

## Chapter 23

1. A sender sends unformatted text. Show the MIME header.

**Answer (example MIME header for unformatted plain text):**

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

(If the text may contain non-ASCII characters use charset=UTF-8 and Content-Transfer-Encoding: 8bit or quoted-printable/base64 as needed.)

এই প্রশ্নে বলা হয়েছে—

একজন প্রেরক (sender) যদি **unformatted text** বা সাধারণ টেক্সট পাঠায়, তখন **MIME header** কেমন হবে, তা দেখাতে হবে।

---

### ▣ MIME Header কী?

**MIME (Multipurpose Internet Mail Extensions)** হলো এমন একটি স্ট্যান্ডার্ড যা ই-মেইলের মাধ্যমে বিভিন্ন ধরনের ডেটা (যেমন: টেক্সট, ছবি, অডিও ইত্যাদি) পাঠানোর সুযোগ দেয়। এর মাধ্যমে ই-মেইল প্রেরণের সময় বার্তার ধরন (content type), এনকোডিং পদ্ধতি ইত্যাদি নির্ধারণ করা হয়।

---

### ☒ উদাহরণ (Unformatted Plain Text-এর MIME Header)

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

---

### ❑ ব্যাখ্যা (বাংলায়)

1. **MIME-Version: 1.0**  
→ এটি MIME প্রোটোকলের সংস্করণ নির্দেশ করে। সাধারণত সব ইমেইল MIME 1.0 ব্যবহার করে।
2. **Content-Type: text/plain; charset=US-ASCII**  
→ এটি বলে দেয় বাতাতি “plain text” অর্থাৎ সাধারণ লেখা।
  - o charset=US-ASCII মানে কেবল ইংরেজি অক্ষর ব্যবহার হয়েছে।

- যদি বার্তায় বাংলা বা অন্য ভাষার অক্ষর থাকে, তবে charset=UTF-8 ব্যবহার করতে হবে।
3. **Content-Transfer-Encoding: 7bit**  
→ এটি বলে দেয় কীভাবে বার্তাটি এনকোড বা রূপান্তর করা হয়েছে।
- 7bit মানে কেবল ASCII অক্ষর (ইংরেজি) ব্যবহৃত হয়েছে।
  - যদি বার্তায় non-ASCII (যেমন বাংলা বা অন্য বিশেষ চিহ্ন) থাকে, তবে 8bit, quoted-printable, বা base64 এনকোডিং ব্যবহার করতে হবে।
- 

## ৭. সংক্ষেপে মনে রাখার মতো

বিষয়	কাজ
<b>MIME-Version</b>	MIME প্রোটোকলের সংস্করণ
<b>Content-Type</b>	বার্তার ধরন (যেমন plain text, HTML, image ইত্যাদি)
<b>charset</b>	কোন বর্ণমালা ব্যবহৃত হচ্ছে (ASCII, UTF-8 ইত্যাদি)
<b>Content-Transfer-Encoding</b>	ডেটা কীভাবে পাঠানো হচ্ছে (7bit, 8bit, base64 ইত্যাদি)

---

৫. যদি টেক্সটে বাংলা বা অন্য non-ASCII অক্ষর থাকে, তাহলে MIME হেডার হবে এমন:

```
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
```

এভাবে MIME header ই-মেইল ক্লায়েন্টকে জানায় কীভাবে বার্তার ডেটা পড়তে এবং প্রদর্শন করতে হবে।

## 2. A sender sends a JPEG message. Show the MIME header.

**Answer (example MIME header for a JPEG attachment):**

```
MIME-Version: 1.0
Content-Type: image/jpeg; name="photo.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="photo.jpg"
```

(Inline images can use Content-Disposition: inline and the name/filename fields are optional.)

---

**3. A non-ASCII message of 1,000 bytes is encoded using base64. How many bytes are in the encoded message? How many bytes are redundant? What is the ratio of redundant bytes to the total message?**

---

### প্রশ্ন:

A non-ASCII message of 1,000 bytes is encoded using base64.

How many bytes are in the encoded message?

How many bytes are redundant?

What is the ratio of redundant bytes to the total message?

