

OpenEMR Weekly Meeting Detailed Summary

September 29, 2026

Attendees: Stephen Nielson, Stephen Waite, Eric Stern (OCE), Sherwin Gaddis, Simon Quigley, David Eschelbacher. Brady Miller and Michael Smith absent.

Italicized indented passages are background and clarification added after the meeting, not statements made on the call.

The Pull Request Backlog

- **Open PRs rose from 206 to 265 in one week**, and that was after nearly 20 were cleared the previous day. Stephen Nielson has been focusing on PR review while Brady concentrates on security work.
- **Progress:** all the passing Dependabot PRs were merged, along with as many straightforward ones as possible. Simpler PRs are next; others need a finer-toothed review. Simon Quigley is now up to eight open PRs.
- **Not unique to OpenEMR.** Dependabot changelogs show the same influx across open source generally — anything with traction is being hit.
- **The worry voiced:** at 50 to 100 new PRs a week, contributors lose interest if their work sits. You have to engage people while they are engaged.

The Harder Question Is Whether to Accept, Not Whether It Works

- **Two separate problems, identified by Eric Stern:** first, do we want this in OpenEMR at all; second, is this the right implementation. AI can help substantially with the second. The first requires judgment about project direction, and is the harder one at scale.
- **Verbose PR descriptions are an obstacle.** Reviewers increasingly skip lengthy AI-generated summaries and go straight to the code.
- **Quality has been good from recent regular contributors** — Simon Quigley and a contributor named Marcelo were mentioned specifically, with interest in having Marcelo join a call. FHIR-related PRs are the exception: they need deep context and careful thought about whether and when to bring them in.
- **Triage by commit type helps a little.** Dependency updates can mostly be applied if nothing breaks; fixes are probably safe; refactors need more attention; features need the “is this what we should be doing” conversation. The caveat: a contributor can label a feature as a fix.

This relies on the Conventional Commits convention, in which each commit message begins with a type such as feat, fix, refactor, docs, or chore. OpenEMR's commit hooks and CodeRabbit already expect it, so the type prefix is available as a sorting signal essentially for free.

Ideas for Scaling Review

- **Shared AI skills in the repository.** OpenEMR has no .claude directory yet. Committing project-specific review skills would let anyone run a review that checks alignment with project goals and contribution guidelines, not just code correctness — and would ease the load on those with merge rights even if it does not add reviewers. Stephen Nielson has skills he could contribute, some of which are security-sensitive and would need care.
- **A technical committee** to prioritize PRs and work them on a schedule, in the way the security backlog is being handled. Counterpoint: even five people cannot keep pace with 100 PRs a week without AI automation.
- **A proposed approval path:** once a PR passes all checks and CodeRabbit is satisfied, a domain expert approves it, and Brady does a final sweep — he has reviewed essentially everything that entered the code base for about 20 years and remains keen to.
- **OCE's internal practice, “PR tennis”:** a reviewer sends back fixes, the author responds, and it goes back and forth until everyone is satisfied. That works for internal tools with shared context but transfers less well to deciding what belongs in OpenEMR.
- **Draft PRs:** useful for pushing work before it is review-ready, but CodeRabbit does not review drafts by default, and it is unclear how much of CI runs on them. CodeRabbit can be triggered manually with a comment.
- Half-serious suggestion that a bot automatically close any PR stating the code has not been tested.

- **Expanding the set of people with review and merge responsibility** was raised as the other option, with caution expressed about moving too quickly.
- **An unsolved problem with AI-authored PRs:** a PR authored by your own AI agent is submitted under your name, so you cannot formally approve it — only comment. Stephen Nielson runs his AI under a separate GitHub account and reviews under his own, but is uneasy about the AI being able to switch between the two.
- **Shared AI sessions:** asked whether teams share Claude Code sessions so others can see where work stands. It is believed to be supported, though OCE does not use it.

FHIR Write APIs and Jerry's SMART App

- **The question raised:** Jerry's FHIR SMART app is the first to use FHIR write operations. FHIR apps have been on the roadmap; how many exist, and is this a first test case?
- **Existing SMART on FHIR use:** a number of modules use SMART on FHIR, and the EHI export module in core uses the SMART mechanism internally. All of these exercise read access only.
- **Eric Stern:** having the full suite of FHIR APIs, including writes, is absolutely something the project wants, and writes have historically lagged. Chris started the work some time ago and Jerry has recently taken it over; he would like to see it land so others can build on it.
- **Stephen Nielson has not yet reviewed Jerry's work** given everything else in flight, but he and Jerry have discussed the concerns at length and Jerry is aware of them. Jerry's write focus is understood to be questionnaires.

