Amazon S3 Cross-Account Repository

This document explains how a client sets up an Amazon S3 bucket in **their own AWS account** so that it can be used as an S3 repository by an AppvanceIQ (AIQ) Controller that is hosted in Appvance's AWS account, using AIQ's **cross-account S3 mode**.

Overview

In this setup:

- The **AIQ Controller** runs in the **Appvance AWS account** (`650030447088`).
- The **S3 bucket** used as the repository lives in the **client's AWS account**.
- The Controller reaches the client's bucket by **assuming an IAM role** in the client account via AWS STS (`sts:AssumeRole`), rather than using long-lived access keys.

This is the recommended pattern because the client never has to share IAM access keys/secrets with Appvance, access is scoped to a single role with only the S3 permissions needed, and the client retains full control (revocable at any time by deleting/disabling the role or its trust policy).

High-level flow

1. Client creates an S3 bucket in their account.
2. Client creates an IAM role in their account with a trust policy allowing the Appvance Controller's specific IAM role to assume the role, and a permissions policy scoping access to just that bucket.
3. Client shares the **Role ARN** (and, if used, the **External ID**) with Appvance.
4. Appvance configures the AIQ Controller's S3 repository settings to use cross-account mode with that Role ARN.
5. At runtime, the Controller's AWS integration layer calls `sts:AssumeRole`, receives temporary credentials, and uses those for all S3 operations against the bucket.

Prerequisites

- Access to the client's AWS Console / CLI with permissions to create S3 buckets and IAM roles/policies.
- The Appvance AWS account ID: **650030447088**, and the Appvance Controller's specific IAM role ARN for this engagement (e.g. `arn:aws:iam::650030447088:role/<clientname>-aiq-controller-role`) — Appvance provisions one such role per customer, so confirm the exact ARN with the Appvance team before starting.
- A name/tag convention for the bucket and role (e.g. `clientname-aiq-repo`, `clientname-aiq-crossaccount-role`).
- Decide whether to use an **External ID** for the trust policy (strongly recommended — see Security Recommendations below).

Step 1 — Create the S3 Bucket

1. In the client AWS account, go to **S3 → Create bucket**.
2. Choose a globally unique bucket name, e.g. `clientname-aiq-repo`.
3. Select the AWS region (ideally the same or nearest region to the Appvance Controller for latency).
4. Recommended settings:
   - **Block Public Access**: Keep all four options enabled (bucket should never be public).
   - **Bucket Versioning**: Enable (protects against accidental overwrite/delete of test artifacts).
   - **Default encryption**: Enable SSE-S3 (AES256), or SSE-KMS — either the AWS-managed default key (`alias/aws/s3`) or a customer-managed key — if the client prefers KMS-based encryption.
   - **Object Ownership**: Bucket owner enforced (ACLs disabled).
5. Create the bucket.

Step 2 — Create the Cross-Account IAM Role

This role is what the AIQ Controller in the Appvance account will assume.

1. Go to **IAM → Roles → Create role**.
2. Choose **Custom trust policy** and enter a trust policy that allows the Appvance Controller's specific IAM role to assume this role. This is the default recommendation — Appvance provisions a dedicated controller role per customer (e.g. `arn:aws:iam::650030447088:role/clientname-aiq-controller-role`), and scoping the trust policy to that exact role is tighter than trusting the whole Appvance account. Example (using an External ID, recommended):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::650030447088:role/clientname-aiq-controller-role"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "REPLACE_WITH_AGREED_EXTERNAL_ID"
        }
      }
    }
  ]
}
```

Replace `clientname-aiq-controller-role` with the actual Controller role ARN Appvance provides for this customer. If Appvance instead indicates the integration should trust the account as a whole (less common), use `arn:aws:iam::650030447088:root` as the Principal instead. Confirm the exact principal with the Appvance team before finalizing.

3. Skip attaching AWS managed permission policies at this step — a custom least-privilege policy will be attached next.
4. Name the role, e.g. `clientname-aiq-crossaccount-role`.
5. Create the role.

Step 3 — Attach a Least-Privilege S3 Permissions Policy

Create and attach a customer-managed policy scoped to only the bucket created in Step 1.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
```

```
        ],
        "Resource": "arn:aws:s3:::clientname-aiq-repo"
    },
    {
        "Sid": "AllowObjectReadWriteDelete",
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject",
            "s3:GetObjectVersion",
            "s3:AbortMultipartUpload",
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "arn:aws:s3:::clientname-aiq-repo/*"
    }
  ]
}
```