---

#### ◦ ধাপ ১: Base64 কী করে?

Base64 encoding এভাবে কাজ করে —

ক্ষেত্রফলে 3 byte ডেটা → 4 byte এ রূপান্তরিত হয়।

মানে, 3 byte মূল ডেটা পাঠাতে 4 byte লাগে।

তাহলে কিছু জায়গা “extra” লাগে, যেটাকেই আমরা বলি **redundant bytes**।

---

#### ◦ ধাপ ২: মোট 1000 byte কে 3-এর দলে ভাগ করো

$$1000 \div 3 = 333.33\dots$$

তাই মোট লাগবে

ক্ষেত্রফলে  $\text{ceil}(1000 / 3) = 334$  গ্রন্থ

(শেষের গ্রন্থটা অসম্পূর্ণ হলে Base64 padding যোগ করে)

---

#### ◦ ধাপ ৩: প্রতিটি গ্রন্থ 4 byte দেয়

তাহলে মোট encoded ডেটা হবে

$$334 \times 4 = 1336 \text{ bytes}$$

ক্ষেত্রফলে Encoded size = 1336 bytes

---

## ◇ ধাপ ৪: অতিরিক্ত (redundant) bytes বের করো

মূল ডেটা ছিল 1000 bytes

কিন্তু Base64 করার পর হলো 1336 bytes

তাহলে অতিরিক্ত হয়েছে:

$$1336 - 1000 = 336 \text{ bytes}$$

☞ Redundant bytes = 336 bytes

---

## ◇ ধাপ ৫: Redundancy ratio (অনুপাত)

অনুপাত = (অতিরিক্ত bytes) / (মোট encoded bytes)

$$336 \div 1336 \approx 0.2515$$

অর্থাৎ 25.15%, প্রায় 25% জায়গা বাঢ়তি লাগে।

---

**4. A message of 1,000 bytes is encoded using quoted-printable. The message consists of 90 percent ASCII and 10 percent non-ASCII characters. How many bytes are in the encoded message? How many bytes are redundant? What is the ratio of redundant bytes to the total message?**

**Answer (calculation):**

- ASCII bytes (pass-through) =  $0.90 \times 1000 = 900$  bytes (remain 1:1).
  - Non-ASCII bytes =  $0.10 \times 1000 = 100$  bytes. Each non-ASCII byte is encoded as =xx (3 bytes) in quoted-printable.
  - Encoded size =  $900 + (100 \times 3) = 900 + 300 = 1200$  bytes.
  - Redundant bytes =  $1200 - 1000 = 200$  bytes.
  - Redundancy ratio =  $200 / 1200 = 1/6 \approx 0.166666\dots \rightarrow 16.67\%$ .
- 

**5. Compare the results of Exercises 3 and 4. How much is the efficiency improved if the message is a combination of ASCII and non-ASCII characters?**

### Answer (comparison):

- Base64 encoded size: **1336 bytes** (redundancy  $\approx 25.15\%$ ).
- Quoted-printable encoded size: **1200 bytes** (redundancy **16.67%**).
- **Absolute saving:**  $1336 - 1200 = 136$  bytes saved by quoted-printable for this 90% ASCII message.
- **Relative reduction (compared to base64):**  $136 / 1336 \approx 0.1018 \rightarrow \approx 10.18\%$  smaller than base64.

**Conclusion:** For a message mostly ASCII (90% ASCII here), quoted-printable is more efficient than base64 (saves ~136 bytes, ~10% reduction). Base64 is better suited when the data is mostly binary/non-text (no readable ASCII).

---

### 6. Encode the following message in base64: (EXAM)

01010111 00001111 11110000 10101111 01110001 01010100

### Answer (step & result):

---

একদম ভালো প্রশ্ন — চল একবার যাচাই করে দেখি, কারণ Base64-এ 6-bit ভাগের সময় ১-টা ভুল হলে পুরো ফলাফল পাল্টে যেতে পারে।

আমাদের মূল ডেটা:

01010111 00001111 11110000 10101111 01110001 01010100

---

### ① সব 8-bit একত্রে লিখে দেই

0101011100001111111000010101110111000101010100

মোট 8<sup>8</sup> bit

---

### ② 6-bit করে ভাগ করি

010101 110000 111111 110000 101011 110111 000101 010100

দেখো—এখানে প্রতিটি গ্রুপ 6-bit করে নিখুঁত হয়েছে (কোনও padding দরকার হয়নি)।

---

### ৩ প্রতিটা 6-bit → Decimal

#### **Bits    Decimal**

010101 21

110000 48

111111 63

110000 48

101011 43

110111 55

000101 5

010100 20

---

### ৪ Base64 table অনুযায়ী character mapping

#### **Decimal Character**

21       V

48       w

63       /

48       w

43       r

55       3

5       F

20       U

---

#### Final Base64 encoding:

Vw/wr3FU

---

#### **7. Encode the following message in quoted-printable:**

01010111 00001111 11110000 10101111 01110001 01010100

---

#### Step 1: Binary message

01010111 00001111 11110000 10101111 01110001 01010100

মোট ৬ byte =

57 0F F0 AF 71 54 (in hexadecimal)

---

### ষষ্ঠি Step 2: কোনগুলো printable আর কোনগুলো না

Quoted-Printable **কেবল printable ASCII (33–126 decimal)** রাখে যেমন অক্ষর, সংখ্যা, কিছু symbol।