Why Writes Are Different

- **Writes substantially increase risk.** This is why Stephen Nielson has been hesitant for a long time until it can be architected properly. Opening write access expands the attack surface and creates legal liability if data is tampered with, and a much higher risk of harm to the patient.
- **The legal context:** the clinician is legally liable for the designated record set. Data cannot simply be written straight into the patient record; there is a distinction between confirmed and unconfirmed data, with a permission model around it.
- **Current position:** writes are allowed for trusted confidential and system apps — where there has been administrative approval for the app to act on behalf of clinicians. Patient-context writes are the main concern.
- **Eric's pushback:** OpenEMR already handles this, just not over FHIR. Does moving it to FHIR change the landscape? Response: yes, because OpenEMR does not currently allow patients to write at all through the API — only authorized clinician users can.
- **Even clinician consent is imperfect:** clinicians tend to approve anything that makes their job faster, which raises the question of how much the project should be accountable for.
- **A safer sequencing suggested:** put the right lower-level services in place first — reliable audit trails and so on — so there is a stable platform, rather than building writes on top of the equivalent of raw SQL everywhere.

How Patient-Submitted Data Is Handled

- **Current OpenEMR approach for questionnaires:** submissions go into the portal's onsite activity table, which surfaces as an alert in the clinician's inbox. The clinician opens it, reviews it, and approves it into the chart. Stephen Nielson noted he does not like how that table is structured.
- **None of the other FHIR write endpoints use that holding area** — neither the earlier limited ones nor recent contributions.
- **The Epic model, as described by Brady:** patient-reported data sits alongside other notes rather than in a separate holding area, but is clearly tagged and highlighted as unconfirmed, and the clinician marks it approved or not. David Eschelbacher noted he likes the Epic portal's questionnaire experience.

FHIR itself supports this distinction. Resources such as Observation and Condition carry a status or verification status (for example, preliminary versus final, or unconfirmed versus confirmed), and a Provenance resource can record who supplied the data. That means an Epic-style “visible but marked unconfirmed” approach is expressible in standard FHIR, rather than requiring a separate holding table.

How to Extend OpenEMR — A Preferred Order

SMART App vs. Module

- **They are different tools and can be combined.** A SMART app can be fully independent. The reason to deliver a SMART app via a module is that a module can auto-register confidential or system apps, which normally cannot be self-registered. People can then use the app in a browser or from the patient portal.
- **SMART on FHIR still enforces its permission model** — a clinician or patient granting an app access to act on their behalf.
- **When to choose SMART on FHIR:** when the user or application is less trusted and a defined contract is wanted; or when building something that should run on OpenEMR, Cerner, Epic, and others through standard APIs.
- **A module is the most integrated option** — full flexibility to do anything.

The Ways a Module Can Reach Data

- **Roughly four approaches:** direct database access, the internal service classes, the APIs (including SMART on FHIR, which is built on top of them), and the event system.
- **Direct database writes to core tables are strongly discouraged.** A module outside the bundled set can do whatever it likes, but it opens itself up to a world of hurt.
- **Service classes are deeper still** — the internal PHP classes — and there is much you can do there that you cannot do through the API. OpenEMR does not have full API parity with its services, and is some way off.
- **API stability:** the APIs have had very few breaking changes — perhaps four or five endpoints where the input or output signature changed, rather than simply gaining fields.

The Order of Preference

- **Eric Stern's ranking for extending OpenEMR:** SMART on FHIR and the FHIR APIs first; then the internal APIs; then a module; then a core contribution — unless the core contribution adds the API needed to do the thing officially.
- **The rationale is upgradability.** The more code goes through official, stable channels, the less likely a version upgrade breaks it — increasingly important with frequent security releases.
- **How the message has evolved:** first it was stop hacking core and use modules. Now the next step: having moved to modules, try to use the APIs.

This ordering is a useful public statement and complements the 9/15 clarification that applicability, not packaging, decides what belongs in core. Together they give contributors a clear decision path: prefer standard APIs, then OpenEMR's APIs, then a module, and change core only when adding a missing extension point.

The Permission Model

How ACLs Are Used in Practice

- **OpenEMR's built-in ACLs** were defined roughly a decade ago based on a few practices' roles and best practices, and have carried forward largely unchanged.
- **Practices customize heavily.** The Ireland deployment, for example, has around 12 roles and about 10 additional access control objects added to gate specific functions.
- **Stephen Waite:** part of that customization reflects a lack of robustness in the default ACLs. Better-defined defaults would reduce customization, though every group will still want its own.
- **Eric Stern asked** how much of the ACL model is genuinely practice-specific versus well-defined industry roles that could simply be built on.

