "ñ" and "í" characters will be correctly presented to the final recipient.

5. The order in which headers are sent does not affect its implementation. The only exception is the MIME-Version header which should always appear before any MIME header.
6. As a general rule no header is case sensitive unless the implementation of a specific headers specifies it.

2.2.3.2. Date and time Headers

These headers are used at the moment the mail is created.

Header	Syntax	Description

Date	**Date: date-time**
	Syntax: [day-of-week ","] date time [CFWS]
	day-of-week = "Mon" / "Tue" / "Wed" / "Thu" / "Fri" / "Sat" / "Sun"
	date = [dd] month [yyyy]
	month = "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" / "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"
	time = time-of-day zone
	time-of-day = hour ":" minute [":" second]
	zone = ("+" / "-") 4DIGIT
		This header indicates the date and time in which the mail was created by the originating MTA. This is the equivalent to the moment when the user presses the Send button, this is why this field should not be confused with the time in which the SMTP Server receives the mail. If for example, a user

Header	Syntax	Description
	Example 1: ***Date: 12 May 2010 00:00:00 -0500*** where: *12* - Day *May* – Month with three letters *2010* – Year with four digits *00:00:00* - Hour:Minutes:seconds in 24hrs format *-0500* - Time zone Example 2: ***Date: Mon, 12 May 2010 -0500*** where: *Mon* – Name of the day with three letters separated by comma	creates a new mail but he hasn't an Internet connection at the moment, the actual time in the mail will reflect the one when the user pressed the Send button and not the time when it got an Internet connection to send the mail.

2.2.3.3. Origin Headers

These headers have information about the system / user that originated the mail.

Header	Syntax	Description
from	**From: mailbox** Examples: *From:<user@domain.com>* *From: "John Doe"<johndoe@domain.com>*	This field contains the original sender mailbox address. This should be the mailbox address of the mail author.
sender	**Sender: mailbox** Example: *From:<john@domain.com>* *Sender:<chris@domain.com>* This mail will be send as: *From Chris on behalf of John*	When a user sends a mail whose author is a different person, this field will distinguish both of them in order for the recipient to know who the real author is in case he wants to contact him. For example. If John writes a mail and asks his assistant Chris to send it. John will be shown in the FROM header and Chris in the SENDER field, when the recipient replies, the response will be addressed to John, for he is the real author of the mail. If the

Header	Syntax	Description
		value of SENDER and FROM is the same, then it is not necessary for the SENDER header to appear in the mail.
reply-to	**Reply-To: address-list** Example: *From:<john@domain.com>* *Reply-To:<john@domain.com>, <chris@domain.com>* When replying to this mail, the mailbox addresses that will appear on the To header will be john@domain.com and chris@domain.com	This header will provide the mailbox or mailboxes to which the mail should be replied to in those situations where it is necessary to inform the recipient about the person or list of people to whose he should reply the mail to and those people do not appear in the FROM header.

2.2.3.4. Destination Headers

These headers give information about the mail recipients.

Header	Syntax	Description
to	**To: address-list** Examples: *To:<john@domain.com>* *To:<john@domain.com>, <chris@domain.com>* *To:"John Doe"<john@d.co>, "Chris"<chris@d.co>*	This header indicates the mailboxes to which the mail is addressed to.
cc	**Cc: address-list** Example: *To:"John Doe"<john@domain.com>* *Cc:"Chris Doe"<chris@domain.com>, <al@d.co>* In this example John Doe will receive the mail as the main recipient and Chris will receive a copy even though the mail is not directly addressed to him.	Carbon Copy is a header that is used when you want to send a copy of the same mail to other people that are not directly related or responsible of the mail topic.
bcc	**Bcc: address-list**	Black Carbon Copy is a header that

Header	Syntax	Description
	Example: *To:"John Doe"<john@domain.com>* *Cc:"Chris Doe"<chris@domain.com>* *Bcc:"Al Doe"<al@domain.com>* In this example the Bcc line indicates that Al mailbox should not be visible to John and Chris.	allows you to send copies of a mail to one or several persons without their mailboxes being visible to the other recipients in the To and Cc headers. This line should be eliminated by the MTA servers. How MTA servers deal with this header depends on the MTA platform.

2.2.3.5. Identification Headers

These headers contain identification information about the original mail and / or its relationship with previous mails.

Header	Syntax	Description
message-id	**Message-Id: msg-id** Examples: *Message-ID:<1C6DA6E38824413B33C67@mta.domain.com>* *Message-ID:<ABC1234EFDAB234>* *Message-ID:<123423423@user.domain.com>*	Each MTA server should generate its own process that ensures that every sent mail contains a unique identifier that identifies among all the mails sent by itself. There is no standard for the ID but these usually use the server hostname, the domain or the user's mailbox that originates the mail.
references	**References: msg-id** Example: *Message-ID: <BCDAFAABCD12AED>* *References: <ABC1234EFDAB234>, <123423423@user.domain.com>* In this example, the Message-ID refers to the	When you reply several times to the same mail it is common to use REFERENCES to include the Message-IDs of the mails involved in the

Header	Syntax	Description
	identifier of the mail that is being sent. References contains the Message-ID of the mails on which a reply has been made.	conversation in order to speed up the indexing and tracing processes of all the mails related with the same topic.
in-reply-to	**In-Reply-To: msg-id** Example: *Message-ID: <BCDAFAABCD12AED>* *In-Reply-To: <ABC1234EFDAB234>* Here, the Message-ID refers to the unique identifier of the mail that is being sent and the In-Reply-To contains the Message-ID of the mail on which a reply has been made.	This header is used to send the original Message-ID of a mail on which a user has replied for the first time. (REFERENCES is currently used instead of this header).

2.2.3.6. Information Headers

These headers contain information that will be interpreted by the recipient.

Header	Syntax	Description
subject	**Subject: text** Example: *Subject: This is the subject of the mail*	This header is used for the recipient to quickly identify what the topic of the mail is. It is usually a short description of the content and it should not overpass the 998 character limit although it is recommended to be of no more of 78 characters.

2.2.3.7. Tracing Headers

These headers contain information used to identify the different points the mail has passed through.

Header	Syntax	Description
return	**Return-Path: path** Example: *Return-Path:<user@domain.com>*	The value from the MAIL command in the mail envelope (MAIL FROM) remains in the final mail in the Return-Path header. There can only be one header of this type on every

Header	Syntax	Description
		mail.
received	**Received: received-token** Example: *Received: from tmcent01.training6.tm ([192.168.75.129]) by tmw3k01.training2.tm with Microsoft SMTPSVC(6.0.3790.3959); Mon, 4 Oct 2010 23:48:27 -0500* *Received: from unknown (unknown [192.168.75.1]) by tmcent01.training6.tm (Postfix) with SMTP id 6755D90041A for <user@domain.com>; Mon, 4 Oct 2010 23:48:09 -0500 (CDT)* This mail originated in the mail server with IP 192.168.75.1 and received by tmcent01.training6.tm. Later, the mail was received from server tmcent01.training6.tm by server tmw3k01.training2.tm. The Received headers must be read from the bottom to the top, being the last Received the first MTA in the mail flow and first being the final MTA server to which the mail was delivered to.	The Received headers are written to have the information of all the MTA servers the mail has passed through. Even when RFC 5322 does not force these headers to exist, RFC 5321 actually forbids these headers to be removed or modified by any MTA once they've been received. Every MTA that receives a mail must stamp a new Received header with its own information about from which server it received the mail and its own hostname and IP.

2.2.3.8. Optional Headers

These headers may or may not appear on the final mail.

Header	Source	Description
X-Headers	SMTP - RFC 5322	These headers may contain any kind of additional information to be used or interpreted by a user or software. The only rule that must be met is for these headers to begin with "X-" or "x-".
Disposition-Notification-To:<user@domain.com>	SMTP - RFC 3798	This is used to send a read receipt notification. The value for <user@domain.com> corresponds to the mailbox to which the notification will be

Header	Source	Description
		sent.
Content-Language: es-MX	SMTP - RFC 3282	This is used to indicate the preferred language for reading the mail. The first two letters indicate the language (es) and the other two correspond to the country (MX), this is because there might be a difference in the same language depending on the country for example es-MX and es-ES (Spain).
Importance: Normal	SMTP - RFC 2156	This is a description about the importance of the mail. The valid values are Low, Normal and High.

2.2.3.9. MIME Headers

These headers identify the mail as MIME compliant.

Header	Syntax	Description
MIME-Version	Syntax: MIME-Version 1.0	This header is defined once for each mail and is used to identify this mail as MIME compliant. The only actual version is 1.0.
Content-Type	Syntax: Content-Type: media type/subtype; parameters *media type= "text" / "image" / "video" / "audio" / "application"/ "multipart" / "message* *subtype = any specific media type* *parameters =any modifier that helps to correctly interpret the information format*	This is used to define the type and format of the information transmitted. Media type is a generic description of the content. Subtype is a more specific format definition. Parameters can be used in conjunction with media types like "text" to define more information about the content. The seven main media types are: text, image, video, audio, application, multipart and message.
Content-Transfer-Encoding	Syntax: Content-Transfer-Encoding: mechanism *mechanism= "7bit" / "8bit" / "binary" / "base64"*	Defines the coding mechanism used in the mail body. The default value is 7bit which corresponds to the US-ASCII charset. 8bit allows transmissions of the full ASCII charset. Binary is used to send data that is not covered by any specific format. Base64 encodes the content in a special charset allowing the transmissions of any character not contained in the US-ASCII.

Header	Syntax	Description
Content-ID	Syntax: Content-ID: msg-id	Used to identify a specific body identity throughout the whole mail. A clear example of its implementation is the use of embedded images inside the body.
Content-Description	Syntax: Content-Description: text	It is a human readable description about an specific body section.

Because of its complexity, the MIME headers will be shown in more detail in the following sections with practical examples.

2.2.4. Body

After closing the header section with the <CRLF><CRLF> syntax, all the remaining text will be treated as visible text and a user will be able to visualize it on the final mail. The only exception is when the body is defined in MIME format in which case the visible parts will be defined by the MIME headers. All of the information included in this section (including attachments) must be ASCII characters, just as requested by RFC 2821.

In the first version of SMTP (RFC 821) it was only allowed to send text that was contained under the US-ASCII charset, any other information was discarded, but after the revision of RFC 2821 it is now possible to send encoded characters for different languages and even file formats, the only condition is to meet the MIME format requirements.

In all of the SMTP Protocol revisions the fact that a system must work as long the syntax and structure of the mail is correct has been always implicit. This means that at any moment the protocol is not responsible nor is it force to perform any kind of verification of the information being transferred even if it is unsolicited data or malware. However, there are certain recommendations of having mechanisms that maintain the system secure against attacks.

The body structure in a mal is specified from RFC 2045 to 2049 about the MIME format which extends the earlier conditions of RFC 822 about IMF (Internet Mail Format) in order to provide the mail servers with the right mechanisms to send and interpret different types of data.

The MIME headers may appear as a result of the following two conditions:

1. As part of the header inside an ordinary RFC 822 compliant mail. This means that the headers may appear in normal mail with MIME format or as part of a RFC 822 mail attached to a MIME format mail.
2. Inside the MIME headers of a Multipart mail. This means, these headers will appear on each body section that define different content format. For example, there will be a MIME header to define a normal text section, another to define a music file format, another for an image file format and so on.

The following headers have been defined o specify that a body is MIME compliant:

MIME-Version

This header indicates the mail is MIME compliant, in which case it is only needed once. If this header is absent then the MTA server may treat this mail as MIME compliant. Right now the only version is 1.0.

Syntax: MIME-Version: 1.0

Content-Type

This is used to describe the body content in such a way the UA (like Outlook for example) can choose the right mechanism or application to present the user when he/she opens the mail. For example, if the mail contains an MP3 file, the UA will ask the OS if it already has an application that can handle such format, in which case the appropriate icon will appear to the user.

The value for this header is called “media-type”. The content and format of the data is described by specifying a type and subtype of the information, where type is a generic description of the data and subtype specifies the format. A media type “image/xyz” for example is enough for the UA (User Agent) to know the data is about an image even when the UA doesn’t recognize the “xyz” format.

RFC 2046 defines a set of 7 media types. Five of them are global data formats and the remaining two refer to mail structure that require an additional processing. The description for each format is shown below:

The main Media Types are:

1. **Text.** This corresponds to any encoded, rich or plain text format. If no value is defined, the default will be “text/plain”, where subtype “plain” defines that all characters included in the section are to be presented with its equivalent to ANSI X3.4-1986 (US-ASCII) representation. This is the simplest way to send text in any mail. This media type accepts a modifier called “charset”, used to define the charset that will be used to interpret the data. When this is not defined, the default value will be “Content-Type: text/plain; charset=us-ascii”. In such a case where neither subtype and charset are not recognized, the data should be treated as “application/octet-stream”.
2. **Image.** This is any data that requires a graphical device to present the information (monitor, printer, fax, etc.). The default value is “jpeg”. Any non recognizable subtype should be treated as “application/octet-stream”.
3. **Audio.** This is any data that requires an audio device to “present” the information. The default subtype is “basic” and is defined as a single coded audio channel with 8-bit ISDN mu-law [PCM] with a sample rate of 8000 Hz. Any non recognizable subtype should be treated as “application/octet-stream”.
4. **Video.** This is any data that requires a device capable of presenting motion picture data, including both software or specific hardware. The default subtype is “mpeg”. Any non recognizable subtype should be treated as “application/octet-stream”.
5. Application. This is any other not interpretable binary information or data that should be processed by a specific non standard application. There are two default subtypes: Octet-Stream which indicates any arbitrary binary information and PostScript which indicates the presence of PostScript language written information.

The two composition media types are:

6. **Multipart.** This is any data that includes multiple independent data entities. There are four basic subtypes: “mixed” to specify a mixed set of several parts, “alternative” to represent the same information in different formats, “parallel” for data sections that must be shown simultaneously, and “digest” for multipart entities in which each part has a default type of “message/rfc822”.
7. Message. Defines a whole or partial “encapsulated” mail included in the original mail. There are three subtypes: “rfc822” used when the content itself is an RFC 822 mail, “partial” used to transmit segments of the body when these are too big, and “external-body” to specify the body is located in an external source.

Content-Transfer-Encoding

Because RFC 2821 only allows the use of 7bit US-ASCII charset in lines no longer than 1000 characters, alternate means of codification are needed to send large amounts of data in different formats. The MIME format extends this definition by allowing information transmission in the following formats: “7bit” to send all the characters of US-ASCII, “8bit” to send the complete ASCII charset, “binary” for binary format data, “quoted-printable” to represent characters as they’ve been received without any coding / decoding operation, and “base64” to send large amounts of data, like files.