যেগুলো non-printable বা special, সেগুলো =xx ফরম্যাটে hex আকারে লেখা হয়।

Byte (hex)	Decimal	Printable?	QP form
57	87	<input checked="" type="checkbox"/> 'W'	W
0F	15	<b>X</b>	=0F
F0	240	<b>X</b>	=F0
AF	175	<b>X</b>	=AF
71	113	<input checked="" type="checkbox"/> 'q'	q
54	84	<input checked="" type="checkbox"/> 'T'	T

---

#### Final Quoted-Printable Encoding:

W=0F=F0=AFqT

---

### 8. Encode the following message in base64:

01010111 00001111 11110000 10101111 01110001

#### Answer (step & result):

চলো ধাপে ধাপে করি দ্বি

আমাদের বাইনারি মেসেজ:

01010111 00001111 11110000 10101111 01110001

---

## ঝঃ Step 1: সবগুলো bit একত্রে লিখি

010101110000111111100001010111101110001

মোট 80 bit আছে।

Base64 প্রতি 28 bit করে (৩ byte) এনকোড করে → এখানে আছে  $\frac{80}{28} = 2.8$  byte = ১টা সম্পূর্ণ 28-bit গ্রুপ + ১টা 16-bit বাকি থাকবে।

তাই শেষে padding লাগবে।

---

## ঝঃ Step 2: ৬-bit করে ভাগ

010101 110000 111111 110000 101011 110111 000100

শেষ গ্রুপে মাত্র 8 bit আছে (0100) → padding দিয়ে 010000 করতে হবে (শেষে == আসবে)।

তাহলে পুরোটা হবে:

010101 110000 111111 110000 101011 110111 000100 000000

---

## ঝঃ Step 3: প্রতিটি ৬-bit কে decimal-এ রূপান্তর

### Bits      Decimal

010101 21

110000 48

111111 63

110000 48

101011 43

110111 55

000100 4

000000 0

---

## ঝঃ Step 4: Base64 character mapping

### Decimal Char

21      V

48      w

63      /

### Decimal Char

48	w
43	r
55	3
4	E
0	A

শেষের অসম্পূর্ণ গ্রহণ (padding) → = যোগ হবে।

---

### Final Base64 encoding:

Vw/wr3EA==

---

### 9. Encode the following message in quoted-printable:

01010111 00001111 11110000 10101111 01110001

### Answer (step & result):

- Bytes: w (0x57), 0x0F, 0xF0, 0xAF, q (0x71).
  - Quoted-printable encodes non-printables: =0F, =F0, =AF. Printable w and q remain.
  - Result:** W=0F=F0=AFq
- 

### 10. Are the HELO and MAIL FROM commands both necessary? Why or why not?

চলুন বিষয়টি বিস্তারিতভাবে ব্যাখ্যা করি ঙু

---

### প্রশ্ন:

### Are the HELO and MAIL FROM commands both necessary? Why or why not?

(HELO এবং MAIL FROM — দুটি কমান্ড কি দুটোই প্রয়োজনীয়? কেন বা কেন নয়?)

---

## ✉ SMTP প্রেক্ষাপট

এই দুটি কমান্ড SMTP (Simple Mail Transfer Protocol)-এর অংশ।

SMTP হচ্ছে ইমেইল পাঠানোর জন্য ব্যবহৃত প্রোটোকল, যা প্রেরক মেইল সার্ভার থেকে প্রাপক মেইল সার্ভারে বার্তা পাঠায়।

### ✉ HELO (বা EHLO) কমান্ড কী করে?

HELO বা EHLO (Extended HELO) হলো SMTP সেশনের প্রথম ধাপ —

যখন একটি মেইল সার্ভার (client) অন্য একটি সার্ভারের (server) সাথে সংযোগ স্থাপন করে, তখন নিজেকে পরিচয় করানোর জন্য HELO পাঠায়।

#### উদাহরণ:

HELO mail.example.com

#### উদ্দেশ্য:

- ক্লায়েন্ট সার্ভারকে জানায় সে কোন ডোমেইন থেকে সংযোগ করছে।
- সার্ভার সেই অনুযায়ী রেসপন্স দেয় (যেমন: "250 Hello mail.example.com").

