[**https://github.com/minio/minio/issues/21611**](https://github.com/minio/minio/issues/21611)

**Payload Rejection in MinIO Server**

**Question 1A.**

When payload is sent, the MinIO server rejects uploads with AWS-chunked encoding. Why?

**EXPLANATION**

When a client uploads a large object to an Amazon S3-compatible storage (like MinIO), the SDK may use a special transfer method called AWS Chunked Transfer Encoding.

This means:

The payload (file data) is divided into chunks.

Each chunk is hashed and signed individually with AWS Signature v4.

This helps upload large data streams without knowing the total size beforehand.

Example header:

Transfer-Encoding: aws-chunked

Content-Encoding: aws-chunked

MinIO does not support AWS chunked encoding for payloads.

Here’s why

Reason Explanation

Simplified Design - MinIO expects a known content length during upload to verify data

integrity using checksums (MD5, SHA256). Chunked uploads break this assumption.

Signature Validation - AWS chunked transfer involves per-chunk signatures and streaming authentication, which MinIO intentionally avoids for simplicity and speed.

Performance Optimization - MinIO is designed for high-throughput uploads via multipart APIs (PUT Object, Multipart Uploads), not chunked streams. Chunked encoding adds unnecessary overhead.

Protocol Compatibility - AWS uses chunked encoding mainly for streaming upload APIs (like PUT without content length). MinIO implements multipart upload instead, which is cleaner and compatible.

S3 API Simplification MinIO aims for S3 API compatibility but not every edge-case behavior — chunked payload encoding is one such rarely used feature.

**SUMMARY**

Point Description

Encoding Type - AWS-Chunked Transfer Encoding

Issue - MinIO rejects payloads using this encoding

Root Cause - MinIO doesn’t support per-chunk signature validation & expects fixed-length payloads

Recommended Fix - Use standard multipart uploads instead of chunked encoding

Example Workaround

If you’re using an AWS SDK, disable chunked encoding:

For Python (boto3):

import boto3

from botocore.config import Config

config = Config(s3={'use\_accelerate\_endpoint': False, 'addressing\_style': 'path'})

s3 = boto3.client('s3',

endpoint\_url='https://minio.example.com',

aws\_access\_key\_id='minioadmin',

aws\_secret\_access\_key='minioadmin',

config=config)

# Disable chunked transfer

s3.upload\_file('data.csv', 'mybucket', 'data.csv', ExtraArgs={'ContentLength': 12345})

Or in AWS CLI, use:

aws s3api put-object --body file.txt --bucket mybucket --key file.txt --no-verify-ssl

**CONCLUSION**

MinIO rejects uploads with AWS-chunked encoding because it expects a known content length and does not implement AWS’s per-chunk signature streaming logic.

Instead, MinIO supports multipart uploads, which are more efficient and standards-compliant within the S3 API model.