How FHIR Scopes Relate to ACLs

- **FHIR scopes are coarse** and are generally treated as the outer layer of defense — broad resource grants. Underneath sits the authorizing user's own permissions, which are much finer-grained.
- **SMART on FHIR v2 added finer scoping** by code system, code type, and individual parameters, and OpenEMR now supports it — but those scopes do not map onto ACOs and ACLs.
- **A resource constraint filter already exists in the API**, intended eventually to support full policy merging.
- **Withheld results:** FHIR allows an optional OperationOutcome reporting that some results were withheld. OpenEMR does not currently report this.
- **Eric's question:** could FHIR-defined permissions become the lowest layer, with roles built as pre-combined FHIR permissions? Stephen's view: FHIR's scoping is far too rudimentary to be the inner layer; it is the outer defense.

SMART App Launch v2 introduced granular scopes that can append search parameters to a resource scope — for example, permitting access only to Observations of a particular category. OperationOutcome is FHIR's general resource for reporting errors, warnings, and informational messages, and servers can include one in a search result to indicate that some matching data was suppressed. Both give the FHIR layer more expressiveness, but neither replaces the application's own access control underneath.

The Four Layers That Need Merging

- **Stephen Nielson's framing:** a correct authorization decision requires merging four things — the user's permissions, the patient's consent, business and facility policy, and any legal holds on the records.
- **Consent handling today is weak.** OpenEMR has checkboxes, but clinicians must manually flag information as sensitive, and nothing merges patient consent with what is being requested or with legal holds. Information-blocking rules add further requirements. All of this needs policy merging at the API.
- **FHIR's Consent resource** covers some of this but is not fully built out — described as experimental rather than normative.
- **The aspiration:** something like Cerbos, with narrowly defined policies merged together in the manner of AWS IAM policies.
- **A security document exists in the repository**, possibly with a flow chart of where merging is needed; Stephen will try to locate it.

The tool named on the call was Cerbos (transcribed as “Cerebos”). Cerbos is an open-source authorization engine that evaluates requests against policies written in YAML, keeping access rules out of application code. Notably, it offers a PHP SDK, so it is a plausible fit for OpenEMR rather than only an architectural reference. The AWS IAM comparison refers to evaluating several independently written policies together, where an explicit deny anywhere overrides any allow — the property that makes it possible to layer consent, legal holds, and facility policy on top of user permissions safely.

On the FHIR Consent resource: it is designated Trial Use rather than Normative in current FHIR releases, meaning its structure may still change. That supports the caution expressed about building a consent model entirely on it.

The Harder Part Is Policy, Not Code

- **Eric Stern:** implementing access controls is the easier half. Understanding the policy and business context — what is in scope for enforcement — is the harder part.
- **Stephen Nielson:** everyone knows how to build a permission system; the difficulty is the medical and legal context on top, plus jurisdictional rules such as GDPR. He tends toward perfect and is trying to aim for good enough for now.

- **A practical constraint:** whatever is built has to remain usable by administrators — able to scale up to an organization weighing Epic, and down to a small med spa with very different compliance needs.
- A wish expressed for access to a dedicated security expert — a human one — to test these ideas against.
- Tabled for now; too large to resolve on the call.

Upgrading from 7 to 8 — Practical Advice

Simon Quigley is upgrading a client installation from 7.0.2 to 8.x, and asked for known rough edges. Advice from the group:

- **With minimal customization, it should be straightforward.** His customizations are around 20–30 layout-based forms and essentially no code changes. In that case the upgrade is mostly replacing the code and carrying over the sites directory. There is a wiki page for upgrades.
- **Sherwin Gaddis upgraded four installations from 7 to 8.4 the previous weekend**, all clean, in about six hours total, with no UUID problems. He is moving to 8.4.1 next.
- **UUID generation:** some large installations had serious trouble with UUID generation, but that was believed to be the 6-to-7 upgrade, not 7-to-8. Already done if you are on 7.
- **Rehearse on a clone of production first.** Recommended by several people — there is always something someone modified without telling anyone. Sherwin had done exactly that before the 8.4.1 upgrade that morning.
- **If upgrading in place, back up the database, documents, and keys.** It was believed that 7 to 8 changes the encryption keys, so without backups an in-place upgrade could leave the system unrecoverable. An upgrade on the side followed by cutover is safer and gives a QA path.
- **Database engine:** the install scripts moved toward MariaDB conventions and better strict-mode compatibility, not yet complete. MySQL users may hit settings issues. Anyone still on MySQL 5.6 has had problems.
- **Stick to tagged releases.** Running from master at one point caused a database change mess for some. Simon confirmed they stay on tagged versions and cherry-pick only when necessary.
- **Known remaining bugs in 8.4:** some were fixed in 8.4.1, others remain.
- **Simon's environment:** Ubuntu 24.04 on a virtual machine, deployed with Ansible, running MariaDB 11.4.4 from a PPA, to be updated to 11.4.13. Considering Ubuntu 26.04.