👉 অর্থাৎ: HELO হলো “পরিচয় দেওয়া” — SMTP সেশন শুরু করার জন্য অপরিহার্য ধাপ।

### ✉ MAIL FROM কমান্ড কী করে?

এরপর আসে MAIL FROM কমান্ড, যা প্রেরকের ইমেইল ঠিকানা নির্ধারণ করে।

#### উদাহরণ:

MAIL FROM:<sender@example.com>

#### উদ্দেশ্য:

- সার্ভারকে জানানো হয় কে বার্তাটি পাঠাচ্ছে।
- এটি প্রাপকের সার্ভারকে bounce বা error message পাঠানোর ঠিকানাও দেয়, যদি মেইল পৌঁছাতে ব্যর্থ হয়।

👉 অর্থাৎ: MAIL FROM হলো “বার্তার প্রেরক নির্ধারণ” — মেইল পাঠানোর জন্য অপরিহার্য ধাপ।

---

## ❖ দুটোই কেন দরকার?

ক্ষমতা	কাজ	প্রয়োজনীয়তা
HELO / EHLO	সার্ভারের সাথে পরিচয় স্থাপন করে ও সেশন শুরু করে	SMTP সংযোগ স্থাপনের জন্য দরকার
MAIL FROM	প্রেরকের ঠিকানা নির্ধারণ করে	মেইল ডেলিভারি ও বাউন্স ম্যানেজমেন্টের জন্য দরকার

- HELO ছাড়া সার্ভার জানবে না কে সংযোগ করছে → যোগাযোগ শুরু হবে না।
- MAIL FROM ছাড়া সার্ভার জানবে না কে মেইল পাঠাচ্ছে → বার্তা প্রক্রিয়া করা সম্ভব নয়।

অর্থাৎ —

**দুটোই প্রয়োজনীয়**, কারণ HELO সেশন শুরু করে, আর MAIL FROM প্রকৃত ইমেইল প্রেরণ প্রক্রিয়া শুরু করে।

---

## ❖ সংক্ষিপ্ত উত্তর:

**Yes, both HELO (or EHLO) and MAIL FROM are necessary.**

HELO identifies the client and establishes the SMTP session, while MAIL FROM specifies the sender's address to begin the mail transaction. Without either, the email transfer cannot proceed properly.

**11. In Figure 23.11 what is the difference between MAIL FROM in the envelope and the From in the header?**

চমৎকার প্রশ্ন  — এটি SMTP (Simple Mail Transfer Protocol) এবং ইমেইল ফরম্যাটের envelope vs header পার্থক্য বোঝার জন্য খুব গুরুত্বপূর্ণ।  
চলুন ধাপে ধাপে দেখি 

---

## ❖ ইমেইলের দুটি স্তর: Envelope এবং Header

একটি ইমেইল আসলে দুই স্তরে কাজ করে:

স্তর

কাজ

Envelope (খাম) SMTP প্রোটোকলের সময় ব্যবহৃত — মেইল সার্ভারের জন্য

Header (শিরোনাম) ইমেইল বার্তার অংশ — পাঠক (মানুষ) ও মেইল ক্লায়েন্টের জন্য

একটি বাস্তব চিঠির মতো কল্পনা করুন —

Envelope হলো ডাকখানার খাম (যাতে প্রেরক ও প্রাপকের ঠিকানা থাকে), আর Header হলো চিঠির উপরের অংশ যেখানে প্রেরকের নাম লেখা থাকে।

### ✉️ MAIL FROM (Envelope-এর মধ্যে)

MAIL FROM কমান্ডটি SMTP conversation-এর অংশ।

এটি ইমেইল প্রেরক সার্ভারের দৃষ্টিকোণ থেকে প্রেরকের ঠিকানা নির্ধারণ করে।

উদাহরণ:

MAIL FROM:<bounce@example.com>

👉 এটি সার্ভারকে বলে:

“যদি এই ইমেইল ডেলিভারিতে কোনো সমস্যা হয়, তাহলে এর মেসেজ বা বাউল মেইল এই ঠিকানায় পাঠাও।”

❖ গুরুত্বপূর্ণ:

MAIL FROM সাধারণত দেখা যায় না — এটি ব্যবহারকারীর ইনবক্সে প্রদর্শিত হয় না।

এটি শুধু মেইল সার্ভারগুলোর মধ্যে যোগাযোগের জন্য ব্যবহৃত হয়।

### ✉️ From (Header-এর মধ্যে)

From: ফিল্ড হলো ইমেইলের header-এর অংশ — এটি বার্তার ভেতরে লেখা থাকে।

উদাহরণ:

From: Alice Johnson <alice@example.com>

👉 এটি মানুষের জন্য দৃশ্যমান অংশ —

যখন আপনি ইনবক্সে “From” কলামে কারো নাম বা ইমেইল দেখেন, সেটি এই ফিল্ড থেকে আসে।

## ❖ গুরুত্বপূর্ণ:

- এটি ব্যবহারকারী যা লেখে, তাই দেখানো হয়।
- এটি SMTP ট্রান্সফার প্রক্রিয়ায় ব্যবহৃত হয় না।
- এটি প্রদর্শন বা পরিচিতির উদ্দেশ্যে।