1. Quoted-Printable. This codification is used to transmit characters that correspond any charset, being the default US-ASCII. This is useful when a specific language includes characters not included here like “ñ”. It is usually used for coding the subject and the readable body content.
2. Base64. This is used to encode non readable characters like binary information. It is usually used to transmit files of any format. The coding / decoding algorithm allows the conversion of any kind of data to a reduced set of only 65 ASCII characters which can be easily transmitted over any SMTP session. Because of this conversion, it is expected for the information to grow up to 33% of the original data. For example, an attached 10MB PPT file will grow up to a maximum of 13MB.

According to standards RFC 2045 through 2049, you can generate any kind of mail that includes any of the file formats you may need. In the following sections we’ll review how the MIME format is really implemented in several types of mails, from the ones that are just plain text to the more complicated that include several sections and attached files.

2.2.4.1. Body Simple

The simplest form of a mail is the plain text mail that only contains US-ASCII characters. Any different character will be represented with its ASCII equivalent. If such characters are used in the SMTP commands, the SMTP Server may respond with one of the following status codes:

```
501 5.5.4 Unrecognized parameter
501 5.5.4 Invalid Address
500 5.3.3 Unrecognized command
```

The definition of this charset is shown in the following table:

Dec	Hex	Symbol	Dec	Hex	Symbol	Dec	Hex	Symbol

32	20	Space ()
33	21	!
34	22	“
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~

Table 2. US-ASCII characters

The simplest mail transmission in plain text is shown in the following SMTP conversation:

Example 1. Simple Body

```
220 tmw3k01.training2.tm Microsoft ESMTP MAIL Service, Version: 6.0.3790.3959 ready at  Tue, 1 Feb 2012 20:31:18 -
0600
EHLO test.training2.tm
250-tmw3k01.training2.tm Hello [192.168.75.3]
250-TURN
250-SIZE 1048576
250-ETRN
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-8bitmime
250-BINARYMIME
```

```
250-CHUNKING
250-VRFY
250-X-EXPS GSSAPI NTLM LOGIN
250-X-EXPS=LOGIN
250-AUTH GSSAPI NTLM LOGIN
250-AUTH=LOGIN
250-X-LINK2STATE
250-XEXCH50
250 OK
MAIL FROM:<asdf@asdf.com>                         }
250 2.1.0 asdf@asdf.com....Sender OK              }  Envelope
RCPT TO:<administrator@training2.tm>              }
250 2.1.5 administrator@training2.tm              }
DATA
354 Start mail input; end with <CRLF>.<CRLF>
from: user1                                       }
to: user2                                         }  Headers
subject:test                                      }
date:mon, 5 nov 1980                              }

este es un correo de prueba                       }  Body
saludos                                           }
.
250 2.6.0  <20110202023051.037496003C@hub.training2.tm> Queued mail for delivery
QUIT                                              }  SMTP Session closure
221 2.0.0 tmw3k01.training2.tm Service closing transmission channel
```

This is the easiest way to transmit a mail that only contains plain text. The main sections are described below:

Envelope Section.

1. **EHLO test.training2.tm.** This is the initial greeting used to start any SMTP conversation between both MTA servers using the SMTP extended version (ESMTP).
2. **MAIL FROM:<asdf@asdf.com>.** This command is used to determine a mailbox to use in case the mail delivery fails. This is the only envelope value that may be saved in the mail headers in the form of Return-path.
3. **RCPT TO:<administrator@training2.tm>.** This command sends the recipient mailbox. For mails with more than one recipient, a separate command must be send for each recipient.
4. **DATA.** This command closes the envelope buffer and starts a new buffer to temporarily store the mail body. The first section contains the Headers. All data contained here will be transmitted within the mail up to its final destination. Once the Header section is finished, the body section starts.

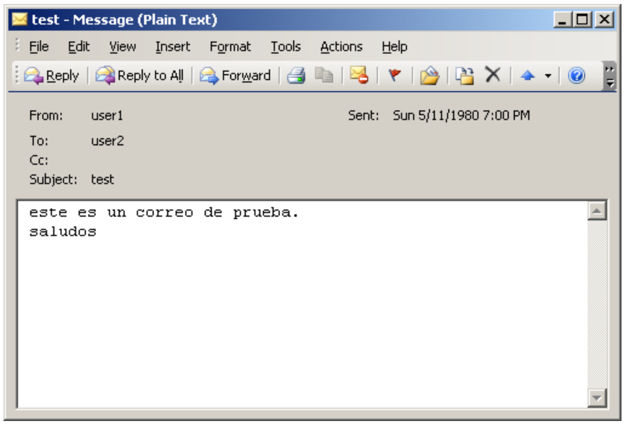
Headers Section.

```
from: user1
to: user2
subject:test
date:mon, 12 may 1980
```

This particular example doesn’t show any information about the MIME format. In these cases, the SMTP Server should assume the following implicit headers but it doesn’t have to add them to the original mail.

MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit

Whether the MIME headers are present or not, the mail will look something like this to the final recipient.



From: user1
Sent: Sun 5/11/1980 7:00 PM
To: user2
Cc:
Subject: test

```
este es un correo de prueba.
saludos
```

Image 16. Simple Body Mail

At this point you should note that information from the envelope and headers may be different, just compare the mailboxes used for the envelope section against the ones in the from and to headers. You should also note something strange in the date header. We’ll talk about these details in the Vulnerabilities section.

Body

este es un correo de prueba
saludos
.

This is the actual visible content of the mail body as can be seen in the previous image. Because the default MIME headers were assumed, given the fact we didn’t send them in the header section, all characters are represented in its US-ASCII equivalent. The last line, which contains only a “.” is not part of the content, this is the syntax used to indicate the SMTP Server the body section should be closed now.

SMTP Session closure.

```
QUIT
```

This command closes the SMTP session. The SMTP Client is allowed to send as many mails as long as the session remains open. In those cases where the session is still opened and the SMTP Client is not sending any more data, the SMTP Server is allowed to close the session when a certain timeout has passed.

2.2.4.2. Alternate Body

Actual User Agents like Outlook can interpret several text formats (plain text, rich text, HTML, etc). For this reason a single mail may contain several text formats for the UA to choose the one that fits the best for the user needs. A mail sent with Outlook will be created by default with two formats, one in HTML and one in text plain for those users with mobile devices that do not understand HTML. This same mail will be presented in HTML format for an Outlook user and in plain text for the mobile user.

The following is an example of such mails.

Example 2. Alternate body with HTML and plain text

```
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Subject: mensaje con vista alterna
Date: Thu, 3 Feb 2011 17:35:48 -0600
MIME-Version: 1.0
Content-Type: multipart/alternative;                          a)
      boundary="----_=_NextPart_001_01CBC3FB.169ED34A"
From: "User1" <user1@training2.tm>
To: "User2" <user2@training2.tm>



This is a multi-part message in MIME format.                    b)

------_=_NextPart_001_01CBC3FB.169ED34A
Content-Type: text/plain;
      charset="iso-8859-1"                                      c)
Content-Transfer-Encoding: quoted-printable


Hola!
=20
Dependiendo del cliente que utilices podr=E1s ver este correo en su =
versi=F3n HTML o en su versi=F3n de texto plano!!
=20
Saludos!
=20

------_=_NextPart_001_01CBC3FB.169ED34A
Content-Type: text/html;
                                                                 d)
```

```
    charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=3DContent-Type content=3D"text/html; =
charset=3Diso-8859-1">
<META content=3D"MSHTML 6.00.3790.3959" name=3DGENERATOR></HEAD>
<BODY>
<DIV><SPAN class=3D062473323-03022011><FONT face=3DArial=20
size=3D5><STRONG>Hola!</STRONG></FONT></SPAN></DIV>
<DIV><SPAN class=3D062473323-03022011><STRONG><FONT face=3DArial=20
size=3D5></FONT></STRONG></SPAN>&nbsp;</DIV>
<DIV><SPAN class=3D062473323-03022011><FONT face=3DArial =
size=3D2><EM>Dependiendo</EM>=20
del cliente que utilices podr=E1s ver este correo en su <STRONG><U><FONT =

color=3D#ff0000>versi=F3n HTML</FONT></U></STRONG> o en su versi=F3n de =
texto=20
plano!!</FONT></SPAN></DIV>
<DIV><SPAN class=3D062473323-03022011><FONT face=3DArial=20
size=3D2></FONT></SPAN>&nbsp;</DIV>
<DIV><SPAN class=3D062473323-03022011><FONT face=3DCourier=20
size=3D2>Saludos!</FONT></SPAN></DIV>
<DIV><SPAN class=3D062473323-03022011><FONT face=3DArial=20
size=3D2></FONT></SPAN>&nbsp;</DIV></BODY></HTML>

------_=_NextPart_001_01CBC3FB.169ED34A--                        }    e)
.
250 2.6.0 <TMW3K01LvUcaL7lbKRg0000000b@tmw3k01.training2.tm> Queued mail for delivery
```

Now let's analyze each of the parts involved in this mail to understand how it was generated.

a) This mail was created to send two slightly different sections. The first section contains the body in plain text for those non HTML capable User Agents. The second section contains exactly the same content but in HTML format. The mail is designed to show only one of the section at any given time, this means the final recipient won't see the same content duplicated in both plain text and HTML format. Based on this, we can conclude we need to use a MIME Content-Type header with multipart/alternative. Multipart defines this mail as composed of more than one section in the same body while alternative specifies that only one body section will be visible. Mail clients use this header to choose which format will be the most appropriate to show the content to the final recipient.

It is worth mentioning here that, because of an efficiency use, the most appropriate format is left to the end, this means, if we were to send both plain text and HTML together, it's advisable to put the plain text first and then HTML, this let mail clients know HTML should be preferred over plain text whenever they are capable of presenting such format. If they can't present the last option, they'll choose the previous, and if this can't be presented too, then the previous, and so on until they reach the first option which should always be plain text. Remember that plain text is a format all mail clients are capable to present because it doesn't involve any particular interpretation.

This definition implies that each part must be separated from each other by means any mail client can implement, otherwise the whole mail would become unreadable. This function is implemented by a modifier parameter called "Boundary" which is responsible of indicating the beginning of each section and the end of the whole MIME format.

This Boundary has four simple and basic rules:

1. It must only contain US-ASCII characters.
2. It may be as long as 68 characters.
3. When the Boundary itself contains the ":" (colon) character, the whole Boundary must be encapsulated using quotes '"'.
4. Only one boundary can be defined for each header. (We'll talk a little more about this rule when we get to more complicated mails).

In our example the boundary is: **----_=_NextPart_001_01CBC3FB.169ED34A**

b) As we saw on 2.2.4.1. about simple bodies, we can close the headers section with two <CRLF> (ENTER) and start writing the mail content right away. For composed mail with a defined Boundary, this is not enough to start writing the body. Any text string that appears after the two <CRLF> characters of the header and before the first presence of the Boundary is known as "Preamble". All this text will be ignored by any SMTP Server as this section is defined for giving just a general description of the body mail structure. In our example, the Preamble has the following description: **"This is a multi-part message in MIME format",** but this won't be interpreted by the SMTP Server.

c) To start writing the first plain text part of the body we must open the section by defining the first Boundary. You should be aware that from the moment you start using the Boundary to start a section inside the mail body, you must follow this syntax:

Syntax: “--"Boundary

At this point you may compare the Boundary definition used in a) and c) to see the difference.

a) **----_=_NextPart_001_01CBC3FB.169ED34A**
c) **------_=_NextPart_001_01CBC3FB.169ED34A**

The Boundary is always defined after a <CRLF> character, followed by two hyphens “--" and ended with the exact same string defined in the Content-Type header followed by a final <CRLF>.

The next line may follow one of the following conditions:

a) The line may start with a Content-Type header with its corresponding definition. In such case the header must be ended with two consecutive <CRLF> as it occurs with the main mail header.

b) The line may be a <CRLF>. In such case, a mail client must assume there are no headers and will treat the following lines as part of a Content-Type: message/rfc822 for a multipart/digest mail or a Content-Type:text/plain for the remaining types.

c) The line may directly start with the corresponding body for that section. In such case all characters should be treated as the header for that section and therefore such characters will not be visible in the mail client when the recipient opens the mail. This is an example of a malformed mail. This kind of mails may be represented in multiple forms depending on the content but it will never be represented as intended in the first place.

After the Boundary definition has been closed, the next part will be the headers but remember these may be present or not depending on the different possibilities already shown.

The headers from our example are defining the content as plain text (Content-Type: text/plain;) using the Latin1 or ISO-8859-1 (charset=ISO-8859-1) charset. You can also notice the body should be presented as it is with no additional modifications or interpretations (Content-Transfer-Encoding: quoted-printable).

You may also notice some strange characters in this body section. This may seem so for a human reading the text, but a mail client will first use the charset definition and look for any corresponding character like the ones shown in this section:

=20 (in Latin1 charset this corresponds to a space character)
=E1 (in Latin1 charset this corresponds to an “á” character)
= (at the end of the third line). It is worth mentioning that RFC 2821 recommends to transfer a maximum of 78 characters per line without including the last <CRLF> even when a maximum of 1000 is supported. Many mail clients will try to fit the content following this rule, is such situation the line must end with a final “=” character, meaning the following line is a continuation. This way the recipient’s mail client will be able to properly understand the syntax and present a continuous line in the final presentation.
=F3 (ó)

d) Once the first plain text part of the mail is finished, the second one will start by defining the boundary again followed by the corresponding headers. The content type to be used will be text/html. The remaining parameters are not changed but the content will now be in HTML format.

e) In order to close the mail the boundary must be defined one last time but using the following format:

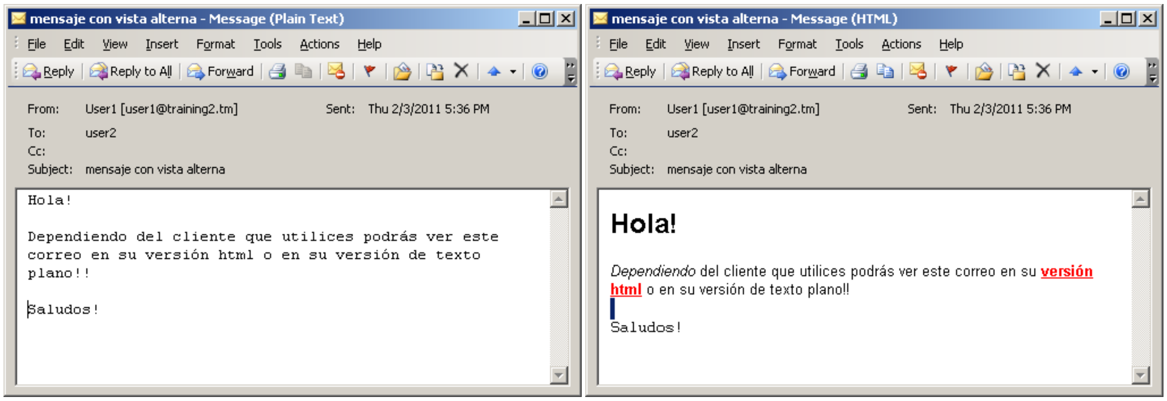
Syntax: “**--**“Boundary”**--**“

In our example the final boundary would be:

------_=_NextPart_001_01CBC3FB.169ED34A--

As with any other mail, the complete body section must be closed by issuing the <CRLF>.<CRLF> sequence. You must be aware that a section will remain between the last boundary and the final closing sequence. This section is called Epilogue. Any data contained here will be ignored by mail clients and it is recommended not to use it to store any kind of information as the mail client will not process it.

