# OpenEMR Weekly Meeting Detailed Summary

November 17, 2026

---

**Attendees:** Brady Miller, Stephen Waite, Stephen Nielson, David Eschelbacher, Eric Stern (OCE), Jerry Padgett, Simon Quigley (Altispeed), Jit Chawla (ACE Health Solutions), Prabhu Pathak (LogiCoy), Ansuman Nayak (LogiCoy), Sherwin Gaddis (Tarevo), Pedro Camellon. Not attending: Seth C, Michael Smith (OCE), Farjad Zaghi (Nefesh).

> *Italicized indented passages are background, clarification, and commentary added after the meeting, not statements made on the call. Items from the meeting chat are folded in where they belong. "Stephen" alone means Stephen Nielson; Stephen Waite is named in full.*

- **Shape of the meeting:** Stephen chaired, working largely from agenda items David had proposed: an update on Stephen's international client project, the module system, the front controller, project priorities and roadmaps, and David's Visit List PR and data importer proposal. Eric's Docker and module plan led into a long discussion of the front controller and of multisite, which Sherwin runs at a scale the group hadn't heard of before. Brady then demonstrated his PR-shepherding process live on David's Visit List PR. Project priorities and the data importer were deferred, and the call ran past the hour.
- **Simon's PRs:** a batch of Simon's PRs was merged last week; seven remain open and 18 have been closed. Simon confirmed seven is right and thanked those who merged them.

## International Project Update and Clinical Informatics

- **The project:** since February, Stephen Nielson has been building an HL7 messaging and referral interoperability layer for a national health service abroad. He is still working out how much can be upstreamed, and needs to review what Jerry has done with the FHIR APIs and transactions. The project used little of OpenEMR's existing transaction and referral features, because the country's data models are so specific.
- **Already upstreamed:** work that lets the patient finder and some reports be extended with additional identifiers, driven by a national identifier with encryption and display requirements. Some of it could help US sites with Social Security numbers — masking them, or permission-gating whether an SSN is included in a CSV export.
- **Next: regional packs:** after this project, the next big piece is regional packs for language and jurisdiction. Installing a pack would apply the right globals, languages, and available code sets for a region.
- **A clinical informatics group:** there is broader interest in forming one for OpenEMR, like the clinical groups in openEHR and OpenMRS, to discuss how data are mapped clinically, along with permissions and safety. Stephen's example: some FHIR contributions passed Inferno, which checked that numbers and strings were sent correctly, but may have put values in the wrong fields. His client would like to contribute and collaborate. Anyone interested should reach out to Stephen.

> *Context: many national health systems run a central messaging broker for referrals, discharge summaries, and results, usually on HL7 v2, alongside a national patient identifier whose use is limited by law. That combination explains both halves of Stephen's work: a messaging layer that fits the national broker, and identifier fields with display and access rules that OpenEMR's US-shaped patient model didn't have.*
>
> *On SSNs: masking all but the last four digits on screen and gating exports by permission are cheap ways to apply HIPAA's minimum-necessary principle to one of the most damaging identifiers in a breach. A permission check on CSV export also closes a common gap, since bulk exports often bypass the screen-level controls.*
>
> *Regional packs have a close model in OpenMRS. Its Initializer module ("Iniz") loads a deployment's configuration — concepts, locations, global properties, and more — from CSV and JSON files in a configuration folder, and OpenMRS bundles such folders into shareable content packages for a distribution. OpenEMR already has the pieces such a pack would combine: translations, globals, and code-set loading through External Data Loads.*
>
> *On the clinical group: openEHR's Clinical Program, led by a Clinical Program Board, has clinicians and domain experts review each clinical model online in its Clinical Knowledge Manager before release. Stephen's Inferno example is the reason for such a group. Conformance tests can confirm that a resource matches its profile, but not that a value came from the right source field. A natural first task would be reviewing the FHIR mappings behind the US Core resources Jerry is now making writable.*