## ঞ্চ মূল পার্থক্য

বিষয়	MAIL FROM (Envelope)	From: (Header)
ব্যবহারের জায়গা	SMTP প্রোটোকলে (সার্ভার লেভেলে)	ইমেইল মেসেজের হেডারে (বার্তার ভেতরে)
দৃশ্যমানতা	ব্যবহারকারীর কাছে দেখা যায় না	ইনবক্সে দেখা যায়
উদ্দেশ্য	বাটন/এরর মেইল পাঠানোর ঠিকানা	বার্তা প্রেরকের নাম বা পরিচয়
নিয়ন্ত্রণ করে কে মেইল সার্ভার		ইমেইল লেখক বা ক্লায়েন্ট
উদাহরণ	MAIL FROM:<mailer-daemon@example.com>	From: Alice <alice@example.com>

## ক্ষি সংক্ষেপে বলা যায়:

**MAIL FROM** হলো “Envelope Sender” — সার্ভার লেভেলে ব্যবহৃত,  
**From:** হলো “Header Sender” — মানুষের জন্য প্রদর্শিত।

- MAIL FROM ঠিকানায় বাটন মেইল যায়।
- From ঠিকানাটি ইনবক্সে “Who sent this email?” হিসেবে দেখা যায়।

## উদাহরণসহ চিত্র (সংক্ষেপে):

SMTP Layer (Envelope)

-----  
MAIL FROM:<bounce@example.com>  
RCPT TO:<bob@example.net>

Message Data (Header + Body)

-----  
From: Alice <alice@example.com>  
To: Bob <bob@example.net>  
Subject: Hello

...

ঞ্চ যখন এই ইমেইল পোঁচে যায়, Bob ইনবক্সে “From: Alice [alice@example.com](mailto:alice@example.com)” দেখবে, কিন্তু যদি মেইল ব্যর্থ হয়, bounce message যাবে [bounce@example.com](mailto:bounce@example.com) ঠিকানায়।

## 12. Why is a connection establishment for mail transfer needed if TCP has already

### established a connection?

যদিও TCP আগে থেকেই দুই সার্ভারের মধ্যে সংযোগ তৈরি করে, SMTP-এর নিজস্ব **application-level handshake** বা “connection establishment” প্রয়োজন হয় কারণ:

1. **TCP শুধু transport স্তরের সংযোগ দেয়**, SMTP কে জানাতে হয় “কে কার সাথে কথা বলছে!”  
অর্থাৎ — সার্ভারকে জানতে হয় প্রেরকের পরিচয়, ডোমেইন ইত্যাদি।
2. **HELO/EHLO কমান্ড ব্যবহার করে** SMTP ক্লায়েন্ট নিজের ডোমেইন পরিচয় দেয়।  
এটি ছাড়া সার্ভার বুঝতে পারে না কোন ক্লায়েন্ট যোগাযোগ করছে।
3. এটি **mail session-এর প্রোটোকল স্তর শুরু করে** — যেমন authentication, capability negotiation (EHLO), ইত্যাদি।

ঝি তাই TCP সংযোগ “path তৈরি করে”, আর SMTP সংযোগ “session তৈরি করে।”

## 13. Show the connection establishment phase from [aaa@xxx.com](mailto:aaa@xxx.com) to [bbb@yyy.com](mailto:bbb@yyy.com).

এটি SMTP conversation-এর প্রথম ধাপ — যেখানে ক্লায়েন্ট ও সার্ভার একে অপরের সাথে পরিচিত হয়।

### □ ধাপে ধাপে:

ধাপ	SMTP Command / Response	অর্থ
①	<b>Client → Server:</b> TCP connection established	TCP স্তরে সংযোগ তৈরি
②	<b>Server → Client:</b> 220 yyy.com Service ready	সার্ভার জানায় যে এটি প্রস্তুত
③	<b>Client → Server:</b> HELO xxx.com	ক্লায়েন্ট নিজের ডোমেইন জানায়

ধাপ	SMTP Command / Response	অর্থ
④	<b>S</b> erver → <b>C</b> lient: 250 yyy.com	সার্ভার বলে “ঠিক আছে, স্বাগতম”

### ই উদাহরণ:

S: 220 yyy.com Service ready  
C: HELO xxx.com  
S: 250 yyy.com

**14. Show the message transfer phase from aaa@xxx.com to bbb@yyy.com. The message is “Good morning my friend.”**

**Message:** “Good morning my friend.”

### □ ধাপে ধাপে:

ধাপ	SMTP Command / Response	অর্থ
①	<b>C</b> : MAIL FROM:<aaa@xxx.com>	প্রেরকের ঠিকানা
②	<b>S</b> : 250 OK	সার্ভার গ্রহণ করেছে
③	<b>C</b> : RCPT TO:<bbb@yyy.com>	প্রাপকের ঠিকানা
④	<b>S</b> : 250 OK	গ্রহণ করা হয়েছে
⑤	<b>C</b> : DATA	বার্তা পাঠানোর সংকেত
⑥	<b>S</b> : 354 Start mail input; end with <CRLF>.<CRLF>	বার্তা পাঠানো শুরু করো
⑦	<b>C</b> : বার্তার বডি পাঠানো হয়	