The way the final mail will be presented, depends on the interpretation capabilities the mail client has. Our test mail will be presented in one of the following ways:


mensaje con vista alterna - Message (Plain Text)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward
mensaje con vista alterna - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward

From: User1 [user1@training2.tm] Sent: Thu 2/3/2011 5:36 PM
To: user2
Cc:
Subject: mensaje con vista alterna

```
Hola!

Dependiendo del cliente que utilices podrás ver este
correo en su versión html o en su versión de texto
plano!!

Saludos!
```

From: User1 [user1@training2.tm] Sent: Thu 2/3/2011 5:36 PM
To: user2
Cc:
Subject: mensaje con vista alterna

Hola!

Dependiendo del cliente que utilices podrás ver este correo en su **versión html** o en su versión de texto plano!!

Saludos!

Table 3. Alternate mail with plain text and HTML format

In America, the most widely used charset is ISO-8859-1 that extends the original US-ASCII set. The following table shows the representation of all characters belonging to this charset. You may use this table as a reference for mail creation or interpretation.

Dec	Hex	Symbol	Dec	Hex	Symbol	Dec	Hex	Symbol
162	A2	¢	194	C2	Â	226	E2	â
163	A3	£	195	C3	Ã	227	E3	ã
164	A4	¤	196	C4	Ä	228	E4	ä
165	A5	¥	197	C5	Å	229	E5	å
166	A6	¦	198	C6	Æ	230	E6	æ
167	A7	§	199	C7	Ç	231	E7	ç
168	A8	¨	200	C8	È	232	E8	è
169	A9	©	201	C9	É	233	E9	é
170	AA	ª	202	CA	Ê	234	EA	ê
171	AB	«	203	CB	Ë	235	EB	ë
172	AC	¬	204	CC	Ì	236	EC	ì
173	AD	-	205	CD	Í	237	ED	í
174	AE	®	206	CE	Î	238	EE	î
175	AF	¯	207	CF	Ï	239	EF	ï
176	B0	°	208	D0	Ð	240	F0	ð
177	B1	±	209	D1	Ñ	241	F1	ñ
178	B2	²	210	D2	Ò	242	F2	ò
179	B3	³	211	D3	Ó	243	F3	ó
180	B4	´	212	D4	Ô	244	F4	ô
181	B5	µ	213	D5	Õ	245	F5	õ

Dec	Hex	Symbol
182	B6	¶
183	B7	·
184	B8	¸
185	B9	¹
186	BA	º
187	BB	»
188	BC	¼
189	BD	½
190	BE	¾
191	BF	¿
192	C0	À
193	C1	Á
214	D6	Ö
215	D7	×
216	D8	Ø
217	D9	Ù
218	DA	Ú
219	DB	Û
220	DC	Ü
221	DD	Ý
222	DE	Þ
223	DF	ß
224	E0	à
225	E1	á
246	F6	ö
247	F7	÷
248	F8	ø
249	F9	ù
250	FA	ú
251	FB	û
252	FC	ü
253	FD	ý
254	FE	þ
255	FF	ÿ

Table 4. ISO 8859-1 characters (only non US-ASCII characters are shown)

Mails that include attachments will be explained in the following section.

2.2.5. Attachments

Before getting started with how mail attachments are creating, let’s start by noticing that in SMTP an attachment is any piece of data attached to the mail body. Based on this, even the text that constitutes the body of the mail should be treated as an attachment. When we talk about multipart/alternative bodies, these also constitute a mail with at least two attachment blocks where only one of them will be presented to the recipient.

By understanding this concept correctly, it will be easy to assimilate that when handling mail attachments they may represent not only audio or video files but instead, the whole body may be understood as an empty box were you can insert section parts like text, images, documents in very precise blocks.

The following tables show the most common MIME types that can be used when attaching several types of data to mail body.

This table shows the MIME types and subtypes for Application content

Type/subtype	Extension	Type/subtype	Extension	Type/subtype	Extension
application/envoy	evy	application/vnd.ms-pkiseccat	cat	application/x-msmediaview	mvb
application/fractals	fif	application/vnd.ms-pkistl	stl	application/x-msmetafile	wmf
application/futuresplash	spl	application/vnd.ms-powerpoint	pot	application/x-msmoney	mny

Type/subtype	Extension	Type/subtype	Extension	Type/subtype	Extension
application/hta	hta	application/vnd.ms-powerpoint	pps	application/x-mspublisher	pub
application/internet-property-stream	acx	application/vnd.ms-powerpoint	ppt	application/x-msschedule	scd
application/mac-binhex40	hqx	application/vnd.ms-project	mpp	application/x-msterminal	trm
application/msword	doc	application/vnd.ms-works	wcm	application/x-mswrite	wri
application/msword	dot	application/vnd.ms-works	wdb	application/x-netcdf	cdf
application/octet-stream	*	application/vnd.ms-works	wks	application/x-netcdf	nc
application/octet-stream	bin	application/vnd.ms-works	wps	application/x-perfmon	pma
application/octet-stream	class	application/winhlp	hlp	application/x-perfmon	pmc
application/octet-stream	dms	application/x-bcpio	bcpio	application/x-perfmon	pml
application/octet-stream	exe	application/x-cdf	cdf	application/x-perfmon	pmr
application/octet-stream	lha	application/x-compress	z	application/x-perfmon	pmw
application/octet-stream	lzh	application/x-compressed	tgz	application/x-pkcs12	p12
application/oda	oda	application/x-cpio	cpio	application/x-pkcs12	pfx
application/olescript	axs	application/x-csh	csh	application/x-pkcs7-certificates	p7b
application/pdf	pdf	application/x-director	dcr	application/x-pkcs7-certificates	spc
application/pics-rules	prf	application/x-director	dir	application/x-pkcs7-cerreqresp	p7r
application/pkcs10	p10	application/x-director	dxr	application/x-pkcs7-mime	p7c
application/pkix-crl	crl	application/x-dvi	dvi	application/x-pkcs7-mime	p7m
application/postscript	ai	application/x-gtar	gtar	application/x-pkcs7-signature	p7s
application/postscript	eps	application/x-gzip	gz	application/x-sh	sh
application/postscript	ps	application/x-hdf	hdf	application/x-shar	shar
application/rtf	rtf	application/x-internet-signup	ins	application/x-shockwave-flash	swf
application/set-payment-initiation	setpay	application/x-internet-signup	isp	application/x-stuffit	sit
application/set-registration-initiation	setreg	application/x-iphone	iii	application/x-sv4cpio	sv4cpio

Type/subtype	Extension	Type/subtype	Extension	Type/subtype	Extension
application/vnd.ms-excel	xla	application/x-javascript	js	application/x-sv4crc	sv4crc
application/vnd.ms-excel	xlc	application/x-latex	latex	application/x-tar	tar
application/vnd.ms-excel	xlm	application/x-msaccess	mdb	application/x-tcl	tcl
application/vnd.ms-excel	xls	application/x-mscardfile	crd	application/x-tex	tex
application/vnd.ms-excel	xlt	application/x-msclip	clp	application/x-texinfo	texi
application/vnd.ms-excel	xlw	application/x-msdownload	dll	application/x-texinfo	texinfo
application/vnd.ms-outlook	msg	application/x-msmediaview	m13	application/x-troff	roff
application/vnd.ms-pkicertstore	sst	application/x-msmediaview	m14	application/zip	zip

Table 5. Application MIME Types

This table shows the MIME Types and subtypes for Audio content.

Type/subtype	Extension
audio/basic	au
audio/basic	snd
audio/mid	mid
audio/mid	rmi
audio/mpeg	mp3
audio/x-aiff	aif
audio/x-aiff	aifc
audio/x-aiff	aiff
audio/x-mpegurl	m3u
audio/x-pn-realaudio	ra
audio/x-pn-realaudio	ram
audio/x-wav	wav

Table 6. Audio MIME Types

This table shows the MIME Types and subtypes for IMAGE content.

Type/subtype	Extension
image/bmp	bmp
image/cis-cod	cod
image/gif	gif
image/ief	ief
image/jpeg	jpe
image/jpeg	jpeg
image/jpeg	jpg
image/pipeg	jfif
image/png	png
image/svg+xml	svg

Type/subtype	Extension
image/tiff	tif
image/tiff	tiff
image/x-cmu-raster	ras
image/x-cmx	cmx
image/x-icon	ico
image/x-portable-anymap	pnm
image/x-portable-bitmap	pbm
image/x-portable-graymap	pgm
image/x-portable-pixmap	ppm
image/x-rgb	rgb
image/x-xbitmap	xbm
image/x-xpixmap	xpm
image/x-xwindowdump	xwd

Table 7.Image MIME Types

This table shows the MIME Types and subtypes for Message content.

Type/subtype	Extension
message/rfc822	mht
message/rfc822	mhtml
message/rfc822	nws

Table 8. Message MIME Types

This table shows the MIME Types and subtypes for TEXT content.

Type/subtype	Extension
text/css	css
text/h323	323
text/html	htm
text/html	html
text/html	stm
text/iuls	uls
text/plain	bas
text/plain	c
text/plain	h
text/plain	txt
text/richtext	rtx
text/scriptlet	sct
text/tab-separated-values	tsv
text/webviewhtml	htt
text/x-component	htc
text/x-setext	etx
text/v-card	vcf

Table 9.TEXT MIME Types

This table shows the MIME Types and subtypes for VIDEO content.

Type/subtype	Extension
video/mpeg	mp2
video/mpeg	mpa
video/mpeg	mpe
video/mpeg	mpeg
video/mpeg	mpg
video/mpeg	mpv2
video/quicktime	mov
video/quicktime	qt
video/x-la-asf	lsf
video/x-la-asf	lsx
video/x-ms-asf	asf
video/x-ms-asf	asr
video/x-ms-asf	asx
video/x-msvideo	avi
video/x-sgi-movie	movie

Table 10.Video MIME Types

2.2.5.1. Multipart/mixed

To get started, will first talk about how data is transmitted by SMTP protocol. From what have been previously presented, we already know RFC 2821 permits only the use of the US-ASCII charset along the whole SMTP conversation, so a model was needed to allow the transmission of any data using only this fixed charset.

The base64 algorithm was introduced to achieve this. Its goal is to allow the encoding of any type of data by using exclusively US-ASCII contained characters. RFC 2046 section 6.8 describes the mechanism to be implemented by any data encoding process. The following paragraph is extracted from the original RFC but it is to be noted this information may only be useful if you need to develop applications or modules that will actually encode / decode using base64. An explanation of the whole algorithm is beyond the scope of this book.

6.8. Base64 Content-Transfer-Encoding

The Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. The encoding and decoding algorithms are simple, but the encoded data are consistently only about 33 percent larger than the unencoded data. This encoding is virtually identical to the one used in Privacy Enhanced Mail (PEM) applications, as defined in RFC 1421.

A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=", is used to signify a special processing function.)

[…]

The encoding process represents 24-bit groups of input bits as output strings of 4 encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating 3 8bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each of which is translated into a single digit in the base64 alphabet. When encoding a bit stream via the base64 encoding, the bit stream must be presumed to be ordered with the most-significant-bit first. That is, the first bit in the stream will be the high-order bit in the first 8bit byte, and the eighth bit will be the low-order bit in the first 8bit byte, and so on.

Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string. These characters, identified in Table 1, below, are selected so as to be universally representable, and the set excludes characters with particular significance to SMTP (e.g., ".", CR, LF) and to the multipart boundary delimiters defined in RFC 2046 (e.g., "-").

Table 1: The Base64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

The encoded output stream must be represented in lines of no more than 76 characters each. All line breaks or other characters not found in Table 1 must be ignored by decoding software. In base64 data, characters other than those in Table 1, line breaks, and other white space probably indicate a transmission error, about which a warning message or even a message rejection might be appropriate under some circumstances.

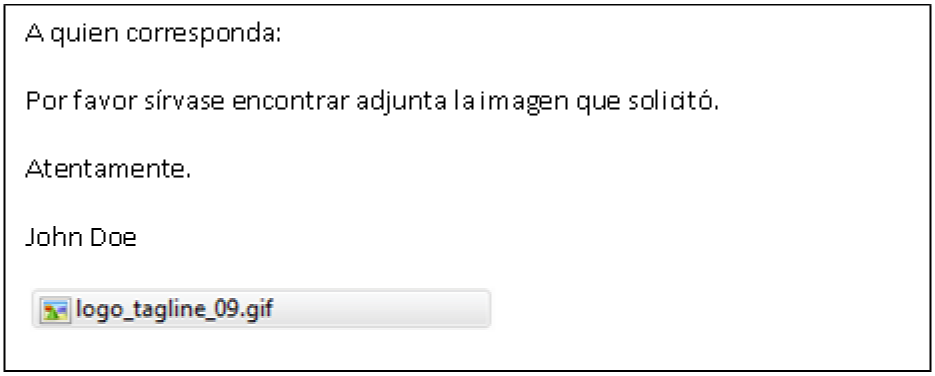
Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a body. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the "=" character. Since all base64 input is an integral number of octets, only the following cases can arise: (1) the final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be an integral multiple of 4 characters with no "=" padding, (2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or (3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character.

Because it is used only for padding at the end of the data, the occurrence of any "=" characters may be taken as evidence that the end of the data has been reached (without truncation in transit). No such assurance is possible, however, when the number of octets transmitted was a multiple of three and no "=" characters are present.

Any characters outside of the base64 alphabet are to be ignored in base64-encoded data.

Care must be taken to use the proper octets for line breaks if base64 encoding is applied directly to text material that has not been converted to canonical form. In particular, text line breaks must be converted into CRLF sequences prior to base64 encoding. The important thing to note is that this may be done directly by the encoder rather than in a prior canonicalization step in some implementations.

To better understand how to use the Media-Type Multipart/mixed we'll now try to generate a mail that is able to transmit the following message block:



A quien corresponda:

Por favor sírvase encontrar adjunta la imagen que solicitó.

Atentamente.

John Doe

Image 17. Using the Multipart/mixed Media-Type

In order to understand how a message like this should be transmitted we first need to understand how the message is composed. By just looking at the structure it is easy to see this mail is composed of two separate blocks: one in plain text and another with an attached image. This is enough to know we need a main definition of multipart/mixed as we will really transmit a mixed content within the same message.

