ARCHITECTURE OF GRACE

IT Deployment & Data Guide

Hosting, the optional Google Sheet sync, the two keys, and the public Voices wall.

READ BEFORE DEPLOYMENT Before enabling sync or distributing this to students, the district must complete its own privacy and legal review and execute a Student Data Privacy Agreement. **The write key described below is public by design** and is not a security control. This document describes **script version 11, released 5 September 2026**, and supersedes the June 2026 guide, which documented settings that no longer exist. It is not legal advice.

1. What this covers, and what changed since June

Architecture of Grace is a set of static web pages. There is no server to run, no database and no build step. By default every response is stored in the viewer's own browser and nothing leaves the device. A school may optionally enable sync, so completed work is appended to a Google Sheet **the district owns**. The site also includes an optional public comment wall ("Voices"). This guide covers all three.

What the June 2026 guide got wrong

The June guide said	What is actually true now
Edit `SCHOOL_SYNC_URL`, `SCHOOL_SYNC_KEY`, `ACCESS_CODE` and `VOICES_URL` near the top of the script section of `index.html`.	None of those settings exist. The site's destination lives in **`aog-sync-config.js`**, and a school normally enters its own values in the dashboard instead — no file editing at all.
One `PASSCODE`, default `sky`, embedded in the HTML.	**Two keys**, both in Apps Script **Script Properties**: `BACKEND_AUTH_KEY` (write only) and `ADMIN_PULL_KEY` (read). There is no default value to forget to change.
A read-back path that can send the passcode as a `?key=` URL parameter.	Removed. Reading is POST-only; the `GET` endpoint is status-only, so **no passcode travels in a URL or a browser history**.
One records tab, plus a Comments tab.	**Seven tabs**, each created automatically on first write: `Responses`, `DailyCheckins`, `SupportCheckins`, `ExitSlips`, `HomeObservations`, `HomeCheckins`, `Practice` — plus Comments if the Voices wall is on.
An unrecognised request was handled loosely.	An unrecognised `action` is **rejected and writes nothing**. Before version 11 it fell through to a screener write and put a garbage row in `Responses`.

2. Hosting

The site is static files, so it runs anywhere that serves HTML. Viewers' saved results live on their own devices and survive redeploys.

- **Netlify (simplest).** Sign in on an institutional account, open the site's Deploys tab, and drag the site folder onto the drop zone. To update later, drag the new folder onto the same site — the link stays the same and saved results are unaffected.
- **District-hosted or another static host.** Cloudflare Pages, GitHub Pages, or any district web server or intranet share. There is no runtime to install.

A service worker caches the site for offline use, so after an update ask staff to hard-refresh once.

3. Optional: collect results in one Google Sheet

> **THIS IS A GOVERNANCE DECISION, NOT A SETUP STEP** The site ships local-only by design. Turn sync on **only after** the district's privacy review is complete and its data-privacy agreement is signed. Once it is on, the district is operating a centralised collection of student wellbeing data.

Step 1 — Create the Sheet on an institutional account

On the **district's** Google Workspace account, create a new Google Sheet. Do not use a personal Gmail account: the FERPA school-official exception requires the agency's direct control, and a Sheet owned by an individual educator does not sit inside that control.

Step 2 — Paste the script

In that Sheet: **Extensions → Apps Script**. Delete any placeholder code, paste the whole of `AoG-Screener-Sync-Code.gs` (published alongside the site), and Save.

Step 3 — Set the two keys

In **Project Settings → Script properties**, add both of these. They must be different from one another.

Property	Value	What it is for
`BACKEND_AUTH_KEY`	A token you generate	Lets a device append a row. **This value gets published on your website**, so treat it as public.
`ADMIN_PULL_KEY`	A second, strong secret	Lets a teacher's dashboard read rows back. Never published. Without it, reading is off entirely.

Step 4 — Deploy as a Web App

Deploy → New deployment → Web app. Set *Execute as: Me* and *Who has access: Anyone*, then Deploy and approve the permission prompt. Copy the Web App URL — it ends in `/exec`.

> **UPDATING LATER** Use **Deploy → Manage deployments → pencil → Version: New version → Deploy.** That keeps the same `/exec` URL, so every classroom link and QR code you have already issued keeps working. Choosing *New deployment* instead mints a brand-new URL that must be pasted back everywhere.

To confirm which version is live, open the `/exec` URL in a browser. It returns a status object with the version number and date, and no student data — that endpoint has no data path.

4. Wiring the site to the Sheet

There are two ways to give the site a destination, and the second is the one to use for a district.

A — In the dashboard (recommended; no file editing)

A school opens the Educator Dashboard and goes to **Set up ▸ Syncing & distribution ▸ Connect your school's Sheet**, then enters its own `/exec` URL, its own write key, and its `ADMIN_PULL_KEY`. Every classroom link and QR code that dashboard then generates **carries the destination inside the link**. Nobody has to send anyone a key, and a destination on a link takes precedence over anything in the config file.