From: aaa@xxx.com  
To: bbb@yyy.com  
Subject: Greeting

Good morning my friend.  
.

|⑧|**S**: 250 Message accepted for delivery | বার্তা সফলভাবে গৃহীত |

### ই উদাহরণ সম্পূর্ণ:

```
C: MAIL FROM:<aaa@xxx.com>
S: 250 OK
C: RCPT TO:<bbb@yyy.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: From: aaa@xxx.com
C: To: bbb@yyy.com
C: Subject: Greeting
C:
C: Good morning my friend.
C: .
S: 250 Message accepted for delivery
```

### 15. Show the connection termination phase from aaa@xxx.com to bbb@yyy.com.

বার্তা পাঠানো শেষে SMTP session বন্ধ করা হয়।

ধাপ	SMTP Command / Response	অর্থ
1	C: QUIT	ক্লায়েন্ট সেশন শেষ করতে চায়
2	S: 221 yyy.com closing transmission channel	সার্ভার সংযোগ বন্ধ করছে

#### উদাহরণ:

```
C: QUIT
S: 221 yyy.com closing transmission channel
```

### 16. User aaa@xxx.com sends a message to user bbb@yyy.com, which is forwarded to

user ccc@zzz.com. Show all SMTP commands and responses.

#### ঞ্চ পরিস্থিতি:

aaa@xxx.com → bbb@yyy.com → (forwarded to) ccc@zzz.com

অর্থাৎ, bbb@yyy.com মেইলটি পেয়ে ccc@zzz.com-এ ফরওয়ার্ড করে।

---

#### ▣ SMTP Commands and Responses (Step by Step)

(1) *aaa@xxx.com → yyy.com* সার্ভার

S: 220 yyy.com Service ready  
C: HELO xxx.com  
S: 250 yyy.com  
C: MAIL FROM:<aaa@xxx.com>  
S: 250 OK  
C: RCPT TO:<bbb@yyy.com>  
S: 250 OK  
C: DATA  
S: 354 Start mail input; end with <CRLF>.<CRLF>  
C: From: aaa@xxx.com  
C: To: bbb@yyy.com  
C: Subject: Greeting  
C:  
C: Good morning my friend.  
C: .  
S: 250 Message accepted for delivery  
C: QUIT  
S: 221 yyy.com closing transmission channel

(2) *Forwarding by bbb@yyy.com → zzz.com* সার্ভারে

S: 220 zzz.com Service ready  
C: HELO yyy.com  
S: 250 zzz.com  
C: MAIL FROM:<bbb@yyy.com>  
S: 250 OK  
C: RCPT TO:<ccc@zzz.com>  
S: 250 OK  
C: DATA  
S: 354 Start mail input; end with <CRLF>.<CRLF>  
C: From: aaa@xxx.com  
C: To: ccc@zzz.com  
C: Subject: Fwd: Greeting  
C:  
C: Good morning my friend.  
C: .  
S: 250 Message accepted for delivery  
C: QUIT  
S: 221 zzz.com closing transmission channel

---

**সংক্ষিপ্ত সারসংক্ষেপ**

**ধাপ**      **ধরণের নাম**

**মূল ক্রমান্তি**

- ① **Connection Establishment** HELO
- ② **Mail Transaction**            MAIL FROM, RCPT TO, DATA
- ③ **Message Body Transfer**    বাতার বড়ি পাঠানো
- ④ **Connection Termination**    QUIT
- ⑤ **Forwarding (optional)**     নতুন MAIL FROM, RCPT TO, DATA সাইকেল শুরু

**17. User aaa@xxx.com sends a message to user bbb@yyy.com. The latter replies.**

**Show all SMTP commands and responses.**

চমৎকার শুভ্রাৰ্থ — এখন আমরা দেখব কীভাবে SMTP প্ৰোটোকলে একটি মেইল পাঠানো এবং তাৰপৰ তাৰ রিপ্লাই পাঠানো কাজ কৰে।

অৰ্থাৎ দুইটি ধাপে প্ৰক্ৰিয়াটি হৰে:

- 1 aaa@xxx.com → bbb@yyy.com (মূল বাৰ্তা পাঠানো)
  - 2 bbb@yyy.com → aaa@xxx.com (রিপ্লাই পাঠানো)
- 