The first part of the header will be something like this:

```
From:"user1"<user1@training2.tm>
To:"user2"<user2@training2.tm>
Subject: Prueba de correo con adjuntos
Date:Mon, 5 Oct 1980 18:00:00 -0600
MIME-Version: 1.0
Content-Type:multipart/mixed;
      boundary=uno
```

Defines the message as composed of more than one “part” with mixed content. It also establishes the “boundary” used to separate each of the parts.

```
This is a multipart message in MIME format

--uno

Content-Type:text/plain;charset=iso-8859-1

Content-Transfer-Encoding:quoted-printable
```

This header defines a plain text section with charset ISO 8859-1 and because it is text we’re defining a coding mechanism of Quoted-Printable to present the characters as they are sent.

```
A quien corresponda:

Por favor s=EDrvase encontrar adjunta la imagen que solicit=F3.

Atentamente.

John Doe
```

This is the coded plain text using charset ISO 8859-1.

```
--uno

Content-Type: image/gif;

            name="logo_tagline_09.gif"

Content-Transfer-Encoding: base64

Content-ID: <image1>

Content-Description: logo_tagline_09.gif

Content-Location: logo_tagline_09.gif
```

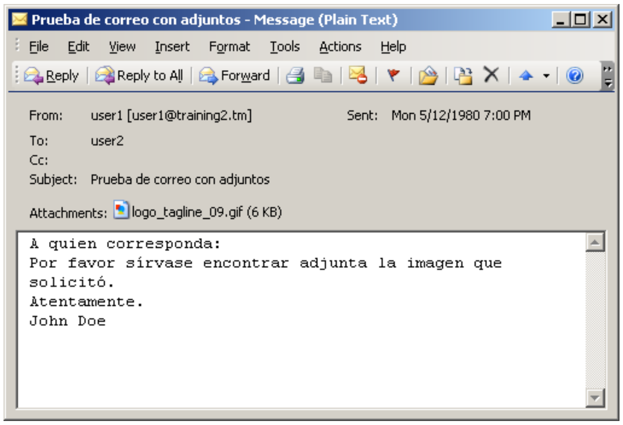
Now we define a new header to start the transfer of the image with a Content-Type: image/gif which also accepts the parameter “name=” that stores the original file name. Content-Transfer-Encoding: base64 indicates the following block is composed of encoded data and not a readable representation of the original content. Content-ID is a unique identifier for this block that may be used as a reference for other bodies. Content-Description is just a description of the file while Content-Location tells the complete path to reach the file.

```
R0lGODlhMQEyAPcAAPmxme45Ls7MzP718tLHye4rKfb29oKAgY+MjfijiPBUPOnp6fnCtOwcJCkl

Jvzh19ExMebl5vN9ZOgPF8TDxPmskuGopsy3t+3t7vSCZdcalfva2vu9qfRybqWkpERAQZWTk/ed

gfJsUfFgRPi5vPNpTHZzdOu5te9KNd7d3Tg0Nf3o4vNyVfzTw+iOgvr6+v3x7uLh4fz8/PPy8vBd

Q//cyu9BMPWLbeOWjPeWmZqYmfWOcfR7Xf7s5v3k3tXV1fmtmu0hJdHQ0fqdjNrZ2faSg7Kysl1Z

[…]

Clmw2QRQBslt3AQgAPFQCaEQA1+d04dN1rJN2wLR2NAdFtBUDdwcDAYw0sVQ0sgQActlEAO9HQri

HQopEAPLEAE4zdo6fQ2wTc+tN93wnR4ZyM/BMNkjXdkYgAEmuN/7VwxpQA0Wzd76oNH1HN8Eld0G

DhYmMg+zYA0ylM/DMAz1bQAUXuEGgA1NEOEvQA70fDoJ/uGoERAAOw==
```

This is the base64 representation of the “gif” image being transmitted.

--uno--.

Once received, the mail will be represented as we intended:

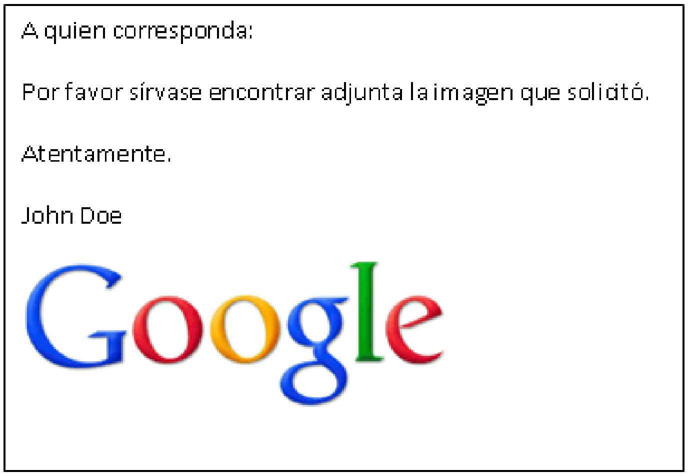

Prueba de correo con adjuntos - Message (Plain Text)
File Edit View Insert Format Tools Actions Help
Reply Reply to All Forward

From: user1 [user1@training2.tm]
Sent: Mon 5/12/1980 7:00 PM
To: user2
Cc:
Subject: Prueba de correo con adjuntos
Attachments: logo_tagline_09.gif (6 KB)

```
A quien corresponda:
Por favor sírvase encontrar adjunta la imagen que
solicitó.
Atentamente.
John Doe
```

Image 18. Multipart/mixed message with an attached image

Now let’s transfer a message that will show an image as a signature:



A quien corresponda:

Por favor sírvase encontrar adjunta la imagen que solicitó.

Atentamente.

John Doe

Image 19. Message with attached "show in line" image

We can manage to send a mail like this by adjusting the Content-Type header to “text/html” to include the image as part of the body. Remember that when dealing with attachments we have the Content-ID parameter we can use to “call” or “insert” a data block inside another body. The following structure shows how to create this kind of mails.

```
From:"user1"<user1@training2.tm>

To:"user2"<user2@training2.tm>

Subject: Prueba de correo con adjuntos

Date:Mon, 12 May 1980 18:00:00 -0600

MIME-Version: 1.0

Content-Type:multipart/mixed;

      boundary=uno
```

The main header will again define a message as composed of mixed content. It will also define the boundary used to separate each data block.

```
This is a multipart message in MIME format
```

```
--uno

Content-Type:text/html;charset=iso-8859-1

Content-Transfer-Encoding:quoted-printable
```

Now we define an HTML text body with quoted-printable text presentation using charset ISO 8859-1.

```
<html>

<body>

<p>A quien corresponda:</p>

<p>Por favor s=EDrvase encontrar adjunta la imagen que solicit=F3.</p>

<p>Atentamente.</p>

<p>John Doe</p>

<br>

<img src=3D"cid:image1">

</body>

</html>
```

HTML Code. Let’s focus on the <img src=3D” cid:image1”> html tag. “=3D” is the html representation of the equal symbol “=” this is to avoid any HTML interpretation problem. The “cid:” parameter points to the image “Content-ID” we want to show in this exact position. Here the ID doesn’t show the “<” and “>” symbols to avoid any HTML representation problem.

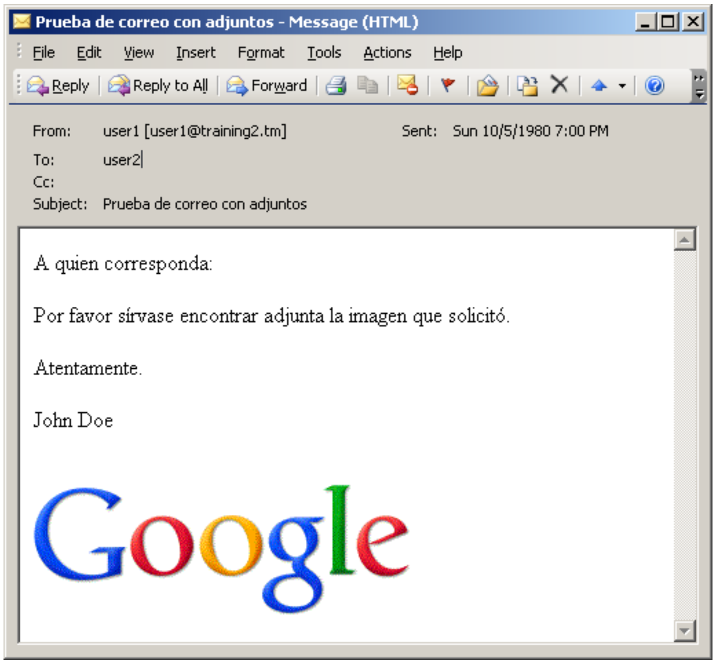
```
--uno

Content-Type: image/gif;

          name="logo_tagline_09.gif"

Content-Transfer-Encoding: base64

Content-ID: <image1>

Content-Description: logo_tagline_09.gif

Content-Location: logo_tagline_09.gif
```

This header encodes the “logo_tagline_09.gif” image in MIME format using the base64 algorithm. It also identifies it with the unique ID “Content-ID:<image_1>”, this allows it to be “called” or “inserted” in any other body part.

R0lGODlhMQEyAPcAAPmxme45Ls7MzP718tLHye4rKfb29oKAgY+MjfijiPBUPOnp6fnCtOwcJCkl
Jvzh19ExMebl5vN9ZOgPF8TDxPmskuGopsy3t+3t7vSCZdcalfva2vu9qfRybqWkpERAQZWTk/ed
gfJsUfFgRPi5vPNpTHZzdOu5te9KNd7d3Tg0Nf3o4vNyVfzTw+iOgvr6+v3x7uLh4fz8/PPy8vBd
[…]
Clmw2QRQBsIt3AQgAPFQCaEQA1+d04dN1rJN2wLR2NAdFtBUDdwcDAYw0sVQ0sgQActlEAO9HQri
HQopEAPLEAE4zdo6fQ2wTc+tN93wnR4ZyM/BMNkjXdkYgAEmuN/7VwxpQA0Wzd76oNH1HN8Eld0G
DhYmMg+zYA0yIM/DMAz1bQAUXuEGgA1NEOEvQA70fDoJ/uGoERAAOw==

```
--uno--
.
```

Once received, the mail will be shown exactly as intended:



From: user1 [user1@training2.tm] Sent: Sun 10/5/1980 7:00 PM
To: user2
Cc:
Subject: Prueba de correo con adjuntos

A quien corresponda:

Por favor sírvase encontrar adjunta la imagen que solicitó.

Atentamente.

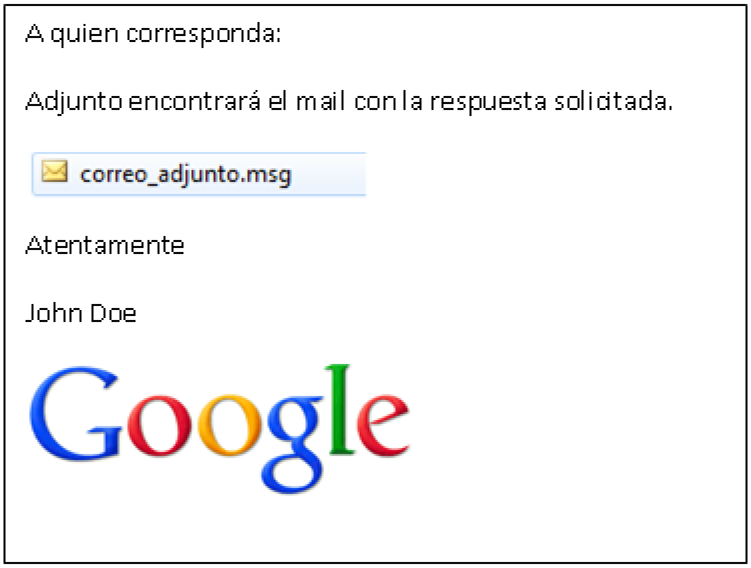
John Doe

Image 20. Multipart/mixed with a visible signature image

2.2.5.2. Multipart/Digest

The Multipart/Digest header is used when there's a need to send a mail as an attachment of a main message. The header helps the Mail Client (e.g. Outlook) to correctly interpret there is at least one composed body with an RFC 822 IMF format compliant mail attached.

An example of this is shown in the image below:



A quien corresponda:

Adjunto encontrará el mail con la respuesta solicitada.

correo_adjunto.msg

Atentamente

John Doe

Image 21. Mail with another mail attached

Let's now analyze how to create each of the parts needed for a message like this.

```
From:"user1"<user1@training2.tm>
To:"user2"<user2@training2.tm>
Subject: Prueba de correo con adjuntos


Date:Mon, 05 Oct 2012 18:00:00 -0600

MIME-Version: 1.0

Content-Type:multipart/mixed;

      boundary=uno
```

First of all we need to define the message as composed of multiple parts of mixed content and boundary "uno" used to separate them.

```
This is a multipart message in MIME format


--uno

Content-Type:text/html;charset=iso-8859-1

Content-Transfer-Encoding:quoted-printable
```

HTML body for the main message.

```
<html>

<body>

<p>A quien corresponda:</p>

<p>Adjunto encontrar=E1 el mail con la respuesta solicitada.</p>

<p>Atentamente.</p>

<p>John Doe</p>

<br>

<img src=3D"cid:image1">

</body>

</html>
```

HTML code for the main body. Notice there is an image inserted by calling its Content-ID.

```
--uno

Content-Type:multipart/digest;

        boundary=dos
```

Now let's define a new multipart/digest section to inform the mail client that the body contained within boundary "dos" belongs to an rfc822 complaint format.

```
--dos

Content-Type:message/rfc822;
```

Boundary "dos" starts the attached mail structure by defining a media-type of message/rfc822. Notice here that boundary "dos" is used instead "uno" to separate the attached message from the main body.

```
From:user3

To:User4

Date:Fri, 13 May 1981 19:00:00 -0600

Subject:Correo adjunto


Este correo contiene el formulario de respuesta solicitado anteriormente.

Saludos.
```

This is the attached mail in rfc822 format. As with any other mail, it has its own headers separated from the body by two <CRLF>.

```
--dos--
```

Boundary "dos" here is closed to finish the structure for the attached mail.

```
--uno
Content-Type: image/gif;
          name="logo_tagline_09.gif"
Content-Transfer-Encoding: base64
Content-ID: <image1>
Content-Description: logo_tagline_09.gif
Content-Location: logo_tagline_09.gif
```

Now that we are back to the boundary “uno” scope, we need to encode the “logo_tagline_09.gif” image inserted in the main body.

R0lGODlhMQEyAPcAAPmxme45Ls7MzP718tLHye4rKfb29oKAgY+MjfijiPBUPOnp6fnCtOwcJCkl

Jvzh19ExMebl5vN9ZOgPF8TDxPmskuGopsy3t+3t7vSCZdcalfva2vu9qfRybqWkpERAQZWTk/ed

gfJsUfFgRPi5vPNpTHZzdOu5te9KNd7d3Tg0Nf3o4vNyVfzTw+iOgvr6+v3x7uLh4fz8/PPy8vBd

[...]

