Chapter 15: Resumable Upload

In a **resumable upload**, the file is uploaded in multiple smaller parts (chunks) rather than all at once. This approach is useful for large files, as it allows the upload to resume from the last successful chunk in case of network interruptions or other errors, without starting over.

Here's a step-by-step description of the resumable upload process:

Step 1: Initiate the Upload Session

1. **Client Request**: The client first requests an upload session by sending a request to an initiation endpoint with metadata, such as the file name and file size.
2. **Server Response**: The server responds with a unique identifier for the upload session (`uploadId`) and an endpoint URL to use for uploading chunks. This `uploadId` will be used to keep track of which chunks belong to which upload.

   Example:

   - **Request**: `POST <https://example.com/resumable-upload/initiate`>
   - **Response**: `{"uploadId": "12345", "uploadUrl": "<https://example.com/resumable-upload/chunk>"}`

Step 2: Upload File Chunks

1. **Chunk Upload**: The client starts uploading the file in chunks. Each chunk is sent with a `Content-Range` header specifying the byte range of that chunk within the file (e.g., `bytes 0-524287/1048576`).
2. **Using the `uploadId`**: The client includes the `uploadId` as a query parameter or header to indicate which upload session the chunk belongs to.
3. **Server Processing**: The server appends each chunk to the file it's constructing on the server side. It verifies the `Content-Range` to ensure the correct placement of each chunk.
4. **Response for Each Chunk**: After receiving and processing each chunk, the server responds with a success message (or an error if there's an issue).

   Example:

- **Request**: `PUT <https://example.com/resumable-upload/chunk?uploadId=12345` >
- **Headers**: `Content-Range: bytes 0-524287/1048576`
- **Response**: `200 OK`

Step 3: Resume if Interrupted

1. **Checking Progress**: If the upload is interrupted (e.g., due to a network issue), the client can request the server for the last successfully uploaded chunk. Alternatively, the client can keep track of the last chunk uploaded.
2. **Resuming**: The client resumes by uploading the next chunk, continuing until the entire file is uploaded.

Step 4: Complete the Upload

Once all chunks are uploaded, the server assembles the file. At this point, the server may send a confirmation response indicating the upload is complete.

Benefits of Resumable Uploads

- **Reliability**: Resumable uploads prevent starting over from scratch after interruptions, saving time and bandwidth.
- **Control**: Large files can be uploaded in chunks, making it easier to manage and monitor upload progress.

This process is widely used for uploading large files in applications like cloud storage, video uploads, or data transfer services.