### SMTP Command Flow — Full Conversation Example

**Phase 1: [aaa@xxx.com](#) → [bbb@yyy.com](#)**

(aaa তাৰ সাৰ্ভাৱ xxx.com থেকে yyy.com-এ মেইল পাঠায়)

ধাপ	SMTP Command/Response	ব্যাখ্যা
1	S: 220 yyy.com Service ready	yyy.com সাৰ্ভাৱ সংযোগ গ্ৰহণেৰ জন্য প্ৰস্তুত
2	C: HELO xxx.com	প্ৰেৰক সাৰ্ভাৱ নিজেৰ ডোমেইন জানায়
3	S: 250 yyy.com	সাৰ্ভাৱ সাড়া দেয়
4	C: MAIL FROM:<aaa@xxx.com>	প্ৰেৰকেৰ ঠিকানা
5	S: 250 OK	গৃহীত হয়েছে
6	C: RCPT TO:<bbb@yyy.com>	প্ৰাপকেৰ ঠিকানা
7	S: 250 OK	গৃহীত হয়েছে
8	C: DATA	বাৰ্তা পাঠানোৰ জন্য প্ৰস্তুতি
9	S: 354 Start mail input; end with <CRLF>.<CRLF>	মেইল কন্টেন্ট পাঠানো শুৰু কৰো
10	C: বাৰ্তা পাঠানো:	

From: aaa@xxx.com

To: bbb@yyy.com  
Subject: Hello

Hi bbb,  
How are you?  
.

1[S: 250 Message accepted for delivery	বার্তা সফলভাবে গৃহীত
1[C: QUIT	সংযোগ বন্ধ
1[S: 221 yyy.com closing transmission channel	সংযোগ বন্ধ নিশ্চিত করা

---

## Phase 2: [bbb@yyy.com](mailto:bbb@yyy.com) → [aaa@xxx.com](mailto:aaa@xxx.com) (Reply)

এবার bbb@yyy.com তার সার্ভার (yyy.com) থেকে aaa@xxx.com (xxx.com সার্ভার)-এ রিপ্লাই পাঠায়।

ধাপ	SMTP Command/Response	ব্যাখ্যা
1	S: 220 xxx.com Service ready	xxx.com সার্ভার প্রস্তুত
2	C: HELO yyy.com	প্রেরক সার্ভার নিজের ডোমেইন জানায়
3	S: 250 xxx.com	সার্ভার সাড়া দেয়
4	C: MAIL FROM:<bbb@yyy.com>	প্রেরক (bbb)
5	S: 250 OK	গৃহীত
6	C: RCPT TO:<aaa@xxx.com>	প্রাপক (aaa)
7	S: 250 OK	গৃহীত
8	C: DATA	রিপ্লাই বার্তা পাঠানোর প্রস্তুতি
9	S: 354 Start mail input; end with <CRLF>. <CRLF>	বার্তা পাঠানো শুরু করো
10	C: বার্তা:	

From: bbb@yyy.com  
To: aaa@xxx.com  
Subject: Re: Hello

Hi aaa,  
I'm fine. How about you?

| 1[S: 250 Message accepted for delivery | মেইল গৃহীত হয়েছে।  
| 1[C: QUIT | সংযোগ বন্ধ।  
| 1[S: 221 xxx.com closing transmission channel | সংযোগ বন্ধ নিশ্চিত।

---

## 💡 সংক্ষিপ্ত ব্যাখ্যা

ধাপ      SMTP কমান্ড      উদ্দেশ্য

HELO      সার্ভার পরিচয় দেওয়া

MAIL FROM প্রেরকের ঠিকানা

RCPT TO প্রাপকের ঠিকানা

DATA      মেইল বার্তা পাঠানো

QUIT      সংযোগ বন্ধ করা

---

## ☑ সম্পূর্ণ উদাহরণ (সংক্ষেপে)

### প্রথম মেইল:

```
S: 220 yyy.com Service ready
C: HELO xxx.com
S: 250 yyy.com
C: MAIL FROM:<aaa@xxx.com>
S: 250 OK
C: RCPT TO:<bbb@yyy.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: From: aaa@xxx.com
C: To: bbb@yyy.com
C: Subject: Hello
C:
C: Hi bbb, How are you?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 yyy.com closing transmission channel
```

### রিপ্লাই মেইল:

```
S: 220 xxx.com Service ready
```

```
C: HELO yyy.com
S: 250 xxx.com
C: MAIL FROM:<bbb@yyy.com>
S: 250 OK
C: RCPT TO:<aaa@xxx.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: From: bbb@yyy.com
C: To: aaa@xxx.com
C: Subject: Re: Hello
C:
C: Hi aaa, I'm fine. How about you?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 xxx.com closing transmission channel
```