Clmw2QRQBsIt3AQgAPFQCaEQA1+d04dN1rJN2wLR2NAdFtBUDdwcDAYw0sVQ0sgQActlEAO9HQri

HQopEAPLEAE4zdo6fQ2wTc+tN93wnR4ZyM/BMNkjXdkYgAEmuN/7VwxpQA0Wzd76oNH1HN8EId0G

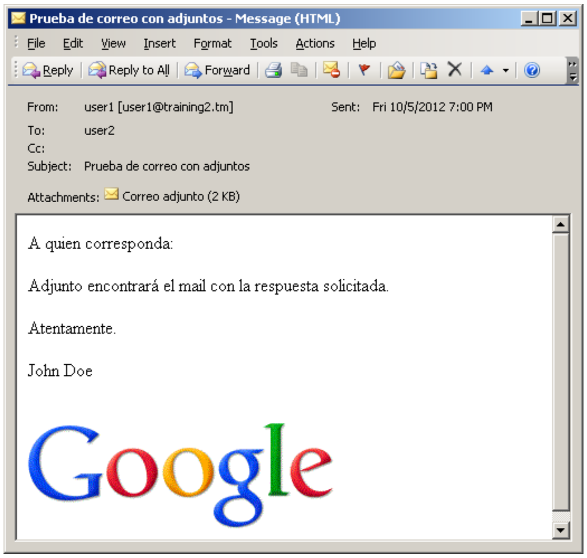
DhYmMg+zYA0yIM/DMAz1bQAUXuEGgA1NEOEvQA70fDoJ/uGoERAAOw==

Signature image encoded in base64.

```
--uno--
.
```

The final step is to close boundary “uno” and close the mail transmission with <CRLF>.<CRLF>

The final recipient will receive this mail as shown in the image below:


Prueba de correo con adjuntos - Message (HTML)
File Edit View Insert Format Tools Actions Help
Reply
Reply to All
Forward

From: user1 [user1@training2.tm] Sent: Fri 10/5/2012 7:00 PM
To: user2
Cc:
Subject: Prueba de correo con adjuntos
Attachments: Correo adjunto (2 KB)

A quien corresponda:

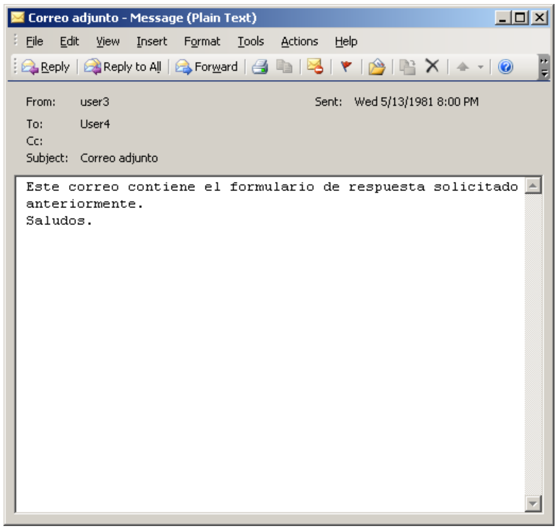
Adjunto encontrará el mail con la respuesta solicitada.

Atentamente.

John Doe

Google

Image 22. Main Multipart/digest message



From: user3

Sent: Wed 5/13/1981 8:00 PM

To: User4

Cc:

Subject: Correo adjunto

```
Este correo contiene el formulario de respuesta solicitado
anteriormente.
Saludos.
```

Image 23. Attached Message/rfc822 mail

The objective of these examples is to show the main procedures used to create and interpret an electronic mail. They are intended to show the rules that must be followed by any mail structure in order for the mail to be received by the final recipient. As you have seen, several types of mails can be generated by combining any of the methods already shown. At the end of this section you should now be ready to correctly understand any mail coding structure and the location of the mail parts inside a plain text transmission no matter where the mail comes from or its contents.

2.2.6. Reply/Error Codes

Now that we are familiar with the mechanisms used to generate, transfer and read an email, it's time to review the codes used to transmit the status of the transmission.

RFC 2821 defines the main codes used as an answer to every SMTP command sent by the SMTP Client. However, for some conditions, these codes are not specific enough to give Mail administrators a clue about remediation actions. In January 2003 a new standard was generated to extend the actual number of codes and its interpretation. This new list includes a much wider number of conditions and gives much more useful information for remediation actions. In the following two sections we'll analyze both types of codes in order to take according actions when the mail flow gets affected or interrupted.

2.2.6.1. Main Status Codes

MTA servers communicate with each other by means of transmitting a 3 numeric digit code, each digit specifying a certain condition that will allow mail servers to take the appropriate decisions. Any system could determine its next action after examining the first digit; by examining the second digit, a more certain approximation of the condition can be made, and by examining the third one a more granular description can be obtained.

The definition of the these codes can be found on RFC 2821 under section 4.2.1. The following list complements these definitions with real life examples of conditions, responses and possible remediation actions when applicable.

4.2.1 Reply Code Severities and Theory

```
 [...]

There are five values for the first digit of the reply code:

   1yz   Positive Preliminary reply
      The command has been accepted, but the requested action is being
      held in abeyance, pending confirmation of the information in this
      reply.  The SMTP client should send another command specifying
      whether to continue or abort the action.  Note: un-extended SMTP
      does not have any commands that allow this type of reply, and so
      does not have continue or abort commands.

   2yz   Positive Completion reply
      The requested action has been successfully completed.  A new
      request may be initiated.

   3yz   Positive Intermediate reply
      The command has been accepted, but the requested action is being
      held in abeyance, pending receipt of further information.  The
      SMTP client should send another command specifying this
      information.  This reply is used in command sequence groups (i.e.,
      in DATA).

   4yz   Transient Negative Completion reply
      The command was not accepted, and the requested action did not
      occur.  However, the error condition is temporary and the action
      may be requested again.  The sender should return to the beginning
      of the command sequence (if any).  It is difficult to assign a
      meaning to "transient" when two different sites (receiver- and
      sender-SMTP agents) must agree on the interpretation.  Each reply
      in this category might have a different time value, but the SMTP
      client is encouraged to try again.  A rule of thumb to determine
      whether a reply fits into the 4yz or the 5yz category (see below)
      is that replies are 4yz if they can be successful if repeated
      without any change in command form or in properties of the sender
      or receiver (that is, the command is repeated identically and the
      receiver does not put up a new implementation.)
```

Errors under this category will always end up with the SMTP Client *queuing the rejected mail. This is the main reason for performance degradation and slow delivery problems.*

5yz   Permanent Negative Completion reply
The command was not accepted and the requested action did not occur. The SMTP client is discouraged from repeating the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct the SMTP client to reinitiate the command sequence by direct action at some point in the future (e.g., after the spelling has been changed, or the user has altered the account status).

Errors under this category will NEVER end up in queuing problems as the SMTP client should delete its original copy of the mail and send an NDR (Non-Delivery-Response) to the original sender informing the reason for which the mail was not delivered.

[...]

The second digit encodes responses in specific categories:

x0z   Syntax: These replies refer to syntax errors, syntactically correct commands that do not fit any functional category, and unimplemented or superfluous commands.

x1z   Information: These are replies to requests for information, such as status or help.

x2z   Connections: These are replies referring to the transmission channel.

x3z   Unspecified.

x4z   Unspecified.

x5z   Mail system: These replies indicate the status of the receiver mail system vis-a-vis the requested transfer or other mail system action.

The third digit gives a finer gradation of meaning in each category specified by the second digit. The list of replies illustrates this. Each reply text is recommended rather than mandatory, and may even change according to the command with which it is associated. On the other hand, the reply codes must strictly follow the specifications in this section. Receiver implementations should not invent new codes for slightly different situations from the ones described here, but rather adapt codes already defined.

This RFC allows for the status codes to show text for human interpretation in more than one line when the condition requires it as long as the following syntax rules are met:

The reply text may be longer than a single line; in these cases the

> complete text must be marked so the SMTP client knows when it can stop reading the reply. This requires a special format to indicate a multiple line reply.
>
> The format for multiline replies requires that every line, except the last, begin with the reply code, followed immediately by a hyphen, "-" (also known as minus), followed by text. The last line will begin with the reply code, followed immediately by <SP>, optionally some text, and <CRLF>. As noted above, servers SHOULD send the <SP> if subsequent text is not sent, but clients MUST be prepared for it to be omitted.

An example of this implementation is the response to the EHLO command:

```
250-hub.rskala.com Hello [192.168.0.89]

250-SIZE
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-STARTTLS
250-X-ANONYMOUSTLS
250-AUTH NTLM
250-X-EXPS GSSAPI NTLM
250-8BITMIME
250-BINARYMIME
250-CHUNKING
250-XEXCH50
250 XRDST
```

All of these lines should be considered as part of a single message from the SMTP Server. All of them are part of the same 250 reply code. Notice there is a space after the last 250 indicating the end of the reply.

This example confirms the previous syntax. As noted above, you should notice the last line ends with a space after the 250 numeric code, indicating the end of the reply description.

The following table shows an ordered list of the RFC 2821 numeric codes as they are shown in the standard.

Code	Description
211	System status, or system help reply
214	Help message (Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user)
220	<domain> Service ready
221	<domain> Service closing transmission channel
250	Requested mail action okay, completed
251	User not local; will forward to <forward-path>
252	Cannot VRFY user, but will accept message and attempt delivery
354	Start mail input; end with <CRLF>.<CRLF>

Code	Description
421	<domain> Service not available, closing transmission channel (This may be a reply to any command if the service knows it must shut down)
450	Requested mail action not taken: mailbox unavailable (e.g., mailbox busy or temporarily blocked for policy reasons)
451	Requested action aborted: local error in processing
452	Requested action not taken: insufficient system storage
455	Server unable to accommodate parameters
500	Syntax error, command unrecognized (This may include errors such as command line too long)
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
550	Requested action not taken: mailbox unavailable (e.g., mailbox not found, no access, or command rejected for policy reasons)
551	User not local; please try <forward-path>
552	Requested mail action aborted: exceeded storage allocation
553	Requested action not taken: mailbox name not allowed (e.g., mailbox syntax incorrect)
554	Transaction failed (Or, in the case of a connection-opening response, “No SMTP service here”)
555	MAIL FROM/RCPT TO parameters not recognized or not implemented

Table 11. Reply / Error codes from RFC 2821

2.2.6.2. Extended Status Codes

As seen in the previous section, the number of situations that can be addressed with the main status codes defined by RFC 2821 is not enough to cover the wide range of conditions a mail transfer may face. RFC 3463 defines a new set to cover this need. The next paragraph is an extract of the RFC defining this new set.

> SMTP [SMTP] error codes have historically been used for reporting mail system errors. Because of limitations in the SMTP code design, these are not suitable for use in delivery status notifications. SMTP provides about 12 useful codes for delivery reports. The majority of the codes are protocol specific response codes such as the 354 response to the SMTP data command. Each of the 12 useful codes are overloaded to indicate several error conditions. SMTP suffers some scars from history, most notably the unfortunate damage to the reply code extension mechanism by uncontrolled use. This proposal facilitates future extensibility by requiring the client to interpret unknown error codes according to the theory of codes while requiring servers to register new response codes.

In this section we’ll show the advantages of this new set and the way it extends the original codes. We’ll start by showing the structure this new set must follow:

```
   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
```

The “class” parameter defines the general category to which the code belongs to. This may be 2-Success, 4-Transient Failure and 5-Permanent Failure. Its definition is given in by the RFC described below:

```
      2.XXX.XXX   Success

         Success specifies that the DSN is reporting a positive delivery
         action.  Detail sub-codes may provide notification of
         transformations required for delivery.


      4.XXX.XXX   Persistent Transient Failure

         A persistent transient failure is one in which the message as
         sent is valid, but persistence of some temporary condition has
         caused abandonment or delay of attempts to send the message.
         If this code accompanies a delivery failure report, sending in
         the future may be successful.

      5.XXX.XXX   Permanent Failure

         A permanent failure is one which is not likely to be resolved
         by resending the message in the current form.  Some change to
         the message or the destination must be made for successful
         delivery.
```

The “subject” parameter specifies the notification status. The meaning of this value applies to any of the three values previously presented. The values for “subject” may be one of the following:

```
      X.0.XXX   Other or Undefined Status

         There is no additional subject information available.

      X.1.XXX Addressing Status

         The address status reports on the originator or destination
         address.  It may include address syntax or validity.  These
         errors can generally be corrected by the sender and retried.

      X.2.XXX Mailbox Status
```

Mailbox status indicates that something having to do with the mailbox has caused this DSN. Mailbox issues are assumed to be under the general control of the recipient.

X.3.XXX Mail System Status

Mail system status indicates that something having to do with the destination system has caused this DSN. System issues are assumed to be under the general control of the destination system administrator.

X.4.XXX Network and Routing Status

The networking or routing codes report status about the delivery system itself. These system components include any necessary infrastructure such as directory and routing services. Network issues are assumed to be under the control of the destination or intermediate system administrator.

X.5.XXX Mail Delivery Protocol Status

The mail delivery protocol status codes report failures involving the message delivery protocol. These failures include the full range of problems resulting from implementation errors or an unreliable connection.

X.6.XXX Message Content or Media Status

The message content or media status codes report failures involving the content of the message. These codes report failures due to translation, transcoding, or otherwise unsupported message media. Message content or media issues are under the control of both the sender and the receiver, both of which must support a common set of supported content-types.

X.7.XXX Security or Policy Status

The security or policy status codes report failures involving policies such as per-recipient or per-host filtering and cryptographic operations. Security and policy status issues are assumed to be under the control of either or both the sender and recipient. Both the sender and recipient must permit the exchange of messages and arrange the exchange of necessary keys and certificates for cryptographic operations.

