Relaying files from S3 to an SFTP server without local staging

A common integration pattern is moving files from a cloud object storage bucket (S3 or any S3-compatible service) to an SFTP server that a partner or downstream system expects to read from. The naive approach downloads each file to local disk first and then uploads it, which wastes disk space and roughly doubles transfer time. AFT!'s `CopyToVFS` method eliminates the intermediate step by streaming data directly between the two VFS back-ends in a single call. This article shows how to build that relay as a scheduled cron job.

How It Works

You open two VFS handles: one for the S3 source and one for the SFTP destination. Both are resolved by name from the VFS Library, so no credentials appear in the script. Then you list the source directory, loop over the files, and call `src.CopyToVFS(srcPath, dest, dstPath)` for each one. The host application handles the streaming; your script is just the orchestration glue.

```
// S3 to SFTP relay
// Reads all files from a named S3 VFS and copies each one directly to a named
// SFTP VFS. No intermediate local file is created; CopyToVFS streams data
// between the two back-ends. Designed to run as a scheduled cron job.

var s3VfsName   = "your-s3-vfs-name";    // source: named S3 VFS in the VFS Library
var sftpVfsName = "your-sftp-vfs-name";  // destination: named SFTP VFS in the VFS Library
var sourceDir   = "/outbox";             // source directory on S3
var destDir     = "/inbound/from-s3";    // destination directory on SFTP

var src  = new VirtualFSByName(s3VfsName);
var dest = new VirtualFSByName(sftpVfsName);

// List the source directory sorted oldest-first so files arrive at the
// destination in the same order they were written.
var listing = src.ReadDir(sourceDir, SortByTime, SortAsc);
```

```
if (!listing.Ok()) {
    Log.Error("failed to list source directory: " + listing.ErrorMsg());
    Exit(1);
}

var entries = listing.Infos();
var relayed = 0;
var failed  = 0;

Log.Info("relaying " + entries.length + " entries from S3 to SFTP");

for (var i = 0; i < entries.length; i++) {
    var f = entries[i];

    // ReadDir returns both files and directories; skip directories.
    if (f.Type !== "FILE") {
        continue;
    }

    var srcPath = sourceDir + "/" + f.Name;
    var dstPath = destDir   + "/" + f.Name;

    var r = src.CopyToVFS(srcPath, dest, dstPath);
    if (!r.Ok()) {
        Log.Error("relay failed for " + f.Name + ": " + r.ErrorMsg());
        failed++;
    } else {
        Log.Info("relayed: " + f.Name + " (" + f.Size + " bytes)");
        relayed++;
    }
}

Log.Info("relay complete: " + relayed + " succeeded, " + failed + " failed");

if (failed > 0) {
    Exit(1);
}
```

`src.ReadDir(sourceDir, SortByTime, SortAsc)`: listing oldest-first means the SFTP server receives files in the order they were originally written to S3. For most downstream systems this ordering is

important, for example when files are numbered sequences or must be processed chronologically.

`f.Type !== "FILE"`: `ReadDir` returns both files and directory entries. Skipping non-file entries prevents the relay from trying to copy directory prefixes as if they were objects.

`Exit(1)` at the end: a non-zero exit code tells the AFT! job engine that the run was not entirely clean. This surfaces in the job history and can trigger alerts in external monitoring tools. A run that relayed everything successfully exits with the default code 0.

Turning the relay into a move

If files should be moved rather than copied, simply use `MoveToVFS` instead of `CopyToVFS`:

```
var r = src.MoveToVFS(srcPath, dest, dstPath);
if (r.Ok()) {
   relayed++;
} else {
   failed++;
}
```

Using this with other back-ends

The same script works for any combination of VFS types. Swap the named VFS profiles in the VFS Library to relay from Azure to GCS, from SFTP to S3, from local disk to Azure, and so on. The script logic does not change at all.

Revision #1
Created 12 April 2026 15:12:34 by DevTeam
Updated 12 April 2026 15:18:10 by DevTeam