୨୦୧୫

- aaa@xxx.com প্রথমে bbb@yyy.com-কে মেইল পাঠায়।
  - bbb@yyy.com তারপরে একই SMTP প্রক্রিয়া ব্যবহার করে রিপ্লাই পাঠায় aaa@xxx.com-এ।  
দুটো ক্ষেত্রেই SMTP কমান্ডের ধারা একই থাকে — কেবল প্রেরক ও প্রাপক বদলে যায়।

**18. In SMTP, if we send a one-line message between two users, how many lines of commands and responses are exchanged?**

চমৎকার প্রশ্ন

এটি SMTP প্রোটোকলের command-response structure বোঝার জন্য খুব গুরুত্বপূর্ণ। চলুন ধাপে ধাপে বিশ্লেষণ করি দ্বি-

ପ୍ରମ୍ଳୟ

In SMTP, if we send a **one-line message** between two users, how many **lines of commands and responses** are exchanged?

## ପାଠ୍ୟମିଶ୍ରମ (Background)

SMTP (Simple Mail Transfer Protocol)-এ একটি ইমেইল পাঠানোর জন্য কয়েকটি ধাপ থাকে:

## 1. Connection establishment (HELO handshake)

2. **Mail transaction** (MAIL FROM, RCPT TO, DATA)
3. **Message transfer** (ইমেইল বডি পাঠানো)
4. **Connection termination** (QUIT)

প্রতিটি SMTP কমান্ডের পরে সার্ভার একটি **response line** পাঠায়।  
তাই মোট লাইন গণনার সময় **command + response** — উভয়কেই গণনা করতে হবে।

---

#### ১৩৪ ধাপে ধাপে SMTP Conversation (একটি এক-লাইন মেসেজ পাঠানোর ক্ষেত্রে)

ধরা যাক:

- **প্রেরক:** aaa@xxx.com
- **প্রাপক:** bbb@yyy.com
- **বার্তা:** "Hi." (একটি মাত্র লাইন)

#### সম্পূর্ণ SMTP ডায়ালগ:

```

S: 220 yyy.com Service ready           ← (response)
C: HELO xxx.com                      ← (command)
S: 250 yyy.com                       ← (response)
C: MAIL FROM:<aaa@xxx.com>          ← (command)
S: 250 OK                            ← (response)
C: RCPT TO:<bbb@yyy.com>            ← (command)
S: 250 OK                            ← (response)
C: DATA                             ← (command)
S: 354 Start mail input; end with <CRLF>. <CRLF>  ← (response)
C: From: aaa@xxx.com                ← (header line, part of message)
C: To: bbb@yyy.com                  ← (header line)
C: Subject: Test                   ← (header line)
C:
C: Hi.                                ← (message body – one line)
C: .                                 ← (end of DATA)
S: 250 Message accepted for delivery  ← (response)
C: QUIT                           ← (command)
S: 221 yyy.com closing transmission channel ← (response)
  
```

---

#### ১৩৫ এখন লাইন গুনি

ধাপ	বিবরণ লাইন সংখ্যা
Connection greeting (220)	1
HELO + response	2
MAIL FROM + response	2

ধাপ	বিবরণ লাইন সংখ্যা
RCPT TO + response	2
DATA + response	2
Message body lines (including headers + one-line message + terminating dot)	প্রায় 5
Final response after DATA	1
QUIT + response	2

---

### মোট:

```

1 (greeting)
+ 2 (HELO)
+ 2 (MAIL FROM)
+ 2 (RCPT TO)
+ 2 (DATA)
+ 5 (message lines)
+ 1 (final 250)
+ 2 (QUIT)
= 17 lines total

```

---

### উত্তর:

একটি এক-লাইন মেইল পাঠাতে প্রায় **১৭টি লাইন** (commands + responses) বিনিময় হয়।

❖ (সংখ্যাটি সামান্য পরিবর্তিত হতে পারে যদি হেডার সংখ্যা কম/বেশি হয়,  
কিন্তু সাধারণভাবে 16–18 লাইনের মধ্যে থাকে।)

---

### সংক্ষেপে:

ধাপ	কমান্ড/রেসপন্স সংখ্যা
Connection setup	3
Mail transaction (MAIL, RCPT, DATA)	6
Message body (headers + one line + ".")	~5
Termination	2

ধাপ

কমান্ড/রেসপন্স সংখ্যা

মোট

~17 লাইন

---

চাইলে আমি এই পুরো প্রক্রিয়ার একটি **line count diagram** (প্রতিটি লাইন নম্বরসহ SMTP ফ্লো) তৈরি করে দিতে পারি — এতে বোঝা আরও সহজ হবে।  
আপনি কি সেটি চান?