The “detail” parameter specifies the particular condition reported by status code. This value applies to each of the “subject” values. The values “detail” may have are shown below:

3.1 Undefined Status and Others

```
      X.0.0   Other undefined Status

         Other undefined status is the only undefined error code.  It
         should be used for all errors for which only the class of the
         error is known.
Examples:
C> RSET
S> 250 2.0.0 user Resetting
Or
S> 250 2.0.0 Resetting
```

This is a very common response code for the RSET command. This command is used to indicate the SMTP Server it must ignore any previous information and to clear all of its buffers to receive a completely new mail.

```
dsn=4.0.0, 452 Too many recipients received this hour (in reply to RCPT
TO command))
```

This code indicates a temporal rejection because the SMTP Server is configured to accept only certain number of recipients in a given time window. From the SMTP Client side there are no actions to mitigate this condition but to wait until the SMTP Server is able to receive connections again.

```
dsn=4.0.0, 451 unable to verify user (in reply to RCPT TO command))
```

In this case, the SMTP Server indicates the mailbox couldn't get validated, this may be because it doesn't exist, mailbox is an undefined alias or the mailbox is temporarily disabled.

```
dsn=4.0.0, 451 qq trouble in home directory (#4.3.0) (in reply to end of
DATA command))
```

This code indicates a problem with the SMTP Server resources, there are no actions to be made from the SMTP Client side to correct this condition, the responsibility belongs to the SMTP Server only.

```
dsn=4.0.0, 451 Could not load DRD for domain (domain.com) rcpt
(user@domain.com) (in reply to RCPT TO command))
```

This code indicates a configuration problem. This may be a relay related problem on the SMTP Server, there are no actions from the SMTP Client side to mitigate this condition.

```
dsn=4.0.0, 452 <user@domain.com> Mailbox size limit exceeded (in reply to
RCPT TO command))
```

This code indicates the recipient's mailbox has reached its storage limit and won't accept new mail until some free space is available. The only one that can resolve this condition is either the mail administrator or the mailbox owner by increasing the storage limit or erasing old mails.

```
dsn=4.0.0, 451 Requested action aborted: local error in processing (code:
11) (in reply to RCPT TO command))
```

This code indicates a configuration / performance problem on the SMTP Server side.

```
dsn=4.0.0, 451 The server is too busy, please try again later (in reply
to RCPT TO command))

dsn=4.0.0, 452 Requested action not taken: insufficient system storage
(in reply to MAIL FROM command))
```

These examples show a condition where the SMTP Server has exhausted its available resources to process new mails. This condition must be fixed on the SMTP Server side.

```
dsn=4.0.0, 451 lowest numbered MX record points to local host (in reply
to RCPT TO command))
```

This code indicates the SMTP Server is probably an SMTP Proxy, this is, an anti-spam gateway appliance that first cleans all mail flow and then delivers the cleaned mail to the internal server. In this case, the error is telling the SMTP Client that the relay for the destination domain is actually pointing to itself, a condition that may provoke a loop where the mail won’t be able to get out of the server. This error is commonly a relay or smart host or delivery routes configuration problem that must be fixed on the SMTP Server side.

```
dsn=4.0.0, 450 <user@domain.com>: User unknown in local recipient table
(in reply to RCPT TO command))
```

This error indicates the mailbox to which the mail is being sent to cannot be found locally in the SMTP Server. There may be several reasons for this condition. The first one is that the mailbox really doesn’t exist in the SMTP Server. The second one is the SMTP Server is actually an SMTP Proxy that doesn’t have any local mailboxes but its relay is configured to accept such mailboxes as its own. A third possibility is the mailbox didn’t contained the domain part, for example, the mailbox was “user” instead of “user@domain”. In these cases most of mail servers will try to auto complete the domain part with its own name. In such cases when the mailbox is trying to be delivered, the SMTP Proxy will try to look in its own user’s table instead to deliver the mail to the internal server.

```
dsn=4.0.0, host:[IP] refused to talk to me: 421 4.0.0 Intrusion
prevention active for [IP])
```

This is an example of an IPS rule being triggered because of file attached to the original mail that contains malware or an exploit.

```
dsn=4.0.0, 450 <user@domain.com>… User information temporarily
unavailable (in reply to RCPT TO command))
```

This error indicates that probably the SMTP Server is using an integrated LDAP service that is not available at the time. In such conditions the SMTP Server will not be able to accept new mails because it is not able to validate if the recipient's mailbox exists. This condition can only be remediated on the SMTP Server side.

```
dsn=4.0.0, 451-Your mail was previously greylisted and the timeout has
not yet expired. 451-You should wait another 298

dsn=4.0.0, 451 Still greylisted - please try again in five minutes. (in
reply to RCPT TO command))
```

This code indicates the SMTP Client' IP address is being blocked temporarily by a black list on the SMTP Server Side and it hasn't reached its time limit to allow new mails to be received again. SMTP Client should wait until its IP address is taken out from the black list.

3.2 Address Status

X.1.0 Other address status

```
          Something about the address specified in the message caused
          this DSN.
Examples:
C> MAIL FROM:<user@domain.com>
S> 250 2.1.0 user@domain.com...Sender OK
```

This example shows the mailbox has been successfully accepted because it complies with the appropriate syntax.

X.1.1 Bad destination mailbox address

```
          The mailbox specified in the address does not exist.  For
          Internet mail names, this means the address portion to the left
          of the "@" sign is invalid.  This code is only useful for
          permanent failures.

Examples:

550 5.1.1 <asdf>: Recipient address rejected: User unknown in local
recipient table

550 5.1.1 unknown or illegal alias:user@domain.com (in reply to RCPT TO
command)
```

Both examples show a condition where the mailbox or alias do not exist (or were deleted) from the SMTP Server. This is a typical Postfix error description. It could also happen that the mailbox does actually exist but the alias is not yet updated on the Postfix alias table.

X.1.2 Bad destination system address

The destination system specified in the address does not exist or is incapable of accepting mail. For Internet mail names, this means the address portion to the right of the "@" is invalid for mail. This code is only useful for permanent failures.

X.1.3 Bad destination mailbox address syntax

The destination address was syntactically invalid. This can apply to any field in the address. This code is only useful for permanent failures.

```
Examples:
C> RCPT TO:<asdf@#(.com>
S> 501 5.1.3 Bad recipient address syntax
```

This condition can only be resolved by the original sender. The SMTP Server cannot mitigate this condition as the mailbox cannot be appropriately validated as a valid mailbox syntax.

X.1.4 Destination mailbox address ambiguous

The mailbox address as specified matches one or more recipients on the destination system. This may result if a heuristic address mapping algorithm is used to map the specified address to a local mailbox name.

X.1.5 Destination address valid

This mailbox address as specified was valid. This status code should be used for positive delivery reports.

```
Examples:
C> RCPT TO:<user@domain.com>
S> 250 2.1.5 user@domain.com
```

This example shows a condition where the SMTP Server has accepted the recipient’s mailbox as valid.

X.1.6 Destination mailbox has moved, No forwarding address

The mailbox address provided was at one time valid, but mail is no longer being accepted for that address. This code is only useful for permanent failures.

X.1.7 Bad sender's mailbox address syntax

```
         The sender's address was syntactically invalid.  This can apply
         to any field in the address.

Examples:
C> MAIL FROM:<kd#”.>
S> 501 5.1.7 Bad sender address syntax
```

This example shows a condition where the SMTP Server rejects a mailbox because it doesn’t have the right syntax.

```
      X.1.8   Bad sender's system address

         The sender's system specified in the address does not exist or
         is incapable of accepting return mail.  For domain names, this
         means the address portion to the right of the "@" is invalid
         for mail.
Examples:
C> MAIL FROM:<user@kdislskdisl.com>
S> 450 4.1.8 <user@kdislskdisl.com>: Sender address rejected: Domain not
found
```

This kind of errors occur when the SMTP Server has a PTR validation filter enabled to make sure the sender’s domain actually exists. This type of filter is not recommended because not all MTA servers have their PTR records registered on public DNS server. This can also result in valid mails being rejected.

3.3 Mailbox Status

X.2.0   Other or undefined mailbox status

```
         The mailbox exists, but something about the destination mailbox
         has caused the sending of this DSN.

Examples:

dsn=4.2.0, 450 4.2.0 <user@domain.com>: Recipient address rejected:
Greylisted (in reply to RCPT TO command))
```

In this example, the SMTP Client’s IP Address is in a black list so temporarily no new mails will be received.

```
      X.2.1   Mailbox disabled, not accepting messages

         The mailbox exists, but is not accepting messages.  This may be
         a permanent error if the mailbox will never be re-enabled or a
         transient error if the mailbox is only temporarily disabled.
```

Examples:

```
dsn=4.2.1, 451 4.2.1 mailbox temporarily disabled: user@domain.com (in
reply to RCPT TO command))

dsn=4.2.1, 451 4.2.1 Mailbox busy, try again later (in reply to RCPT TO
command))

550 5.2.1 user disabled; cannot receive new mail: user@domain.com (in
reply to RCPT TO command)
```

This code means the mailbox does exist but it not able to receive new mail anymore. There may be several causes for these, for example, the mailbox may be full and the mail administrator has temporarily or permanently disable it; the user may no longer work for the company and the mailbox is temporarily disabled until the separation process is finished. It could also happen the user has left the company and its mailbox has been disabled but it has never been deleted.

```
      X.2.2   Mailbox full

         The mailbox is full because the user has exceeded a per-mailbox
         administrative quota or physical capacity.  The general
         semantics implies that the recipient can delete messages to
         make more space available.  This code should be used as a
         persistent transient failure.
```

```
451 4.2.2 user over quota; cannot receive new mail: user@domain.com (in
reply to RCPT TO command)

dsn=4.2.2, 452 4.2.2 Recipient Unable to accept message – mailbox
full(host) (in reply to RCPT TO command))

dsn=4.2.2, 450 4.2.2 <user@domain.com>… Account user@domain.com has
exceeded storage allocation. Please try again later. (in reply to RCPT
```

This error is very straight forward and tells the user’s mailbox is full. The only way to resolve this if for the mail administrator to increase the mailbox storage quota or for the user to free some space.

```
      X.2.3   Message length exceeds administrative limit

         A per-mailbox administrative message length limit has been
         exceeded.  This status code should be used when the per-mailbox
         message length limit is less than the general system limit.
         This code should be used as a permanent failure.


      X.2.4   Mailing list expansion problem

         The mailbox is a mailing list address and the mailing list was
         unable to be expanded.  This code may represent a permanent
```

failure or a persistent transient failure.

3.4 Mail system status

X.3.0 Other or undefined mail system status

The destination system exists and normally accepts mail, but something about the system has caused the generation of this DSN.

Examples:

```
dsn=4.3.0, 451 4.3.0 <user@domain.com>: Temporary lookup failure (in
reply to RCPT TO command))

dsn=4.3.0, 451 4.3.0 Message temporarily deferred. Please try again
later. (in reply to RCPT TO command))
```

These codes indicate a problem with the mailbox resolution service, probably a an LDAP service is not available at the time to validate the existence of the recipient’s mailbox.

X.3.1 Mail system full

Mail system storage has been exceeded. The general semantics imply that the individual recipient may not be able to delete material to make room for additional messages. This is useful only as a persistent transient error.

Examples:

```
dsn=4.3.1, 452 4.3.1 Insufficient system storage (in reply to MAIL FROM
command))
```

This error means there is no more free space in the mail server to accept new mail, in this case all mailboxes are affected.

X.3.2 System not accepting network messages

The host on which the mailbox is resident is not accepting messages. Examples of such conditions include an immanent shutdown, excessive load, or system maintenance. This is useful for both permanent and persistent transient errors.

Examples:

```
dsn=4.3.2, host [IP] refused to talk to me: 421 4.3.2 Too many open
connections.)

dsn=4.3.2, to talk to me: 421 4.3.2 rejected: Too much connections from
xxx.xxx.xxx[XX.XX.XX.XX])
```

```
dsn=4.3.2, 452 4.3.2 Connection rate limit exceeded. (in reply to MAIL
FROM command))

dsn=4.3.2, 451 4.3.2 Please try again later (in reply to RCPT TO
command))

dsn=4.3.2, 451 4.3.2 Please try again later (in reply to MAIL FROM
command))

dsn=4.3.2, 451 4.3.2 Greylisting is in effect (in reply to end of DATA
command))
```

These error are generated when the SMTP Client is sending too many connections to the SMTP Server causing a temporary blocking of its IP address. Usually there are no action to implement in order to correct this condition by the SMTP Client, after some time the IP may be taken out the blocking list but in case this doesn't happen, the SMTP Client administrator should directly contact the SMTP Server admin to manually take out its IP from the black list.

X.3.3 System not capable of selected features

```
   Selected features specified for the message are not supported
   by the destination system.  This can occur in gateways when
   features from one domain cannot be mapped onto the supported
   feature in another.
```

X.3.4 Message too big for system

```
   The message is larger than per-message size limit.  This limit
   may either be for physical or administrative reasons.  This is
   useful only as a permanent error.
```

Examples:

```
C> MAIL FROM:user@domain.com SIZE=9999999999
S> 552 5.3.4 Message size exceeds file system imposed limit
```

The recipient cannot implement any actions to correct this condition, the sender in this case should consider the size limitation on the SMTP Server to correct the condition.

X.3.5 System incorrectly configured

```
   The system is not configured in a manner that will permit it to
   accept this message.
```

3.5 Network and Routing Status

X.4.0 Other or undefined network or routing status

```
Something went wrong with the networking, but it is not clear
what the problem is, or the problem cannot be well expressed
with any of the other provided detail codes.
```

X.4.1   No answer from host

```
The outbound connection attempt was not answered, because
either the remote system was busy, or was unable to take a
call.  This is useful only as a persistent transient error.
```

Examples:

```
dsn=4.4.1, connect to host[IP]: Connection refused
dsn=4.4.1, connect to host[IP: Connection timed out
dsn=4.4.1, Network is unreachable
```

Even when this error denotes a network problem, it is usually received when there is no service listening on port 25. When receiving such errors, the SMTP Client admin should first validate if the domain is actually still receiving mail and if it does, validate if it has a corresponding MX record associated, if it doesn’t exist then the IP address where the SMTP Client is trying to deliver mail corresponds to an A record which is generally used as a Web server, this is the reason for the connection being rejected all the time. Notice that the error reported here is a 4XX type, this is because the domain does exist given the fact that it has an A record published, otherwise the error would be a 5XX type. The “Network is unreachable” is usually caused by physical problems may it be a network card, switch, firewall or any other device that is temporarily unavailable.

X.4.2   Bad connection

```
The outbound connection was established, but was unable to
complete the message transaction, either because of time-out,
or inadequate connection quality.  This is useful only as a
persistent transient error.
```

Examples:

```
4.4.2, status=deferred (lost connection with xx.xx.xx.xx[xx.xx.xx.xx]
while sending end of data — message may be sent more than once)

4.4.2, status=deferred (lost connection with xx.xx.xx.xx[xx.xx.xx.xx]
while sending message body)

4.4.2, status=deferred (delivery temporarily suspended: lost connection
with xx.xx.xx.xx[xx.xx.xx.xx] while sending end of data — message may be
sent more than once)

dsn=4.4.2, timed out while receiving the initial server greeting)

dsn=4.4.2, timed out while sending DATA command)

dsn=4.4.2, timed out while performing the HELO handshake)
```

```
dsn=4.4.2, lost connection with host[IP] while performing the HELO
handshake)

dsn=4.4.2, lost connection with host[IP] while sending RCPT TO)
```

This code implies a mail was being transmitted while the network connection suddenly failed. In this case both SMTP Client and Server should wait for a time-out after which they should drop the connection. The SMTP Client should assume this code and queue the mail for a later delivery. The SMTP Server simply drops the connection.