B — In `aog-sync-config.js` (a site-wide default)

The file next to `index.html` holds a default destination for the classes run by whoever publishes that site. It sets four things: the `/exec` URL, the write key, a human label, and `schools` — the list of school identifiers whose classroom links may use this destination.

> **SET `SCHOOLS` DELIBERATELY** `schools: ["*"]` means **any** classroom link may sync to this destination, whatever school identifier it carries. That is convenient for a single-teacher pilot and wrong for a district. Narrow it to your own identifiers — but only after your links have been regenerated so they carry their own destination, and after you have decoded one QR code and confirmed it. Narrowing it on the assumption that the links already carry a destination has silently turned syncing off for a whole fleet before.

This file is public: anyone can read it, and therefore anyone can read the write key in it. That is the design. See section 6 for exactly what that key can and cannot do.

5. How sync behaves, for support

- **Write.** A completed item POSTs one record plus the write key. The script appends **exactly one row** to the tab that matches the kind of record. No path writes more than one row per request.
- **The device is the source of truth.** Records are saved on-device first and re-sent if a send fails, so nothing is lost when the network drops.
- **Read-back.** In the dashboard, *Pull from sheet* requests rows over POST using `ADMIN_PULL_KEY` and merges them in, marked as coming from the sheet, without overwriting local records.
- **A bare URL stays private.** Sync turns on only when a destination is configured or the link carries managed parameters.

The nine actions

The script accepts nine actions and rejects anything else. Four write, five read; every read requires `ADMIN_PULL_KEY`.

Action	Direction	Tab
(default screener write)	write	`Responses`
`checkin`	write	`DailyCheckins`, or `SupportCheckins`, or `Practice` — routed by `checkinType`
`exitslip`	write	`ExitSlips`
`home`	write	`HomeObservations`
`homecheckin`	write	`HomeCheckins`
`pull`	read	`Responses`
`pullCheckins`	read	`DailyCheckins` + `SupportCheckins` + `Practice`
`pullExitSlips`	read	`ExitSlips`
`pullHome`	read	`HomeObservations`
`pullHomeCheckins`	read	`HomeCheckins`

> **FOR FUTURE EDITORS** New kinds of row are routed **inside an action the script already understands**, using the `checkinType` field — never by inventing a new verb. Version 11 rejects unknown actions, but a building still running an

older paste will fall through to a screener write and put a garbage row in `Responses`, so the rule stands until every deployment has updated.

Columns

Each tab is created on first write with its own header row. When a new field is added in a later version, the script appends the column **on the right**, never in the middle, so a Sheet that has been collecting since June keeps its existing columns and its existing rows stay aligned.

6. Security, stated plainly

What the public write key can do

NOT SOFTENED Anyone who reads `aog-sync-config.js` has the write key. With it they can append rows carrying real student identifiers, invented scores and tiers, and the two safety flags — and can post a check-in attributed to a named staff member with a follow-up marker set. **Forged rows are not distinguishable from real ones in a teacher's report.** What the key cannot do: read any row, alter an existing row, or delete one. Every write appends.

Rotate the write key by changing the Script Property and the value in `aog-sync-config.js` together. Rotating `ADMIN_PULL_KEY` only requires the Script Property and each teacher's dashboard.

What reading is, and is not, scoped to

The five read actions are each gated by `ADMIN_PULL_KEY`. That key is **not scoped**: a holder receives every row in that Sheet — every student, every class, every teacher — and `pullCheckins` also returns practice records and This Is Me pages.

DESIGN AROUND THE SHEET BOUNDARY One Sheet per boundary that matters. **If two schools must not see each other's data, give them two Sheets and two key pairs.** The product has no per-user login and no row-level scoping; two logins against one Sheet is not something it can do. Plan the deployment around this rather than discovering it later.

The dashboard code

The results dashboard is gated by a configurable code that performs no encryption and does not restrict anyone who can open the page on that device. It is a convenience gate. A setting that needs authenticated, per-user access needs a hosted version behind district SSO, not this one.

What release 2.0 hardened (version 11, 5 September 2026)

Three findings from an outside IT and privacy review were closed:

- **Formula-injection escape.** Every string written to any tab is escaped, so a submitted value beginning with `=`, `+`, `-` or `@` can no longer become a live formula for whoever opens the Sheet. Previously that was an exfiltration path if the published write key leaked. The escape is a format prefix, not data: values read back through every pull path exactly as sent.
- **Size caps.** A body over 64 KB is rejected before parsing — a real submission is under 10 KB — and any single string field over 2,000 characters is truncated.
- **Strict action whitelist and rate limit.** Unknown actions are rejected. Each key is limited to 300 posts per minute; a whole grade finishing at once is far inside that, a flood from a leaked key is not. The limiter **fails open**, so a student's row is never the thing that gets dropped.

