# Harborline stated control brief: privileged production MFA (current state)

This is **Harborline's current stated control**. It is a source document, not your design (your design lives on the Design tabs). It is the version used in SOC 2 and SOX ITGC binders. Use it in **Build** as the rule the monitor tests.

It is not automatically a good design. Complete **Assess** and **Design** before you treat this as finished. After **Break**, you will say what should change.

Harborline's control identifier: `HL-CC-PRIV-MFA-01`.

## Monitoring versus testing

**Periodic control testing** samples evidence after the fact (often once a quarter) to see whether a control operated in a period. A screenshot from 12 June does not prove the control operated every week in Q2.

**Control monitoring** evaluates a defined population on a defined cadence using source-system data, records the result, and keeps enough evidence that someone else can re-perform the evaluation.

**Continuous** does not mean real-time streaming. A control may run hourly, daily, weekly, on an event, or on another recurring cadence depending on risk, how often the source systems update, and what a failure would cost. Harborline currently *claims* a **weekly** run. Your Design must defend the frequency you choose.

Ask yourself:

- Why weekly?
- What risk would justify daily monitoring?
- What would make quarterly monitoring insufficient?
- What changes if the source data updates only once per day?

Frequency is a control-design decision, not a dashboard setting.

## The business problem

GRC still sends evidence requests. Owners send screenshots. Files sit in a tracker. Nobody can say, without a hunt, who was in the privileged production population last Tuesday, or whether MFA was actually required for that population.

A monitor is useful because it evaluates the **population you defined**, the same way, every run, and keeps an **evidence package**.

A technically correct rule applied to the wrong population produces false assurance.

## Authoritative sources (as stated today)

| Question | Trust this file |
|---|---|
| Who does GRC *think* is privileged? | `privileged_roster.csv` |
| Is MFA configured, which method, when last verified? | `auth_config.json` |
| Who has privileged access on a cloud account right now? | `cloud_entitlements.csv` |
| Is there a time-bound approved treatment? | `approved_exceptions.csv` |
| Which cloud accounts exist, and how are they tagged? | `cmdb_accounts.csv` |
| When were those files extracted? | `source_manifest.json` |

If two sources disagree, that disagreement **is** a finding. Do not "fix" the files to make them match.

## Review freeze

All monitor math uses:

`review_as_of` = `2026-09-15T12:00:00Z`

Do not use your laptop's current time.

## Stated evaluation rules for this Build

**Population (as tagged in this extract):** identities with `privileged = true`, `access_status = active`, and `environment_tag = production` on `cloud_entitlements.csv`. One evaluation per `user_id`.

Sandbox access is out of this population.

**Technical MFA requirement is met** only when all of these are true for the joined auth record:

- `mfa_enabled` is true
- `mfa_method` is `totp` or `hardware_key` (SMS does not meet the stated requirement)
- `mfa_last_verified` is present and not more than **90 days** before `review_as_of`

**PASS:** the technical MFA requirement is met.

**FAIL:** the technical requirement is not met, the identity is evaluable, and there is **no** valid approved exception.

**WATCH:** the technical requirement is **not** met, but a **valid** approved exception exists (`control_id = HL-CC-PRIV-MFA-01`, `status = open`, `expires_on` blank or on/after the freeze). An approved exception is a risk treatment. It is not a technical PASS.

**UNKNOWN:** evidence is missing, blank, conflicting, stale beyond 90 days, the identity is not on the privileged roster, or human MFA logic does not apply (service accounts and break-glass / shared emergency accounts).

Join keys: `user_id`. Normalize email to lowercase when you use it.

A closed or expired exception is not a valid treatment.

## Monitoring-assurance control (required)

Automated monitors need a control of their own. Each run must record:

- expected source systems
- source extraction timestamps
- expected population count (baseline)
- actual population count
- run timestamp
- run completion status

Harborline's stated assurance thresholds for this freeze. These are **Harborline scenario values for this exercise**. They are not industry-standard GRC or continuous-monitoring thresholds.

- Expected sources: the five files named in `source_manifest.json`
- A source is stale if `extracted_at` is more than **14 days** before `review_as_of`
- Baseline population count: **12** (last successful weekly job in `control_runs.csv`)
- Population variance above **25%** versus baseline is a monitoring-assurance failure. **25% is Harborline's scenario threshold for this exercise.**

Would you keep a 25% threshold in a real privileged-access environment? What factors would determine an appropriate threshold? Consider population size, privilege level, expected hiring and termination activity, system changes, acquisitions, source reliability, control frequency, and risk tolerance. There is not one correct percentage.

Unexpected variance, missing sources, stale extracts, or a failed run are **monitoring exceptions**. They are not the same thing as an MFA FAIL on a person. Put them in run metadata.

If someone asks "how do you know the monitor is working?", this control is the start of the answer.

## Evidence package

A monitoring result is not a screenshot. Each run should preserve enough that a reviewer can see:

- what population was evaluated
- which source files and extraction times were used
- when the monitor ran
- which rule version ran (`stated-v1-weekly` for this Build)
- what statuses were produced
- which untreated failures, approved treatments, and unknowns existed

**Standard minimum outputs (Python path, in Optional_Python_Path/outputs):**

- `run_metadata.json`
- `control_results.json`
- `control_health_summary.md`

**Also produce if you can (Deep Dive or Challenge):** `population_snapshot.json` and `exceptions.json` (FAIL and WATCH rows).

Spreadsheet path: equivalent tabs in the Master Workbook (Control_Results, Exceptions, Run_Metadata, and 3 Health Summary).

Changing a source record must change the evaluation.

## SOX, lightly

SOX ITGC work cares whether the **population is complete** and whether evidence is reliable enough to support a conclusion. This project is not a SOX course. Completeness and evidence quality still matter because Harborline's financial systems sit next to the same privileged cloud access.

## Privileged methods (stated list)

Accepted: `totp`, `hardware_key`. Not accepted as meeting the requirement: `sms`, `none`, blank.

## Why a monitor (spreadsheet or Python)

Spreadsheets can do this for a small freeze file. The skill is encoding the **same rules** so 2,500 identities would not change the logic: load → population → evaluate technical condition → apply treatment → assurance → evidence package.

Python is the recommended technical path. It is not required. Spreadsheet is a complete Build path and does not cap your score. See Stage 3 in START_HERE_Learner_Guide.pdf.