# Getting alerted when a remote file has not been updated recently enough

Upstream systems fail silently all the time. An ETL job crashes, a partner's export process hangs, a cron entry gets accidentally deleted. When that upstream system deposits a file on an SFTP server for you to consume, you might not notice the failure until it bubbles up into a downstream data quality problem hours or days later. This article shows how to write a scheduled AFT! script that independently verifies that a specific remote file exists and has been updated within an acceptable time window, and sends an email alert if either condition fails.

## How It Works

The script connects to an SFTP server, calls `Stat` on the expected file path, and checks the modification timestamp. There are three outcomes: the connection itself fails (the server is unreachable), the file does not exist, or the file exists but is too old. The script sends an alert email for all three and exits with code `1`. If the file is current, a single informational log line is written and the script exits normally with no email sent.

```
// File staleness alert
// Checks whether an expected file on a remote SFTP server has been updated
// recently enough. Sends an email alert if the file is missing or has not been
// modified within the configured threshold. Designed to run on a schedule,
// for example every hour, to catch upstream pipeline failures early.

var expectedFilePath = "/remote/data/latest-export.csv"; // full remote path to verify
var maxAgeHours      = 25;    // alert if the file is older than this many hours

var alertFrom    = "aft-monitor@example.com";
var alertTo      = "ops-team@example.com";
var alertSubject = "AFT alert: stale or missing file on SFTP";

var cli = new SftpClient();
cli.Host = GetSecret("sftp-host");
```

```
cli.User = GetSecret("sftp-user");
cli.Pass = GetSecret("sftp-password");

if (!cli.Connect()) {
    // A connection failure means we cannot verify the file, so treat it as an alert condition.
    Log.Error("SFTP connection failed, sending alert");
    SendMail(
        alertFrom, alertTo, alertSubject,
        "AFT could not connect to the SFTP server to verify " + expectedFilePath + "."
    );
    Exit(1);
}

try {
    var info = cli.Stat(expectedFilePath);

    if (info === null) {
        var missingMsg = "Expected file is missing on the SFTP server: " + expectedFilePath;
        Log.Error(missingMsg);
        SendMail(alertFrom, alertTo, alertSubject, missingMsg);
        Exit(1);
    }

    var ageMs    = new Date().getTime() - new Date(info.TimeStamp).getTime();
    var ageHours = ageMs / (1000 * 60 * 60);

    if (ageHours > maxAgeHours) {
        var staleMsg = "File has not been updated in " + Math.floor(ageHours) + " hours " +
                       "(limit: " + maxAgeHours + "h). Path: " + expectedFilePath;
        Log.Warn(staleMsg);
        SendMail(alertFrom, alertTo, alertSubject, staleMsg);
    } else {
        Log.Info("file is current: " + expectedFilePath +
                 " (last modified " + Math.floor(ageHours) + "h ago)");
    }
} finally {
    cli.Close();
}
```

`maxAgeHours = 25`: set this slightly higher than the expected update interval to give a tolerance margin before alerting. A daily file that is normally updated at midnight should use something like 25 or 26 hours so a minor scheduling slip does not trigger a false alarm.

`cli.Stat(expectedFilePath)`: `Stat` returns a `DirListItem` object containing `Name`, `Type`, `Size`, and `TimeStamp`, or `null` if the file does not exist. Checking for `null` before accessing `info.TimeStamp` is essential; skipping the check will throw a runtime exception if the file is missing.

Connection failure is treated as an alert, not just an error to log and ignore: if AFT! cannot reach the SFTP server, it also cannot confirm the file is there, which is itself an alarm condition worth surfacing to your team.

## Alerting via Slack instead of email

Replace `SendMail` with `SendToSlackWebHook` everywhere in the script:

```
var webhookUrl = GetSecret("slack-webhook-url");
SendToSlackWebHook(webhookUrl, staleMsg, "AFT Monitor", ":warning:");
```

Store the webhook URL as a secret so it never appears in script source.

## Checking multiple files in one job

Wrap the `Stat` and age check logic in a helper function and call it for each path:

```
var monitors = [
    { path: "/feeds/customers.csv", maxHours: 25 },
    { path: "/feeds/orders.csv",    maxHours: 2  }
];
```

A single cron job can then cover your entire estate of expected files, and you get one combined alert email if anything is amiss.

---

Revision #1
Created 12 April 2026 14:40:22 by DevTeam
Updated 12 April 2026 14:47:00 by DevTeam