## Modules, Docker, and the Front Controller

### Eric's Docker Plan — Issue 15359

**Issue:** https://github.com/openemr/openemr/issues/15359 (by Firehed — Eric)

- **Where things stand:** Eric hasn't been able to focus on core OpenEMR much lately, but still tries to fit in improvements. He hopes to make progress on modules and especially the front controller — standardizing on them, using the new documentation for agents and contributors, and starting to deprecate older code paths that cause security issues.
- **Yesterday's work:** he revisited changes he began in April toward a single source of truth for Docker. Besides simplifying the Docker stack, the goal is to actually use the front controller and the dependency-injection container, and to have a standard way to manage modules. Stephen had pointed out last week that there is no good story for installing modules in the official Docker image. Eric posted the plan as issue 15359; the checklist is mostly AI-generated, but the plan is his.
- **Option 1 — build on the official image:** operators write their own Dockerfile that starts from the official image and runs a composer command to require the modules they need, then build a custom image. If the project wants a read-only file system in Docker — a big security win — customizing the image at build time is unavoidable, and installing modules at runtime through an admin screen is incompatible with that model. Eric sees this as the more straightforward plan.
- **Option 2 — OpenEMR as a Composer project:** a longer-term idea Michael has raised several times. You would install OpenEMR itself through Composer, so a site's whole code base is basically two files, composer.json and composer.lock, listing OpenEMR and its modules; a tool could then generate the Dockerfile. Probably better in the long run, but a heavy lift: it is incompatible with all the existing modules, which serve directly from OpenEMR's document root.

> *Issue 15359, opened November 16, proposes replacing today's several Docker images — each with a startup script of over 1,000 lines — with a single image for development and production. It would be built from the checked-out source, run as non-root on a read-only file system with only /tmp and the site volume writable, and have no startup script: PHP-FPM behind an nginx stage that serves static assets and routes requests through the front controller. Install and upgrade would become explicit one-shot jobs with a readiness check, and the old images would be retired only after a deprecation period with a migration guide. Eric's comment after the call says the group "basically confirmed that option 1 is the only sensible short-term implementation option," while option 2 depends on fully replacing the current module system.*
>
> *A read-only root file system is a standard hardening control: the CIS Docker Benchmark recommends treating the container's root file system as a "golden image" mounted read-only, so nothing can be modified at runtime. That is why Eric's comment rejects a module-install screen inside the running application: anything that can write code at runtime defeats the control.*

### Static Assets and Writable Files

- **Obstacles to option 2:** one participant favored the composer.json approach, but noted problems today with the sites directory, and that some modules have writable files inside their own folders. Everything that stores documents would also need to be on Flysystem first.
- **Eric on running at scale:** writable directories and temp files need to be standardized so Docker Compose can handle them. Without the Redis-based sessions you need sticky sessions, which cause problems. The project has fixed most places that depended on repeated requests reaching the same server, though a few may linger, and outside modules are unknown.
- **Which modules write to their own folders?** one participant didn't see why any module would: a correct module writes only to temp or to a writable directory under documents. The harder case is Kubernetes: you can't run Composer in one pod and not the others, so adding a module becomes a rollout. Eric agreed that running Composer in a live pod is "fighting the system": the Docker image is what goes to production, with as little as possible depending on the runtime. The Flysystem work serves the same goal, since documents can then live in S3 or similar services, CouchDB, or a NAS without OpenEMR caring which.
- **The real problem — static assets:** today's modules install with Composer plus a custom installer on top, which works. The difficulty is that they package static assets in a directory that must be web-readable and sits alongside application code, which is where much of the security risk comes from. In the next-generation design, a module declares its static-asset directory, and those files are copied at install or Docker build time into a safe web directory away from code. The comparison to Symfony's asset bundling came up, and Eric thought the project might be able to use Symfony's component.
- **An interim bridge:** Stephen suggested a small script or installer that takes a site's composer.json and generates the Dockerfile it needs today. Eric said that is pretty much how he envisions it. David's question was answered: he wanted a general update on the plans.

> *Symfony's version of this is its assets:install command, which copies each bundle's public files into the application's public directory under bundles/, so only those copies are ever served. Eric's design does the same for OpenEMR modules, and issue 15359 adds a single audited list of static file types that drives both the nginx configuration and the export, so what may be served is defined in one place.*
>
> *On sessions: OpenEMR's Docker images have accepted a REDIS_SERVER setting for PHP sessions for years, and 15359 proposes configuring Redis sessions from a single setting. Shared sessions are what make it safe to run more than one container behind a load balancer; sticky sessions instead pin each user to one container and break when that container is replaced.*

