The email query service automatically downloads new emails for a given email id. The email-id/passwords will be stored in a user credential database.

The service downloads the following parameters of an email:
Sender, Recipient, Subject, Message body, Attachments.

Each user will have his/her own folder to which emails are downloaded. Within the user folder, each email is assigned a folder with a unique name (referred to as Message ID (MID)), which is formatted as:
{<recipient_emailId><sender_emailId><date-timestamp>}

All files for a given email will be saved in its own folder. The attachments (formats to be downloaded can be selected e.g. PDF, .docx, txt) will be downloaded & saved to the directory, and the parameters are saved in an email metadata file (formatted as text, XML or JSON).

On completion of download of metadata & attachments, the query service will call the email dispatch service. The email dispatch service retrieves data for the particular email from its MID folder, and prepares it for transmission across the air-gap. The SHA value of the allowed formats (PDF, docx, txt) is calculated and the files are compressed & encrypted (optional).

The receiver remains in an infinite loop: it listens for a CONTROL packet (of a constant length). If the packet is available, it reads it; else a time out is triggered that restarts the process. The CONTROL packet sent by the sender can be of two types: a DATAH (data header) which indicates that the next packet will be a data packet, or a FINISH, which indicates that the current file transfer is over, and forces the previously received DATA packets to be processed (hash verification, decompression and flush to disk).

The DATAH packet will contain the number of bytes in the upcoming DATA packet, and a value that indicates the probable transfer time (plus 10%). This ensures that the receiver increases the serial port "bytes to be read" and "read timeout" value, which is initially set to a few bytes & seconds when listening for the DATAH (header) packet (to ensure quick, low latency setup of transfer). The new larger byte-value & longer timeout allows large packets containing data to be received.

The DATA packet will consist of:
<length(MID+filename.format)><MID><filename.format><fileSHA256value><PacketNumber><TotalPackets><TotalPAYLOADBytes><PAYLOAD><CRC>

The FINISH packet is used to indicate the end of transfer, and merely instructs the receiver to process the preceding data packets: verify hashes, decompress and write to MID folder. The receiver returns a value on completion, goes to the start of its infinite loop & waits for the next control packet.

SENDER   **RECEIVER**

Begin

Write DATAH control packet → Read control packet ← Timeout or Invalid data

Check control packet TYPE

Packet TYPE = FINISH

Packet TYPE = DATA

Send ACK_DATAH packet

INVALID data — Read packet

Set timeout/bytes to read based on DATA_H

ACK = Receiver ready

Read DATA packet

Send DATA packet "n"

Verify CRC

Valid   Invalid

Read reply ← Reply ACK (Valid data)   Reply INVALID Request retransmission

If INVALID, re-transmit packet "n"

Reply = ACK or INVALID?

Save Packet ⟷ Fetch packets

If ACK

If YES, n = n+1 (move on to next packet)

More packets to be sent?

1) Verify SHA256 hash
2) Decrypt/ Decompress
3) Write to directory on disk

Write FINISH packet → Read STATUS packet → Return (STATUS)

Write "SUCCESS" or "FAIL"