For a further analysis about this code you can read our following posts:

[SPANISH] Análisis de pérdida de conexión en IMSVA (dsn=4.4.2)

Postfix Lost connection analysis | 421 4.4.2 host Error: timeout exceeded

X.4.3   Directory server failure

```
The network system was unable to forward the message, because a
directory server was unavailable.  This is useful only as a
persistent transient error.
The inability to connect to an Internet DNS server is one
example of the directory server failure error.
```

Examples:

```
4.4.3, status=deferred (Host or domain name not found. Name service error
for name=domain.com type=MX: Host not found, try again)
```

To correct this condition, the SMTP Client should check its DNS server or use a public one like 8.8.8.8.

X.4.4   Unable to route

```
The mail system was unable to determine the next hop for the
message because the necessary routing information was
unavailable from the directory server.  This is useful for both
permanent and persistent transient errors.
A DNS lookup returning only an SOA (Start of Administration)
record for a domain name is one example of the unable to route
error.
```

Ejemplo:

```
5.4.4, status=bounced (Host or domain name not found. Name service error
for name=domain.com type=AAAA: Host not found)
```

This condition generally falls under the responsibility of the sender and may be caused by the sender wrongly typing the domain name part of the recipient’s mailbox.

X.4.5 Mail system congestion

The mail system was unable to deliver the message because the mail system was congested. This is useful only as a persistent transient error.

Examples:

```
dsn=4.4.5, host[IP] refused to talk to me: 421 4.4.5 Directory harvest
attack detected)
```

In this case, the SMTP Server has received several mails to nonexistent mailboxes and has temporarily blocked the SMTP Client IP address assuming a Directory Harvest Attack (DHA) may be running against it.

```
dsn=4.4.5, 452 4.4.5 Insufficient disk space; try again later (in reply
to MAIL FROM command))
```

In this example, the SMTP Server has ran out of free space to process new mails.

```
dsn=4.4.5, to talk to me: 421 4.4.5 Too many SMTP connections from this
host)
```

In this example, the SMTP Server has received too many new connections from the SMTP Client IP address and has decided to temporarily block it to preserve some free open connections for other SMTP Clients.

X.4.7 Delivery time expired

The message was considered too old by the rejecting system, either because it remained on that host too long or because the time-to-live value specified by the sender of the message was exceeded. If possible, the code for the actual problem found when delivery was attempted should be returned rather than this code.

Examples:

```
BA13F20ABD: from=<user@domain.com>, status=expired, returned to sender
```

In this example taken out from a Postfix maillog, a 5.4.7 error code has been assumed to record this line. This code, when needed, should be a 5XX type because the mail won’t get re-queued again.

3.6 Mail Delivery Protocol Status

X.5.0 Other or undefined protocol status

Something was wrong with the protocol necessary to deliver the

```
         message to the next hop and the problem cannot be well
         expressed with any of the other provided detail codes.
```

Ejemplo:

```
dsn=4.5.0, 450 4.5.0 <user@domain.com>... Account user@domain.com is
temporarily unavailable. Please try again later. (in reply to RCPT TO
command)
```

This error may be received because the SMTP Client IP address is being blocked, or because it uses an LDAP service to validate the existence of the mailbox but it is not available at the time.

X.5.1 Invalid command

```
         A mail transaction protocol command was issued which was either
         out of sequence or unsupported.  This is useful only as a
         permanent error.
```

Examples:

```
503 5.5.1 Error: send HELO/EHLO first
```

This problem can only be resolved on the SMTP Client by issuing the HELO/EHLO command first.

```
dsn=4.5.1, 451 4.5.1 Mailbox full (in reply to end of DATA command))
```

This is a typical error of implementation of this code since it is not related to this DSN description and also there is already a DSN for full mailbox specific issues.

X.5.2 Syntax error

```
         A mail transaction protocol command was issued which could not
         be interpreted, either because the syntax was wrong or the
         command is unrecognized.  This is useful only as a permanent
         error.
```

Examples:

```
500 5.5.2 Error: bad syntax
502 5.5.2 Error: command not recognized
504 5.5.2 <..>: Helo command rejected: need fully-qualified hostname
504 5.5.2 <a>: Sender address rejected: need fully-qualified address
504 5.5.2 <a>: Recipient address rejected: need fully-qualified address
```

This problem must be corrected at the SMTP Client side by issuing the proper commands and syntax.

X.5.3 Too many recipients

More recipients were specified for the message than could have been delivered by the protocol. This error should normally result in the segmentation of the message into two, the remainder of the recipients to be delivered on a subsequent delivery attempt. It is included in this list in the event that such segmentation is not possible.

Examples:

```
451 4.5.3 Too many recipients specified. (in reply to RCPT TO command)
```

The consequence for this kind of errors is not that serious because of an implicit rule in RFC 2821 that suggest that whenever this situation occurs, the SMTP Server should accept the mail for the already accepted recipients. For the rejected ones, the mail should remain queued in the SMTP Client for a later retry only for the remaining recipients. Because of this, mail logs may show duplicate entries for the same mail. For example, in the Trend Micro IMSVA anti-spam solution, the Message Tracking logs will show duplicate or more entries for the same mail under this condition, the same applies for other solutions as well. If you have the need to impose a limit in the number of recipients, you should set the SMTP limit as higher as possible to avoid the duplicate entries and then configure a number of recipients policy.

X.5.4 Invalid command arguments

A valid mail transaction protocol command was issued with invalid arguments, either because the arguments were out of range or represented unrecognized features. This is useful only as a permanent error.

Examples:

```
501 5.5.4 Syntax: MAIL FROM:<address>
501 5.5.4 Syntax: RSET
501 5.5.4 Bad message size syntax
555 5.5.4 Unsupported option: k
```

X.5.5 Wrong protocol version

A protocol version mis-match existed which could not be automatically resolved by the communicating parties.

3.7 Message Content or Message Media Status

X.6.0 Other or undefined media error

Something about the content of a message caused it to be considered undeliverable and the problem cannot be well expressed with any of the other provided detail codes.

X.6.1 Media not supported

The media of the message is not supported by either the delivery protocol or the next system in the forwarding path. This is useful only as a permanent error.

X.6.2 Conversion required and prohibited

The content of the message must be converted before it can be delivered and such conversion is not permitted. Such prohibitions may be the expression of the sender in the message itself or the policy of the sending host.

X.6.3 Conversion required but not supported

The message content must be converted in order to be forwarded but such conversion is not possible or is not practical by a host in the forwarding path. This condition may result when an ESMTP gateway supports 8bit transport but is not able to downgrade the message to 7 bit as required for the next hop.

X.6.4 Conversion with loss performed

This is a warning sent to the sender when message delivery was successfully but when the delivery required a conversion in which some data was lost. This may also be a permanent error if the sender has indicated that conversion with loss is prohibited for the message.

X.6.5 Conversion Failed

A conversion was required but was unsuccessful. This may be useful as a permanent or persistent temporary notification.

3.8 Security or Policy Status

X.7.0 Other or undefined security status

Something related to security caused the message to be returned, and the problem cannot be well expressed with any of the other provided detail codes. This status code may also be used when the **condition** cannot be further described because of security policies in force.

Examples:

```
421 4.7.0 mx.server.com Error: too many errors
```

Almost all Mail servers have an error amount limit, once exceeded you would see errors like this.

X.7.1 Delivery not authorized, message refused

The sender is not authorized to send to the destination. This can be the result of per-host or per-recipient filtering. This memo does not discuss the merits of any such filtering, but

```
         provides a mechanism to report such.  This is useful only as a
         permanent error.
```

Examples:

```
554 5.7.1 <user@domain.com>: Relay access denied.
```

This code is received when a mail is sent to a mailbox belonging to a domain that is not within the relay of the SMTP Server. This may be caused by a wrong relay configuration where the appropriate domain should be added to the relay list. If the relay is ok, then this code may indicate an attack where an attempt is being made to the SMTP Server trying to find if it is an Open Relay Server.

```
550 5.7.1 <asdf@domain.com>: Sender address rejected: Service
unavailable; SPF check failed and transaction closed due to the
organization's policy.
```

This error occurs whenever the sender's mailbox or the domain used in the HELO/EHLO command is not explicitly authorized to be used by the sending IP address. This behavior is typical in servers validating incoming SMTP Connections with the Sender Policy Framework (SPF) Protocol.

```
dsn=4.7.1, 450 4.7.1 You've exceeded your sending limit to this domain.
(in reply to end of DATA command))
```

This example shows an SMTP Server configured to receive only certain amount of mails from the same IP address in the same SMTP session or in a given time window; whenever either of both is reached, the IP is automatically blocked. To mitigate this condition, the SMTP Client should limit the amount of mails sent to a given domain in both, the same session or in a certain time window.

```
dsn=4.7.1, 450 4.7.1 <host.domain.com>: Helo command rejected: Host not
found (in reply to RCPT TO command))
dsn=4.7.1, 450 4.7.1 Client host rejected: cannot find your hostname,
[IP] (in reply to RCPT TO command))
```

This is error is received when the SMTP Client uses its own FQDN as the value for the HELO/EHLO command but it doesn't have an associated A record in its public DNS server. There are some servers that will try to validate if the argument of EHLO does exist before accepting new mail, they do this by requesting the A record for the FQDN presented as argument for the EHLO command. The problem here is that not all mail servers have an associated A record to its FQDN. For example, let's suppose the mail server for the DOMAIN.COM domain has this FQDN antispam.localdomain. When this kind of validation is performed, the SMTP Server will try to resolve the A record for antispam.localdomain, which obviously will fail because this name doesn't have an associated public A record. This condition can be mitigated or resolved by renaming the server to an FQDN that has an A record, or creating the appropriate A record in the public DNS or instructing the mail server to use only the domain name as argument of the EHLO command (sending

DOMAIN.COM instead of antispam.localdomain) or by removing this kind of verification on the SMTP Server.

```
dsn=4.7.1, 450 4.7.1 <user@domain.com>: Relay access unavailable. (in
reply to RCPT TO command))
```

This is a very common error when a company hosts its mail service with an online host but the service is no longer available, may it be because the company didn’t pay the corresponding fee, it didn’t renew the service but the host is still keeping its records online or it has simply changed from one mail host to another and the DNS records change is being made at the moment. If the MX record is correct and the company is still up, this error may occur because the SMTP Client is using either its DNS cache or a Smart host to send mail directly to that domain using a specific address without requesting the MX record from public DNS servers, in such cases just deleting the smart host to force the MX record query will solve the problem. For the other scenarios no action can be made on the SMTP Client side as the domain cannot be relayed.

```
dsn=4.7.1, 450 4.7.1 <user@domain.com>: Recipient address rejected:
ERROR-GL100 System busy, please try again later. (in reply to RCPT TO
command)
dsn=4.7.1, 451 4.7.1 Service unavailable - try again later (in reply to
DATA command))
dsn=4.7.1, 450 4.7.1 <user@domain.com>: Recipient address rejected:
Policy Rejection- Please try later. (in reply to RCPT TO command))
dsn=4.7.1, 450 4.7.1 <user@domain.com>: Recipient address rejected:
Service temporarily unavailable (in reply to RCPT TO command))
dsn=4.7.1, 451 4.7.1 Please try again later (in reply to DATA command))
```

This example shows a scenario where the SMTP Client IP address may be temporarily blocked or the SMTP Server is temporarily unable to process new mail. If the domain does exist and it has been long time without a successful mail deliver then the best reason would be for the SMTP Client IP being blocked by the SMTP Sender.

```
dsn=4.7.1, 451 4.7.1 Greylisting in action, please come back later (in
reply to RCPT TO command))
dsn=4.7.1, 450 4.7.1 <user@domain.com>: Recipient address rejected:
Greylisted (in reply to RCPT TO command))
dsn=4.7.1, 450 4.7.1 <xxx.xxx.xxx[XX.XX.XX.XX]>: Client host rejected:
Greylisted for 5 minutes (in reply to RCPT TO command))
```

Here the SMTP Client IP address is simply blocked.

X.7.2 Mailing list expansion prohibited

```
The sender is not authorized to send a message to the intended
mailing list.  This is useful only as a permanent error.
```

X.7.3 Security conversion required but not possible

A conversion from one secure messaging protocol to another was required for delivery and such conversion was not possible. This is useful only as a permanent error.

X.7.4 Security features not supported

A message contained security features such as secure authentication that could not be supported on the delivery protocol. This is useful only as a permanent error.

X.7.5 Cryptographic failure

A transport system otherwise authorized to validate or decrypt a message in transport was unable to do so because necessary information such as key was not available or such information was invalid.

X.7.6 Cryptographic algorithm not supported

A transport system otherwise authorized to validate or decrypt a message was unable to do so because the necessary algorithm was not supported.

X.7.7 Message integrity failure

A transport system otherwise authorized to validate a message was unable to do so because the message was corrupted or altered. This may be useful as a permanent, transient persistent, or successful delivery code.

Examples:

```
dsn=4.7.7, 451 4.7.7 Excessive userid unknowns from [IP] (in reply to
MAIL FROM command))
```

This example shows a scenario where an SMTP Server is receiving several mails for nonexistent mailboxes and assumes it is being under a possible attack, as a defensive measure it reacts by temporarily blocking the SMTP Client IP address. This situation may be avoided by dropping mails for which the SMTP Client knows the mailbox doesn’t exist.

By understanding the fundamentals by which the status / reply codes are defined, you’ll be in a better position to understand what the status of the transmission is by just looking at the numeric codes.

2.3. Native SMTP Vulnerabilities

Now that we have a wider vision about the conforming standards of electronic mail let's now study the security holes present in SMTP and how to handle them to avoid attacks into your Organization.

For a better understanding we'll classify them according to the mail transmission section where they can be found: Envelope, Headers, Body and Attachments.

2.3.1. Envelope Vulnerabilities

The SMTP structure makes the protocol vulnerable even from the very moment when the transmission is starting. The following sections will show the different kind of attacks that can be made at the Envelope level.

2.3.1.1. Initial connection and the HELO / EHLO Command

Section 3.1 of RFC 2821 establishes that when making the initial connection to an SMTP Server, this should respond with a 220 status code and identify itself in the greeting with its software and SMTP versions. For security reasons, it is allowed for an Organization not willing to expose this kind of information to avoid such kind of greetings.

It is also allowed for the initial connection to be rejected by the SMTP Server as long as the following conditions are met:

1. Instead of a 220 status code it should respond with a 554.
2. It should wait for the QUIT command and should respond to any other command with a response like "503 Bad sequence of commands.".

Any SMTP Server is free to reject incoming connections as long as there is a technical reason to do so, dropping connections without a reason is forbidden by the RFC. Some of the reasons for this scenario may be:

a) The server has reached the maximum number of incoming connections it can handle.
b) The server doesn't have enough free storage space to process new incoming mails.
c) The SMTP Client IP address is black-listed.
d) The SMTP Client IP address is in violation of internal security policies.

When the SMTP Client receives a 220 greeting, the next step is the Handshake between both MTA servers. Here, the client has the responsibility to send the HELO/EHLO command to identify itself with its FQDN or domain name. The problem with this rule is that as long as the syntax is correct, the SMTP Client is not forced to “tell the truth” about its identity.

