Comma

ARCHIVE EXPORT - S3 SETUP

Deliver Your Archive to Amazon S3

Step-by-step setup for delivering your Comma compliance archive into a customer-owned S3 bucket

Comma Compliance · 2026-07-06

ARCHIVE EXPORT - S3 SETUP

Deliver Your Archive to Amazon S3

Step-by-step setup for delivering your Comma compliance archive into a customer-owned S3 bucket

This guide covers everything you provision on the **Amazon S3** side so Comma can deliver your archive into a bucket you own.

Delivery uses a **static IAM access key you create and hand to Comma**. The engine only ever calls three S3 actions, scoped to one prefix:

Action	Used for
`s3:PutObject`	Writing each delivery batch and attachment
`s3:GetObject`	Reading back the one connection-probe object during Verify & activate
`s3:DeleteObject`	Removing that one probe object

No `ListBucket`, no bucket-level permissions, nothing outside your prefix.

What to gather

Have these four values ready - you will send the first three plus the key to Comma:

- **Bucket name** - must be DNS-compatible: 3-63 chars, lowercase `a-z`, `0-9`, and hyphens, starting and ending alphanumeric. **No dots** (Comma rejects dotted bucket names because they break virtual-hosted-style TLS).
- **Region** - e.g. `us-east-1`.
- **Prefix** - the object-key prefix Comma delivers under, e.g. `archive/` (or blank for the bucket root).
- **Access key** - the IAM access key id + secret you create below.

Step 1 - the least-privilege IAM policy

This is the exact policy. Replace `YOUR_BUCKET_NAME` and `YOUR_PREFIX/` (use `arn:aws:s3:::YOUR_BUCKET_NAME/*` if you deliver at the bucket root):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CommaBlobExportDelivery",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME/YOUR_PREFIX/*"
    }
  ]
}
```

Step 2 - create the delivery user

Run this as an AWS admin (it needs `aws` v2). It creates a dedicated IAM user, attaches the scoped policy, and prints an access key:

```
#!/usr/bin/env bash
set -euo pipefail

# ---- fill these in ----
BUCKET="your-archive-bucket"     # lowercase, hyphens, no dots
REGION="us-east-1"
PREFIX="archive/"                # trailing slash; or "" for the bucket root
IAM_USER="comma-blob-export"
# -----------------------

POLICY=$(cat <<JSON
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CommaBlobExportDelivery",
      "Effect": "Allow",
      "Action": ["s3:PutObject", "s3:GetObject", "s3:DeleteObject"],
      "Resource": "arn:aws:s3:::${BUCKET}/${PREFIX}*"
    }
  ]
}
JSON
)

aws iam create-user --user-name "$IAM_USER" >/dev/null 2>&1 || true
aws iam put-user-policy --user-name "$IAM_USER" \
  --policy-name comma-blob-export --policy-document "$POLICY"

echo "Access key (store the secret securely - it is shown only once):"
aws iam create-access-key --user-name "$IAM_USER"
```

Step 3 - verify the key works in Comma's shape

Before you hand the key over, prove it can write, read back, and delete under your prefix - the same probe Comma runs on Verify & activate. Export the **delivery** key (not your admin key) and run:

```
#!/usr/bin/env bash
set -euo pipefail

BUCKET="your-archive-bucket"; REGION="us-east-1"; PREFIX="archive/"
# export AWS_ACCESS_KEY_ID=...  AWS_SECRET_ACCESS_KEY=...   # the delivery key

KEY="${PREFIX}_comma_probe_$(uuidgen | tr '[:upper:]' '[:lower:]')"
printf 'comma blob export connection probe' > /tmp/comma-probe.txt

aws s3api put-object --region "$REGION" --bucket "$BUCKET" --key "$KEY" \
  --body /tmp/comma-probe.txt --checksum-algorithm SHA256 >/dev/null && echo "
write  OK"
aws s3api get-object --region "$REGION" --bucket "$BUCKET" --key "$KEY" /tmp/comma-
rb.txt >/dev/null \
  && [ "$(cat /tmp/comma-rb.txt)" = "comma blob export connection probe" ] && echo
"  read   OK"
aws s3api delete-object --region "$REGION" --bucket "$BUCKET" --key "$KEY" >/dev/
null && echo "  delete OK"

echo "PASS - this key can deliver in the shape Comma needs."
```

If any line fails with `AccessDenied`, the policy is missing that action on your prefix - fix Step 1 and re-run.

Step 4 - send Comma the details

Send your account team, over a secure channel: **provider (Amazon S3), bucket, region, prefix, the access key id, and the secret access key.** Comma runs a one-time Verify & activate probe and turns on delivery. Delivery is go-forward from activation.

Optional hardening

- **Restrict by source IP.** All delivery leaves Comma from one static egress IP. To lock the key to it, add an `aws:SourceIp` condition to the policy statement. Ask your account team for the current egress IP first - confirm it before locking, as it can change with an infrastructure move.

```
"Condition": { "IpAddress": { "aws:SourceIp": ["<comma-egress-ip>"] } }
```

- **SSE-KMS buckets.** SSE-S3 (AES256) default encryption works with no extra permissions. If your bucket enforces **SSE-KMS with a customer-managed key**, also grant the delivery user `kms:GenerateDataKey` (for writes) and `kms:Decrypt` (for the read-back probe) on that key. `kms:Encrypt` is not needed - S3 SSE-KMS uses `GenerateDataKey`, not `Encrypt`.
- **Immutability (Object Lock).** Comma only ever writes new, content-addressed objects and deletes just its own probe object - it never overwrites or deletes your delivered data. S3 Object Lock / a retention policy on the bucket is fully compatible and strengthens your WORM posture.

Good to know

- **Integrity is verified server-side.** Every `PutObject` declares an `x-amz-checksum-sha256`; S3 validates the body against it and rejects a mismatch.
- **TLS always.** Comma uses the standard regional virtual-hosted S3 HTTPS endpoint - never a custom endpoint.
- **Deterministic, idempotent.** A retried batch re-writes the same objects with byte-identical content, so re-delivery is always safe.

> **ALSO AVAILABLE ONLINE**
>
> The same guide lives at docs.commacompliance.com under Guides - "Archive export to Amazon S3." Your Comma account team provisions the destination once you hand over the details.