### The Front Controller and Legacy Modules

- **The dependency order:** Eric explained that the front controller must be in place to unlock the next module system, and running it is much easier in the standardized Docker image. But it has difficulties with existing modules that serve static content. The current front controller rewrote, as a PHP compatibility layer, the .htaccess rules of the modules that ship with OpenEMR — and there is no good way to do that for third-party modules, short of telling sites to customize their own nginx configuration.
- **Breaking third-party modules:** it was noted that moving everything to the front controller would break how third-party modules serve their files. Eric suggested a best-effort allowlist of safe file types — images, maybe PDFs — starting conservative and adding types as modules need them. He doesn't love it as a long-term system, but dropping support for every existing module isn't an option.
- **A cutoff:** one participant said supporting legacy modules indefinitely won't be possible. The project needs a cutoff and a definition of what a module is and how it integrates; third parties will have to adapt. Eric agreed it needs a time cap. In semantic-versioning terms this would be OpenEMR 9.0: add the new system, encourage migration, deprecate the old one while it keeps working, and announce when it will be retired. Migration is easier now with agentic coding than a year ago.
- **Chicken and egg:** Brady framed step one as a module system that works with the front controller, so people can build compatible modules now; then move to the front controller; then tell people the old modules must change. Can modules be built to work with both? Stephen suggested a middle ground: carve the modules folder out of the front controller and serve everything else through it, while modules migrate to connecting through events and hooks and serving their views through endpoints. Then, after six months or so, cut it off.
- **Closer than thought:** re-reading the router code during the call, Eric found it already has a pass-through list of asset types — PNG, SVG, CSS, JS, and so on — that can be expanded. The .htaccess translation was only for places that defined their own front controller, such as the API and OAuth2 endpoints, which became the backwards-compatibility router. In theory, supporting a module with its own front controller is one line in the fallback router's rewrite logic, or perhaps a line read from the module's composer.json. “So this may be closer than I thought.”
- **Why so conservative:** the project has been cautious about pushing the front controller because of multisite and installs in a sub-path. Eric: both could be worked around, but if they are going away anyway it is much cleaner to avoid them from the start, which is part of why the Docker route matters. Someone added that since squads were removed, multisite could go too, and thought the project had already agreed to that. Eric noted the front controller and its DI container technically support multisite, but the site would come from an environment variable rather than being auto-detected.
- **Deprecation mode:** one more change is wanted first: the front controller hard-codes PHP deprecation notices to fatal errors, which Jerry caught in the architecture review. That belongs only in development. Eric confirmed it: globals.php has the same setting as an opt-in environment variable, and making the front controller match is about a three-line change. If the goal is to get traffic through the front controller soon, he agreed it should be environment-based.

> *The approach Stephen and Brady described is the strangler-fig pattern that the architecture documents (PR 14865) adopt as their migration strategy: route new behavior through the new path while the old path keeps working, then retire the old path once nothing depends on it. Eric's finding narrows the risk — if third-party assets already pass through, the remaining work is mostly modules that run their own front controller, and each needs only a declared rule.*
>
> *A published deprecation timeline is what makes the cutoff fair. Symfony, for example, deprecates in one release with runtime warnings and removes only in the next major version. For OpenEMR, the warnings could appear in the module manager and the logs when a module uses the old path, so site owners learn of it before anything breaks. Turning deprecations into fatal errors in development, but never in production, is the usual way to find incompatibilities early without breaking a working site over a harmless notice.*

### Multisite at Scale

- **Sherwin's case:** he runs 40 sites on one server with multisite. What does “flattening” mean — 40 copies of OpenEMR? The answer: 40 databases and 40 installations. They wouldn't need 40 database servers, but each site keeps its own database; in fact sites already have separate databases. One participant said they had hit MySQL's practical limits the hard way: with about 300 tables per site, tens of thousands of tables on one server is more than MySQL handles comfortably.
- **Stephen:** months ago the group had a long conversation about security issues with sites and proposed one Docker image running as many containers — 40 for Sherwin — optionally sharing one MySQL server. That gives stronger guarantees against one site taking down the others or cross-site contamination, at the cost of operational complexity. At the time there was little data from large multisite users, so he wants Sherwin's operational feedback.
- **Sherwin's setup:** he re-engineered multisite for his clients to address problems he found with it. He consolidated many practices onto one server, and splitting them back out means 40 separate upgrades. Others replied that many containers can run on one server, and that scaling is a single number in the Docker Compose file — though Eric agreed there is real overhead in any high-tenancy deployment.
- **Eric's middle ground:** nginx can pass an environment variable to PHP, so with path-based routing a single nginx in front could set the site ID for each path. Instead of 40 deployments, it would be 40 blocks in one nginx configuration — more or less what multisite does now — while keeping a single deploy. OAuth and the APIs still raise multisite questions.
- **Next step:** Stephen is open to revisiting the decision, and asked Sherwin to come next week for 15 to 25 minutes to explain his customizations, the problems he has hit, and what would help, with anything he can post to issue 15359 beforehand. Sherwin had already sent Stephen his multisite README.