Notes:

- Replace `clientname-aiq-repo` with the actual bucket name in both Resource lines.
- If the Controller writes into a specific prefix/folder only (e.g. `aiq/`), further scope Resource to `arn:aws:s3:::clientname-aiq-repo/aiq/*` for tighter isolation.
- If the bucket uses SSE-KMS with a **customer-managed key**, also attach `kms:Decrypt`, `kms:GenerateDataKey`, and `kms:DescribeKey` permissions scoped to that key's ARN, and add the role as a key user in the KMS key policy.
- If the bucket uses SSE-KMS with the **AWS-managed default key** (`alias/aws/s3`), the same three KMS actions (`kms:Decrypt`, `kms:GenerateDataKey`, `kms:DescribeKey`) are still required for cross-account access — a default key does not exempt a cross-account principal from needing explicit KMS permissions. IAM policies require the concrete key ARN rather than the alias, so resolve it first (e.g. `aws kms describe-key --key-id alias/aws/s3`) and use that ARN as the Resource. Also confirm the default key's key policy permits use by this role, since AWS-managed default keys have limited policy editability compared to customer-managed keys.

Attach this policy to the role created in Step 2.

Step 4 — (Optional but Recommended) Restrict the

Bucket Policy

In addition to the IAM role's permissions policy, add a bucket policy on the S3 bucket itself that only allows access from the specific cross-account role. This provides defense-in-depth: even if the IAM policy were ever broadened by mistake, the bucket policy still limits who can touch it.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAIQCrossAccountRoleOnly",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::CLIENT_ACCOUNT_ID:role/clientname-aiq-crossaccount-role"
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::clientname-aiq-repo",
        "arn:aws:s3:::clientname-aiq-repo/*"
      ]
    }
  ]
}
```

Replace `CLIENT_ACCOUNT_ID` with the client's own AWS account ID.

Step 5 — Share the Connection Details with Appvance

Once the bucket, role, and policies are created, share the following with the Appvance team so the AIQ Controller can be configured:

Item	Example	Notes

Bucket name	`clientname-aiq-repo`	
Bucket region	`us-east-1`	Must match what's configured in AIQ
Role ARN	`arn:aws:iam::CLIENT_ACCOUNT_ID:role/clientname-aiq-crossaccount-role`	Used by the Controller for `sts:AssumeRole`
External ID	`REPLACE_WITH_AGREED_EXTERNAL_ID`	Only if used in the trust policy
KMS Key ARN (if applicable)	`arn:aws:kms:...`	Required if bucket uses SSE-KMS — either the resolved default key ARN or a customer-managed key ARN

These details should be shared through a secure channel (not plaintext email), since the Role ARN plus External ID together control access to the bucket.

Step 6 — Configure the AIQ Controller (Appvance side)

On the AIQ Controller, the S3 repository is configured in **cross-account mode**, providing:

- Bucket name and region from the table above.
- Role ARN to assume in the client account.
- External ID, if the trust policy requires one.

Internally, AIQ's AWS integration layer performs an `sts:AssumeRole` call against the client's role, receives short-lived temporary credentials, and uses those credentials for all S3 operations (list, get, put, delete) against the bucket — no long-lived client credentials are ever stored in the Controller.

Step 7 — Validate the Setup

1. From the Controller, trigger a test scenario/run that writes to the S3 repo.
2. Confirm the object appears in the client's bucket (via S3 Console or CLI: `aws s3 ls s3://clientname-aiq-repo/`).
3. Confirm the Controller can list and read back existing objects from the bucket.

4. Check CloudTrail in the client account for `AssumeRole` and `s3:*` events from the Controller to confirm access is going through the intended role.

Security Recommendations

- Always use an **External ID** in the trust policy — mitigates the "confused deputy" problem and is an AWS-recommended best practice for third-party role assumption.
- Scope the trust policy Principal to the Appvance Controller's specific IAM role rather than the whole Appvance account root, where possible.