The SMTP Server may use any verification mechanism to verify if the identity presented in the HELO/EHLO command really belongs to the IP address from which the transmission is being received. To mitigate these problems, the SMTP Server may request the PTR record for the domain from which the mail is being received from to make sure that name has an associated public IP address. It can also make use of alternative protocols like SPF and Sender-ID that pretend to identify if the connecting IP address has a valid domain and if it is actually authorized to use such domain. The problem here is that not all MTA servers have a fixed name or legitimate public DNS records, so there will always be a chance for such verifications to still fail.

If the SMTP Server accepts a connection with code 220, it will place itself in a situation where it is not allowed to close the connection until the SMTP Client finishes the mail transmission or until an inactivity time limit has been reached. This can be used by an attacker to generate dozens, hundreds or even thousands of null sessions, overwhelming the available free connections of the server and diminishing its capacity to receive new mails from other sources. In this case, an IP reputation list (either local or in the cloud) and the tuning for the timeouts periods are the best solution.

Because MX records are public, an attacker may easily choose any of the IP addresses of the available mails servers to directly send SPAM or start a DoS attack. The logic under this kind of attack is that some Organizations decide to only protect the MTA servers on higher priorities while leaving lower priorities with minimum or null security. The best practice when we face environments where more than one MX record must be published is not to leave any of such services without any security measures. If the Organization is not able to maintain the same security level on all servers a Risk Assessment Plan will help in providing the right justification for such decisions and the required level for each server.

2.3.1.2. MAIL

At this level there are several vulnerable points to attack either the SMTP Server of the final users. On of the most important was the syntax established by RFC 821 for this command:

```
RFC 821: MAIL <SP> FROM:<reverse-path> <CRLF>
```

Where <reverse-path> is the full route to reach the final recipient, including not only the mailbox name but the series of MTA servers the mail should be relayed trough. In the times when the protocol was defined this was not a problem but in the earlier beginnings of year 2000 this syntax was discouraged and forbidden because of the use spammers were giving to mail servers. For this reason, RFC 2821 and 5321 have now changed the syntax to:

```
RFC 2821/5321: MAIL FROM:<reverse-path> [SP <mail-parameters> ] <CRLF>
```

Where <reverse-path> includes only the recipient's mailbox, however the new syntax of <mail-parameters> includes new security holes that an attacker can take advantage of.

The following is a description of the vulnerabilities that may be found under this new definition of the MAIL command:

a) **MAIL FROM: <>.** RFC 2821-6.1 defines the mechanism by which a mail system can send notifications to users. The sender for such notifications (as they are being generated automatically by the MTA server) should be a null reverse-path represented by a "<>" syntax. Whenever an MTA receives such kind of mails it should assume there is no return path in case such mail cannot be delivered to the final recipient, this is to avoid a loop between MTA servers where each would respond with a null reverse-path to the Non-Delivery-Response (NDR) from the other MTA. This definition implies the Return-path header should be empty as well, so there is no way to identify the original mailbox that created the mail. Under the notification context this is not a problem since the intention of such mails is just to inform an event to original sender without waiting for a reply.

   An attacker can send several mails with a null Return-path hiding its own mailbox and there would be no way, based on this rule, to identify what the original sender's mailbox was, if it really did existed one. The same RFC forbids the blocking of mails that have a null sender because all the automatic SMTP notification system would be blocked, this makes it a little harder to mitigate this kind of attack. However, there is a simple SMTP rule that can be implemented to minimize this kind of attack, this rule is:

Any automatic mail notification with the intention to inform of a mail delivery failure can only contain one recipient, and this must be the one that appeared on the original Return-path header of the mail that caused the delivery failure, therefore, a mail with a null sender can only have ONE recipient, any mail with a null sender and more than one recipient is in violation of the SMTP Standards and may be blocked or deleted without generating a new NDR.

b) **MAIL FROM:<fake_mailbox>.** The protocol establishes only the syntax to be used for the MAIL command, but it doesn't force a verification to confirm if the argument is a legitimate mailbox or if even belongs to the domain it is supposed to be coming from. This allows the reception of mails where an SMTP Client identifies itself as EHLO xxx.com and then sends a MAIL command with another domain like user@yyy.com. According to the protocol this is not a violation and may be exploited by an attacker to send mails from a mail server using any kind of mailboxes he/she wants.

   A variant from this vulnerability is to send mails with a sender in the Envelope that is different from the sender appearing in the *From* Header. This security hole is implicit in the SMTP definition and is very difficult to detect because the syntax for the *From* Header is different than that of the MAIL command, so no unique comparison is always possible to detect such deviations.

   One method used to try to verify the existence of the sender's mailbox is by sending a mail probe to the SMTP Client IP address before accepting the RCPT command, if the sender is accepted then one may assume the user does exist in the original domain, however this will only work if the MTA checks its recipients against an LDAP service or an updated list of local users, otherwise it would accept such probe just to later realize the mailbox didn't existed. In case the probe is accepted the SMTP Server may continue to receive the rest of the mail, otherwise it would send a 4XX or 5XX code to end the transmission.

c) **SIZE.** When the SMTP Sever requires a restriction on the mail size it is able to handle, this definition should appear in the Handshake response to indicate the maximum size it is willing to accept. If no such limit is presented, it can be assumed such MTA has no limit. An attacker can verify this kind of condition to send larger mails to overwhelm the Organization's bandwidth and mailbox storage capacity. For example, if an SMTP Client send a 30 MB mail with a MAIL FROM:

<user@domain.com> SIZE=1024, an SMTP Server with a 10MB (10485760) limit and will later realize the mail was in violation of its size quota limit. Also notice that in an SMTP Client doesn't send the SIZE parameter, this limitation is useless. To avoid this kind of attacks it is advisable to configure size based policies within your internal server or at the anti-spam solution, this way, the total SIZE will be calculated on the actual mail stopping it before it reaches the recipient's mailbox.

2.3.1.3. RCPT

This command may be used in attacks using the DSN-RCPT-NOTIFY method by issuing a command like RCPT TO:<user@domain> NOTIFY=SUCCESS and RCPT TO:<user@domain> NOTIFY=FAILURE. As a result of such notification, an attacker may get aware if any given mailbox is valid in order to direct SPAM campaigns or other attacks to such addresses. Besides this, these methods force the SMTP Server to generate new mails to notify the attacker which means more resources are being used on the Organization's side.

A spoofing attack is also possible when the attacker sends mails to the right mailbox but the "To:" Header is different, creating a confusion to the user by not being sure if the mail was intended for him / her. We'll talk more about spoofing in the Header Vulnerabilities.

2.3.2. Header Vulnerabilities

These attacks exists mainly because SMTP is designed to be compliant with a certain syntax but it doesn't give any mechanisms to the SMTP Server on how to protect itself from its contents. The following sections will show some of the attacks your Organization may face at this level.

2.3.2.1. Automatic Notifications

The MDN header Disposition-Notification-To:<user@domain> may be used to learn if a certain SPAM campaign or the delivery of malware files has been successful and at what rate. This header will try to get the user's confirmation to send an automatic notification to the sender telling the mail was read. With this information, an SMTP Client may measure the success of this type of campaigns. The Return-Receipt-To:<user@domain> header may be used instead to try to bypass the user acknowledge to send the notification, however, this will not necessarily be implemented on al Mail User Clients. The best recommendation in these cases is to educate users in not responding to suspicious mails and be cautions in opening such kind of suspicious messages.

2.3.2.2. Content mismatch

The MIME format allows the transmission of mails with alternate body sections, this is for the User Agents to easily decide which section should be presented to the final user, however, this characteristic may be used by an attacker to send mails with normal information in one section and a malicious one or SPAM in another. An anti-spam engine should be able to verify the body content on any of the MIME compliant mail no matter what the final format to be shown to the user is. Remember that according to MIME rules, the last format section should be the preferred one to be shown to the user unless it cannot process it. This may cause that any mail that at glance seems to be normal, may in fact include malicious content. Because of this, the best solution is an anti-spam solution that is able to verify SPAM and malicious content in any of the sections for an alternate MIME compliant message.

2.3.2.3. File formats mismatch

When sending attached files, the MIME format offers a special header which indicates the “media-type” to which the file belong to, however, this cannot guarantee the file doesn’t belong to any other format. An attacker can literally present any media-type he wants to obfuscate the fact that his file is really an executable for example. Because of this, it is advisable to check for the True-File-Type of attached files instead of using the MIME headers.

2.3.2.4. Spoofing

There is no official mechanism that allows for the recipient or sender identities verification. For this reason, any mail can be send using any mailbox address, even one that actually belongs to the Organization itself. By design, internal mail sent between the Organization’s users should be process by the internal mail servers without passing through any Gateway point, based on this, it wouldn’t be possible for a mail with an Organization sender mailbox address to be received form an external network like Internet, this scenario falls under the spoofing attack. Any mail that uses a legitimate domain name for which he/she is not the owner falls also under this category, for example a mail supposedly coming from @facebook, when actually the SMTP Client is not authorized to use such domain name. Unfortunately the SMTP protocol doesn’t include a mechanism to prevent such attacks. To mitigate these attacks it is advisable to use alternative protocols like SPF or Sender-ID. For internal spoofing attacks, rules can be configured to intercept mails that contain the Organization’s own domain name in both *From* and *To, Cc*, headers.

2.3.3. Body and Attachment Vulnerabilities

Because of the SMTP Protocol native definition, the body contents cannot be verified against content or file security policies. An attacker could use such limitations to send SPAM or introduce malware into the Organization if no proper security measures are in place to detect them. Attacks under this category are described in the following sections as:

2.3.3.1. SPAM

By definition, SMTP doesn't impose any restriction on the mail content context. This may cause the delivery of unwanted mails into the Organization even when they may come from legitimate sources. SPAM, as early described on Chapter 1, may make use of several techniques to achieve its goal of being delivered to the final user. The most advisable solution for this problem is an anti-spam solution that may incorporate one or more of the following features: heuristic detection, patterns, word and sentence correlation, image detection and IP reputation verification among others.

2.3.3.2. Content

By design, the SMTP protocol doesn't allow for a mail content to be matched against ongoing security policies in the Organization. Content attacks may fall under several sub-categories like pornography, offensive and racial language, data leakage, embedded malicious URL's, etc. To avoid such attacks, is advisable to use a solution capable of content inspection inside the several sections of and URL reputation inspection before the mail is delivered to the final recipient.

2.3.3.3. Malware

The MIME Format allows any MTA to send and receive any kind of file formats. Whether such files are legitimate or not doesn't fall under the SMTP scope. There is a need to inspect the file contents and format to decide whether such files should be delivered to the final recipient. Even when MIME allows any MTA server to know what the attached file format is, the same structure may be used to hide the real file format as described in previous sections. The most secure way to know what the format of any given file is, is to extract the True-File-Type of the file directly from the file itself on not to relay on the

MIME headers or the file extension. A proper attachment and malware scanning solution is advisable to mitigate these attacks.

2.4. Summary

This Chapter has covered the complete structure of any electronic mail (e-mail) either in plain text or MIME format. We analyzed each of the points from the corresponding RFC's that indicate the rules and syntax to use each of the fields, commands, headers and sections that must be incorporated in the message transmission in order for it to be considered SMTP compliant and legitimate.

By completing these sections you should now be able to fully understand how an e-mail is formed and how these structures can be used by attackers to deliver malicious content into your Organizations along with the measures you should be aware of to prevent them. Each portion of a mail along with the methods used in its transmission obey to well defined structures, any deviation may have consequences on the security or reputation of the Organization.

References

SMTP References

RFC 821. Simple Mail Transfer Protocol (Agosto 1982)

http://tools.ietf.org/html/rfc821

RFC 822. Standard for the format of ARPA Internet Text Messages (Agosto 1982)

http://tools.ietf.org/html/rfc822

RFC 2821. Simple Mail Transfer Protocol (Abril 2001)

http://tools.ietf.org/html/rfc2821

RFC 2822. Internet Message Format (Abril 2001)

http://tools.ietf.org/html/rfc2822

RFC 5321. Simple Mail Transfer Protocol (Octubre 2008)

http://tools.ietf.org/html/rfc5321

RFC 5322. Internet Message Format (Octubre 2008)

http://tools.ietf.org/html/rfc5322

RFC 2554. SMTP Service Extension for Authentication (Marzo 1999)

http://tools.ietf.org/html/rfc2554

RFC 5428. A Registry for SMTP Enhanced Mail System Status Codes

http://tools.ietf.org/html/rfc5248

RFC 3463. Enhanced Mail System Status Codes

http://tools.ietf.org/html/rfc3463

RFC 4409. Message Submission for Mail.

http://tools.ietf.org/html/rfc4409

RFC 4954. SMTP Service Extension for Authentication

http://tools.ietf.org/html/rfc4954 (Julio 2007)

RFC 4422. Simple Authentication and Security Layer (SASL)

http://tools.ietf.org/html/rfc4422

RFC 1870. SMTP Service Extension for Message Size Declaration

http://tools.ietf.org/html/rfc1870

RFC 3461. Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notification (DSNs)

http://tools.ietf.org/html/rfc3461

RFC 3798. Message Disposition Notification

http://tools.ietf.org/html/rfc3798

RFC 2119. Key words for use in RFCs to Indicate Requirement Levels

http://tools.ietf.org/html/rfc2119

RFC 3282. Content Language Headers

http://tools.ietf.org/html/rfc3282

RFC 2156. MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME

http://tools.ietf.org/html/rfc2156

MIME Format References

RFC 2045. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

http://tools.ietf.org/html/rfc2045

RFC 2046. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

http://tools.ietf.org/html/rfc2046

RFC 2047. Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text.

http://tools.ietf.org/html/rfc2047

RFC 2048. Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures.

http://tools.ietf.org/html/rfc2048

RFC 2049. Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples.

http://tools.ietf.org/html/rfc2049

DNS References

RFC 1034. Domain Names – Concepts and Facilities

http://tools.ietf.org/html/rfc1034

RFC 1035. Domain Names – Implementation and Specification

http://tools.ietf.org/html/rfc1035

Microsoft Exchange Server 2003 References

Exchange Server Message Security Guide

http://technet.microsoft.com/en-us/library/aa996417%28EXCHG.65%29.aspx

Microsoft Exchange Server 2007 References

Exchange 2007 Security Guide

http://technet.microsoft.com/en-us/library/bb691338%28EXCHG.80%29.aspx

Microsoft Exchange Server 2010 References

Exchange 2010 Security Guide

http://technet.microsoft.com/en-us/library/bb691338.aspx

Tools & Utilities References

Base64 Online Decode and Encode

http://www.motobit.com/util/base64-decoder-encoder.asp

EA DomainKeys/DKIM for IIS SMTP Service and Exchange Server

http://www.emailarchitect.net/webapp/downloads.asp

Putty

http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

Starwind

http://www.starwindsoftware.com/download-free-version