DATA COLLECTION AGENT · CUSTOMER-HELD KMS KEY

Encrypting your Milestone data with your own KMS key

Milestone can keep your raw data in its own S3 bucket, encrypted with a KMS key in **your** AWS account. If you disable the key or remove Milestone's access to it, Milestone can no longer read or write that data. Every use of the key shows in your own CloudTrail.

Your Milestone contact sends you the values used below:

- `<bucket>` — the name of your data's S3 bucket;
- `<milestone-account-id>` — Milestone's AWS account ID;
- `<test-link>` — a test upload link, on request (step 4).

1. Create the key

- A **symmetric** encryption key (key spec `SYMMETRIC_DEFAULT`, usage *Encrypt and decrypt*).
- In **us-east-1**, the bucket's Region. A key in another Region doesn't work.
- Don't add Milestone's account through the console's "Other AWS accounts" option: it grants more than Milestone needs, with no conditions. Use step 2.

2. Add Milestone's statements to the key policy

Edit the key policy, keep your own administrators' statement, and add these two statements to its `Statement` list. Replace every `<bucket>` and `<milestone-account-id>`.

```
{
  "Sid": "AllowMilestoneS3UseForTenantBucket",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::<milestone-account-id>:root"},
  "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "s3.us-east-1.amazonaws.com",
      "kms:CallerAccount": "<milestone-account-id>"
    },
    "StringLike": {
      "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<bucket>/*"
    }
  }
}
```

```
{
  "Sid": "AllowMilestoneKeyCheck",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::<milestone-account-id>:root"},
  "Action": "kms:DescribeKey",
  "Resource": "*",
  "Condition": {"StringEquals": {"kms:CallerAccount": "<milestone-account-id>"}}
}
```

What they allow:

- **The first statement** lets Milestone encrypt and decrypt with the key only through S3 in us-east-1, only for objects in `<bucket>`, and only from Milestone's account.
- **The second** lets Milestone check the key when it is set up. It returns the key's settings, never data.
- **They name Milestone's account, not its individual services.** AWS allows a cross-account key use only when both your key policy and Milestone's own IAM policies allow it, so Milestone's IAM policies decide which of its services can use the key. A new Milestone service then needs no change on your side. Naming Milestone's individual roles instead isn't supported.

- **Your organization's own guardrails must allow it.** If your AWS Organization restricts KMS access to its own accounts (for example a resource control policy), add an exception for this key and Milestone's account.

The bucket encrypts each object separately (S3 Bucket Keys are off). So every read and write of an object is a KMS call, logged in your CloudTrail, and the first statement can be limited to your bucket's objects. If your trail excludes KMS events, these calls show only in CloudTrail's event history. Milestone's account pays for these calls and they count against its KMS request limits, not yours.

3. Send Milestone the key's ARN

Send your Milestone contact the key's full ARN, `arn:aws:kms:us-east-1:<your-account-id>:key/<key-id>`. An alias doesn't work. Milestone then makes the key the bucket's default encryption, and checks it with a test write and read-back before any of your data is stored there.

4. Allow the bucket on your network

Every host that runs the Data Collection Agent, or another Milestone tool that uploads files, must reach `<bucket>.s3.amazonaws.com` on outbound port 443, in addition to the hosts in the agent's install guide. Milestone tells you before it switches your uploads to the bucket; add the host **before** then. If a host reaches S3 through a VPC endpoint whose policy lists allowed buckets, add `<bucket>` to that policy too.

To check a host, ask Milestone for `<test-link>` and run this on the host, with the link in single quotes. The link uses the same host as the agent's uploads. If the host reaches the internet through a proxy, run it with the same proxy settings (for example `HTTPS_PROXY`).

```
curl -sS -X PUT -H 'Content-Type:' --data-binary 'milestone-network-check' \
  -w '\nHTTP %{http_code}\n' '<test-link>'
```

- `HTTP 200`: the host reaches the bucket. Tell Milestone, which confirms it received the upload.
- `ExpiredToken`, or `Request has expired`, in the response: the link expired, which can happen in less than its hour. Ask Milestone for a new link. This is not a network problem.
- `AccessDenied` in the response: the host reached the bucket, but the upload was refused. If the message names `kms:GenerateDataKey`, check the key policy (step 2). Otherwise, send Milestone the full response.

- `KMS.DisabledException` or `KMS.KMSInvalidStateException`: the key is disabled or scheduled for deletion. Restore it (step 6).
- A connection error, a time-out, or `HTTP 000`: the host can't reach the bucket. Check your firewall or proxy allowlist.

5. What your key covers

Your key covers **the contents of every file stored in `<bucket>`**, whichever Milestone service writes it: the agent's uploads, including its own logs and settings, AI-tool telemetry, deployment events, and the raw tables built from them.

It doesn't cover, and Milestone keeps under its own encryption:

- data built from it outside the bucket, such as analytics tables, databases, dashboards, backups and service logs;
- copies kept elsewhere: the agent's logs in Milestone's log service, and working copies made while processing or investigating your data;
- the original files you send Milestone outside the agent, such as HR or software-catalog files;
- the names, sizes and dates of the files in `<bucket>`, and the names and types of your data's columns, never their contents.

Removing a repository. When you ask Milestone to remove a repository, its raw files are kept for up to 90 days as a recovery copy under Milestone's encryption, outside your key, and then deleted.

6. Revoking and rotating

- **Revoke** by disabling the key, scheduling its deletion, or removing Milestone's statements. Milestone can then no longer read or write the contents of the files in `<bucket>`. Everything listed as not covered in step 5 stays readable by Milestone. Re-enabling the key, or restoring the statements, restores access. A key scheduled for deletion must first have its deletion cancelled, within the waiting period, and then be enabled.
- **Deployment events need a retry.** While the key is unavailable, Milestone refuses the deployment events your systems push to it (HTTP 500) and doesn't store them. An event not re-sent is lost, so have the sender keep each event until it gets a 2xx response, or re-send them once the key is back.
- **A deleted key can't be undone.** The data in `<bucket>` becomes permanently unreadable.
- **Automatic key rotation is fine.** It keeps the key's ARN, and older data stays readable.

- **Switching to a different key isn't supported.** Talk to Milestone first.

7. Opting out

Tell Milestone. Milestone moves the data in `<bucket>` off your key, onto its own. **Keep the key enabled and its policy unchanged until Milestone confirms the move is done.** After that you may disable or delete the key. Data not yet moved when the key stops working stays unreadable, and is lost for good if the key is deleted.