- Scope the IAM policy to the specific bucket (and prefix, if applicable) — never grant `s3:*` on `Resource: "*"`.
- Keep Block Public Access enabled on the bucket at all times.
- Enable bucket versioning so accidental deletes/overwrites can be recovered.
- Enable server-side encryption (SSE-S3 minimum, SSE-KMS if stricter compliance is required) — remember that SSE-KMS, even on the AWS-managed default key, requires explicit KMS permissions on the cross-account role.
- Enable S3 server access logging or CloudTrail data events on the bucket for auditability.
- Rotate the External ID and update the trust policy if the relationship with Appvance ever needs to be re-scoped or revoked.
- **Revocation**: access can be fully revoked at any time by deleting the IAM role, removing the trust policy statement, or removing the bucket policy statement — no coordination with Appvance is required.

Troubleshooting

Symptom	Likely Cause	Fix
`AccessDenied` on `AssumeRole`	Trust policy principal or External ID mismatch	Confirm the exact Principal ARN Appvance uses, and that the External ID matches exactly on both sides
`AccessDenied` on S3 operations after successful `AssumeRole`	IAM policy or bucket policy too narrow (wrong bucket name/prefix)	Verify Resource ARNs in both the role's permissions policy and the bucket policy match the actual bucket/prefix

Objects upload but can't be read back	KMS permissions missing on the cross-account role — this applies to both customer-managed keys and the AWS-managed default key (`alias/aws/s3`)	Add `kms:Decrypt` / `kms:GenerateDataKey` / `kms:DescribeKey` to the role's IAM policy scoped to the resolved key ARN, and for customer-managed keys also add the role as a key user in the KMS key policy
Intermittent `ExpiredToken` errors	Temporary credentials from AssumeRole expired mid-operation on long-running jobs	Confirm the Controller's AWS SDK layer is refreshing/re-assuming the role rather than caching credentials indefinitely

FAQ

Is IMDSv1 required on the client side for this setup?

No. Neither IMDSv1 nor IMDSv2 is required in the client's AWS account for this integration, and clients can safely leave IMDSv1 disabled.

This setup does not involve any EC2 instances running in the client's AWS account. It consists entirely of an IAM role, a trust policy, an S3 bucket policy, and STS `AssumeRole` calls initiated from the Appvance side. The AIQ Controller runs in Appvance's AWS account (`650030447088`) and calls `sts:AssumeRole` against the client's role to obtain temporary credentials — no compute resource in the client's account is ever involved, so there is no instance in the client's environment that would query the Instance Metadata Service (IMDS) as part of this flow.

Whatever IMDS credential-fetching occurs for the Controller itself happens on Appvance's own EC2 instance, in Appvance's own AWS account, and is entirely independent of and invisible to the client's environment.

For a client's security review: AWS's current guidance favors **IMDSv2-only (with IMDSv1 disabled)** as the more secure default, since it protects against SSRF-based credential theft. Leaving IMDSv1 off is aligned with AWS best practice, not against it.

This answer is scoped specifically to the S3 cross-account repository integration described in this document. If a client is also running other AIQ components inside their own AWS account — for

example, self-hosted TestNodes or agents on their own EC2 instances — those are separate compute resources with their own IMDS considerations, unrelated to this S3 cross-account setup.

Summary

The client only needs to create an S3 bucket, an IAM role with a scoped trust + permissions policy, and share the Role ARN (and External ID) with Appvance. No credentials ever leave the client's account — the AIQ Controller in Appvance's AWS account (650030447088) uses temporary, audit-able STS credentials obtained by assuming that role, keeping the client in full control of their own data at all times.

Revision #1
Created 2026-08-03 14:57:08 UTC by mmora
Updated 2026-08-03 14:57:55 UTC by mmora