> *On the table limit: MySQL has no fixed cap on the number of tables, but every open table uses table-cache entries and, with InnoDB's file-per-table storage, a file handle. Forty sites with about 300 tables each is roughly 12,000 tables, enough to strain the table cache, the table-definition cache, and the operating system's open-files limit unless they are tuned.*
>
> *The trade-off is isolation against overhead. Multisite shares code and upgrades but runs every practice in one PHP process space, so a bug or compromise in one site's path can reach the others. One container per site isolates them at the cost of 40 processes to run and patch, though from a single image. Eric's nginx approach keeps one code base and one deploy while making the site explicit instead of guessed — the change 15359 asks for when it calls for unifying multisite detection — but it doesn't add process isolation. When the 40 sites are separate practices, each a separate covered entity, isolation carries extra weight.*

## Brady's PR Shepherding Demo

### How the Process Works

- **The goal:** review PRs systematically and find the ones that are actually ready, instead of clicking into each one. It is hard for a person to see from a PR page whether everything is ready. Open PRs are down from about 320 at one point to roughly 270.
- **The steps:** you choose a scope — an author, or a subject such as FHIR. The AI goes through the PRs and checks readiness: whether the CodeRabbit review has been addressed and CI is passing, allowing for known flaky tests. PRs that aren't ready are put on hold, waiting on the author. For ready ones, it offers a “phase two” brief — a quick review rather than a full one, since full reviews use a lot of tokens, unless CodeRabbit never reviewed the PR. A maintainer then looks at the PR and decides.
- **Where it lives:** a short set of documents on PR shepherding in the repository's docs directory, deliberately not very specific, and revised as the process evolves. Brady uses two commands: “take stock,” which surveys the PRs and analyzes them or refreshes the analysis, and “show dashboard,” which shows the current state without new analysis. It isn't packaged as a Claude skill: he wants it to work with any AI.
- **The briefs:** each review produces a markdown brief that Brady reads in PhpStorm: the commits, a verdict, the CodeRabbit findings, an assessment, a baseline, follow-ups, and a recommendation. Verdicts are marked green, a yellow diamond for approval with minor issues, or red for not ready. The real value is in large batches — authors with 10 or 15 PRs, or a subject like FHIR — where it picks out the straightforward fixes that just need eyes.
- **Live on David's PR:** run against David's Visit List PR (15312), the process found one PR and recommended approval with a minor issue: CodeRabbit had paused, so it hadn't reviewed the latest changes. David commented “@coderabbitai resume” during the call; afterward, CodeRabbit reported its review finished.
- **Why CodeRabbit is hard to read by hand:** each CodeRabbit review is incremental and may report success for the last increment without saying whether earlier comments were addressed. Many contributors don't resolve threads, and “Rabbit always says yes” — it accepts almost any reply. The shepherding process checks whether a finding was actually addressed, including when someone simply declined to fix a major issue. One participant had seen CodeRabbit delete an item entirely when a commit was pushed before the reply was posted.
- **Next for the queue:** Brady keeps separate sessions by topic — FHIR, Simon's PRs, Jerry's. He is thinking of working through the oldest PRs chronologically and closing those that can't be merged.

> *CodeRabbit pauses automatic incremental reviews by default after five reviewed commits, and “@coderabbitai resume” restarts them; for a PR whose last commit was already recorded as reviewed, CodeRabbit's documentation recommends “@coderabbitai full review” instead. That explains David's PR: a completed full review, small follow-up changes, then an automatic pause.*
>
> *Brady's check for whether findings were actually addressed fills the gap that “Require conversation resolution” can't (10/27): resolution records that someone clicked a button, not that the code changed. A brief that names each unaddressed finding gives the maintainer the same answer in a minute.*