Nothing about the payload contract changed — same field names, keys, tabs, column order and response shapes. Pages already in the field keep working unmodified.

7. Upgrading an existing deployment

Paste the current `AoG-Screener-Sync-Code.gs` over your existing `Code.gs`, then **Deploy ▸ Manage deployments ▸ pencil ▸ Version: New version ▸ Deploy.** Same `/exec` URL. Nothing in your Sheet is touched, and every link and QR code you have issued keeps working.

Every reply carries an integer version. Nothing is ever gated on it — an old script keeps working in both directions, an unknown field is ignored and a missing one is written blank. A reply with no version at all means version 5 or older, which is expected and not an error.

Version	When	What it added
1	Jun 2026	The first sync — a **different file**, one passcode for read and write. Superseded; do not run it.
2	Aug 2026	`Responses` plus the `ADMIN_PULL_KEY` read path — the two-key model
3	Aug 2026	`DailyCheckins` and `pullCheckins`
4	Aug 2026	`ExitSlips` and `pullExitSlips`
5	Aug 2026	`HomeObservations`, forced text columns, an append lock
6	Aug 2026	Reports its own version
7	Aug 2026	`HomeCheckins`
8	Aug 2026	`SupportCheckins` — the adult team check-in stops sharing a tab with the student's own check-in
9	Sep 2026	The `Practice` tab becomes readable, through the existing `pullCheckins`
10	Sep 2026	Percentage-independent on `Practice`
11	**5 Sep 2026**	**Release 2.0 — the hardening pass. Current.**

8. Scaling to a district

A single Sheet suits one school. Across a district, a separate Sheet, script and key per building becomes the bottleneck. `AoG-District-Directory-Sync.gs` and `aog-identity.js` turn that into a hub-and-spoke model that IT stands up once: **one Apps Script Web App, one Sheet, one Directory tab** populated from the authoritative source — an SIS export, OneRoster, Clever or ClassLink — never by hand. Each completed item is routed to a per-school tab created automatically on first write. The privacy model, the two-key contract and the device-first behaviour in sections 3 to 6 are unchanged.

Student identity is handled without accounts: the roster mints one opaque token per student, and the link carries only the token, which resolves on-device to the anonymised code. Google Workspace SSO is available and is **off unless a district turns it on**; enabling it stores verified student emails in the roster, which is a policy decision that must be reflected in the data-privacy agreement first.

> **READ SCOPE APPLIES HERE TOO** The hub model concentrates more data behind one `ADMIN_PULL_KEY`, and that key is still unscoped. Decide who holds it before the first roster import, not after.

9. Before you enable sync: the consent question

Enabling sync is a data decision. Administering the check-in is a **consent** decision, and they are separate.

Architecture of Grace's position is that the check-in layer should run on written parental consent obtained in advance, with the questions attached, rather than on notice and an opt-out — following the Department of Education Student Privacy Policy Office's Dear Colleague Letter of 26 August 2026. That guidance is contested, and the Data & Privacy Statement sets out why the product follows it anyway. A consent packet is published alongside this guide in English and Spanish.

The curriculum and the academic practice pages are not surveys, collect nothing about a child's inner state, and are unaffected by any of this. A district can adopt the teaching material without adopting the check-in layer — that separation is a property of the software, and it is worth preserving in how the district describes and records the adoption.

10. The Voices comment wall

The wall is controlled independently of check-in sync. Posts go to a separate Comments tab in the same Sheet and appear publicly on the page. Both the page and the script reject email addresses, telephone numbers and hyperlinks, filter common profanity, and cap length. It is **open, not pre-moderated**: an accepted post appears immediately and is removed afterwards by whoever operates the Sheet. A district should decide deliberately whether to run it at all, and name who watches it while it is live.

Appendix — where the files are

File	What it is
`AoG-Screener-Sync-Code.gs`	The Apps Script receiver. Paste this into the Sheet. Current version 11.
`aog-sync-config.js`	The site-wide default destination: `/exec` URL, write key, label, and the `schools` list.
`AoG-District-Directory-Sync.gs`	The district hub — a superset of the above, backward-compatible with per-school deployments.
`aog-identity.js`	The token-based student identity layer. No accounts, no emails, unless SSO is turned on.

The script file is the authority for anything in this guide. Where this document and the script disagree, the script is right and this document is stale — the version banner at the top of the script names the release it belongs to.

Version 2.0. Written 6 September 2026 against sync script version 11 (5 September 2026). Supersedes the June 2026 guide in full. Not legal advice. · Technical contact: Architecture of Grace, Theopus77@yahoo.com.