### Skills, Costs, and Local Models

- **Why not a Claude skill?** asked whether the process was packaged as a skill, Brady said he didn't want it tied to Claude. He expects subscription costs to rise: heavy use already pushes him past the lower monthly plan, and beyond the plans, per-token API pricing gets expensive quickly, especially with several subagents running at once. More affordable options exist, and he doesn't want to be locked in.
- **In the chat:** Pedro said a skill would make a lot of sense for these reviews, as part of the repo, and proposed splitting the current AGENTS.md into more specific skills. Stephen replied that the project is breaking much of AGENTS.md into the architecture context to minimize context usage (PR 14865), that smaller sub-skills seem like a good idea too, and that skills can be shared among different agents. He also said he was bringing in the architecture PR.
- **Local models:** Pedro noted that a GPT model had become much cheaper. Stephen is planning a $10,000 to $15,000 local system — at least 128 GB of VRAM, hoping for 256 GB — and is weighing the trade-offs as a business-continuity risk. Eric said Michael does some local work; his own next upgrade would run $6,500 or more. Pedro thinks at least 128 GB is needed to be useful, and is "100 in" on sovereign AI. Stephen: many countries' jurisdictions are going that way.

> *Pedro's suggestion and Brady's concern are compatible. Agent Skills became an open standard at agentskills.io on December 18, 2025, and the same SKILL.md folder is now read by Claude Code, OpenAI Codex, GitHub Copilot, Cursor, Gemini CLI, and many other tools. A shepherding skill in the repository would therefore not lock the project to Claude, and because a skill loads only when relevant, it also serves Stephen's goal of keeping context small.*
>
> *PR 14865, the architecture documents, splits the guidance so that agents "load only the slice relevant to a task rather than one ~1,600-line file," and points AGENTS.md at docs/architecture/ with a trigger list. It includes the front-controller and strangler-fig migration guides discussed above.*

## David's Visit List PR

- **What it does:** a visit list that lives in the calendar. A provider sets a block, like an office or vacation block — say 9 to 1 — and patients in it are seen in order of arrival rather than at set times, while the rest of the calendar works normally; a patient scheduled after lunch still appears as usual. It adds a new provider category and a new visit category. This suits practices that see patients first-come, first-served, as is common in some countries; Puerto Rico was mentioned.
- **Reaction:** positive. One participant said people have been asking for this kind of thing forever, and that putting the list inside the calendar is smart, because there is no separate list to maintain. David recalled that the weekly call helped him design it a few years ago.
- **Scope:** this first PR covers the day view. In week and month views the block shows, but not the list. David plans to add more in later PRs, following the advice to break work into pieces. One view was that the list belongs in the day view anyway — that is where staff would work through it — though week and month might show the block with a patient count.
- **Does it scroll?** with ten patients, does the block scroll? David hadn't tested that case yet. In the chat, Pedro asked whether it should scroll or enlarge the section, calling it a UX concern but not a blocker: better to get the feature in.
- **Encounters on arrival:** Stephen asked whether checking a patient in still creates an encounter. David: they are regular appointments, so they should do everything regular appointments do. A test patient marked as arrived had no encounter, most likely because the test data was inserted directly into the database rather than through the appointment pop-up, which fires the events that create encounters. Others suggested testing arrival through the pop-up and the flow board for two or three patients. David tested during the call: marking a patient arrived showed a pop-up saying a new encounter was created.
- **Other notes:** every patient in the list carries the block's start time — 9 a.m. — which is fine because they are separate patients. Encounter times are currently set by day and probably should use the actual time. List patients also appear on the flow board, though perhaps not yet in the right order. It was suggested that David turn his issue into an epic with phases. David said that with AI he built this in about a day; without it, it would have taken months.

> *Background: David proposed this on the forum on June 11, 2022, after it was discussed on a weekly call — a "Patient List" provider-event category whose block would display an orderable, untimed list of patients, aimed at urgent care, informal scheduling, and home visits. A reply then suggested walk-ins use the Flow Board instead. The new PR combines the two: the list lives in the calendar, and its patients appear on the flow board.*
>
> *On encounters: the calendar global "Auto-Create New Encounters" defaults to one encounter per day, with "one encounter for each appointment" as the alternative, and creates the encounter when the patient is checked in. Long-standing forum reports note it fires only on the appointment's own date. For visit lists, one per day is fine for different patients; a patient seen twice in one day would need the per-appointment setting.*

## Jerry's Referral and Write-Audit Work

- **Referral:** Jerry is pretty much done. One bug remains — it looked only at Direct when it should let you choose the recipient — and he is fixing it. He considered a FHIR-based referral (heard as “need referral”) but is leaving that for the future, waiting to see what Stephen does with his international project. The tests are written and it seems to be working.
- **Write audits:** he has finished and tested the entire write-audit epic. Part of the transcript here is garbled, but he appears to have decided not to make the final cutover — enforcing the new patient-permission flow at the endpoint — the default yet. Sites can turn it on, and it might be turned on by default in the next release. He asked for objections; there were none.
- **Next time:** Stephen suggested focusing next week on Jerry's work. Jerry has no more PRs coming for now, but needs the group's confirmation on a few things.

> *This completes the three-phase audit epic described on 10/6 and 11/10. Shipping the enforcement off by default, with the plan to switch it on in a later release, is the safe way to change behavior that patient-facing apps depend on, provided the release notes announce the switch in advance so app developers can test against it.*

## Housekeeping and Next Week

- **Wiki access:** Jit couldn't reset his wiki password to post his entry. Stephen Waite will set a password and send it to him.
- **Longer calls?** one participant suggested extending the calls to two hours.
- **Next week:** multisite with Sherwin; Jerry's work; and project priorities and roadmaps, deferred from today, where e-prescribing is one topic to discuss. David's data importer proposal was not reached.

## Items Discussed and Proposed

*Recorded as topics raised and directions favored, not as assigned work or commitments.*

| Area | Discussed / Proposed |
|---|---|
| **Client project** | HL7 messaging and referral layer for a national health service; identifier extensions upstreamed; regional packs next. |
| **Clinical informatics** | Interest in forming a group to review clinical data mapping, permissions, and safety. |
| **Docker plan** | Single hardened image proposed (issue 15359). Option 1, modules added in a derived image, favored for now. |
| **Module assets** | Next-generation modules to declare static assets, copied to a safe web directory at build time. |
| **Front controller** | Asset pass-through already exists; legacy modules need a migration path and a cutoff date. |
| **Deprecation errors** | Front controller to treat deprecations as fatal only in development, via an environment variable. |
| **Multisite** | Large-scale use raised by Sherwin; decision to drop multisite to be revisited. |
| **PR shepherding** | Brady demonstrated an AI process that triages PRs for readiness and writes review briefs. |
| **Skills and models** | Repo-based skills suggested; local models raised as a cost and continuity hedge. |
| **Visit List PR** | Well received; day view first. Scrolling and encounter creation to be checked. |
| **Referral** | Nearly complete; one recipient-selection bug remaining. |
| **Write audits** | Epic complete; final enforcement left for sites to enable. |

## Future Discussion and Next Meeting Agenda

*Proposed for upcoming meetings, drawn from this week's discussion, chat, and notes; not a fixed agenda.*

| Item | Details |
|---|---|
| **Multisite review** | Sherwin's 40-site setup and customizations; per-site containers versus nginx routing. |
| **Jerry's work** | Referral, write audits, and the confirmations he needs. |
| **Priorities and roadmaps** | A ranked list of project priorities, e-prescribing included; a wiki page to track them. |
| **Data importer** | David's proposal, deferred this week. |
| **Module migration** | Definition of a module, timeline for legacy modules, the fallback-router approach. |
| **Docker plan** | Next steps on issue 15359; documentation for derived images. |
| **Clinical informatics** | Who is interested; a first review task. |
| **Regional packs** | Scope and format, and what can be upstreamed. |
| **PR shepherding** | Whether to package the docs as a portable skill; closing stale PRs. |
| **Visit List** | Week and month views, scrolling, list view. |
| **Carried over** | HTI-5 cleanup plan; HTI-4 deadlines; telemetry publication; flaky tests; PRs 8865 and 14254. |

## Notes

- **The connecting thread:** making deployment and contribution repeatable. A single hardened Docker image, modules added at build time, a front controller with explicit routing rules, regional packs, and Brady's written shepherding process all replace steps that depend on someone's memory or a hand-tuned server with something written down and reproducible.
- **Multisite is not a dead issue:** the plan to drop multisite assumed few sites used it at scale. Sherwin's 40 sites change that assumption, and Eric's nginx approach suggests the project may not have to choose between security and his deployment model.
- **The agenda list worked:** proposing items ahead of time gave the call a structure. The two items that ran out of time — priorities and roadmaps, and the data importer — move to next week.
- Ways to help: if you run OpenEMR multisite, especially at scale, post your setup and concerns on issue 15359 before next week; if you have clinical informatics experience, contact Stephen about the proposed group; and if you maintain a third-party module, check where it serves its static files and whether it writes inside its own folder. Comments on the forum or at the weekly call are welcome.