MariaDB 11.4 is a long-term support release, so staying within the 11.4 series during the upgrade is a sensible, low-risk choice. MySQL 5.6 reached end of life in February 2021, which is consistent with the problems described. For in-place upgrades, the OpenEMR sites directory holds site-specific configuration, uploaded documents, and encryption keys, which is why it is the critical item to carry over and back up.

Simon's Company

- **Currently one client requesting features**, with a second OpenEMR client that has not asked for any. The long-term trajectory is an ongoing conversation; new requests arrive every couple of weeks.
- **On client confidentiality:** a reminder that vendors generally cannot disclose client details without permission, so questions about specific clients are best avoided on the call.

AI Sandboxing — Approaches Compared

- **Brady's approach uses LXC containers**, an established technology, and appears to be working well — making him very productive on security work.
- **Simon uses LXD locally** and noted Incus, a recent fork. He asked why Brady chose plain LXC.
- **Stephen Waite tried Docker's sandbox on Ubuntu 24.04** as a more plug-and-play option, launched in a parent folder holding both the openemr and openemr-devops repos. Docker's sandbox has global policy settings and uses a proxy, so the token never reaches the AI.
- **Stephen Nielson uses OpenShell**, focused on outbound controls. Its advantage is that credentials are substituted at the network boundary, so the AI never handles the real credential.
- **The gap in container-only approaches:** local isolation is handled, but outbound data is not. That is fine when working only on open source, but risky when mixing open source with client work.
- **GitHub permissions remain the weak point.** Credentials are coarse enough that a compromised agent could do far more than the task requires.
- **Documentation:** Brady's sandbox guidance is in the contributor documentation, and there is a draft PR for the sandbox. Simon was encouraged to share his LXD setup.

LXC is the low-level Linux container technology; LXD is a management layer built on it. In 2023 LXD moved under Canonical's direct control, and the Linux Containers community forked it as Incus, which is why all three names came up. They are closely related options for system containers — full lightweight Linux environments rather than single-application containers like Docker.

Testing and the Case for API-First Features

- **Browser automation with AI is now working well.** Stephen Nielson is using it for user acceptance testing on another project across hundreds of patient scenarios.
- **Suggestion: overhaul OpenEMR's end-to-end tests.** A green check mark would be more meaningful with better coverage. Service classes are getting there; the web UI is not.
- **Eric's proposal: draw a line — every new feature must be API-based** and not use the legacy templating systems, because otherwise it is very hard to test.
- **Layout-based forms again:** Simon has 20-plus LBF forms. Refactoring the LBF system to be testable would go a long way.

Next Steps and Housekeeping

- **Proposed agenda for an upcoming meeting:** revisit and finalize the deprecation decisions that have been floated — multisite, running OpenEMR from a subdirectory, and pushing toward Docker. Eric's work has been blocked waiting on these. Stephen thought a multisite decision had been reached but wants to confirm.
- **Why a formal decision matters:** stating that something is OpenEMR's direction — even for a version months away — gives far more leverage than waiting on community feedback that does not come.
- **Timing:** Stephen may not attend next week, so it may move to October 13. He intends to post notes on the forum thread after the summary goes up.
- **Last week's summary link was broken on the wiki.** It works when the URL is entered manually. Suspected to be a Cloudflare setting; it behaved differently across browsers.
- Appreciation expressed for the summaries — useful for checking what was actually said versus what people remember.

The 8/25 and 9/8 summaries both touched on this sequence: deprecate non-webroot installs, enable the front controller, then the module overhaul. The multisite RFC and the document-root issue were linked in the 7/28 summary.

Items Discussed and Proposed

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

Area	Discussed / Proposed
PR backlog	265 open PRs, up from 206 a week earlier. Dependabot and simple PRs being cleared first; features need a direction decision before code review.
Review tooling	Commit shared AI review skills to the repo (a .claude directory) that check alignment with project goals, not just correctness.
Technical committee	Proposed to prioritize and schedule PR review. Needs AI automation to be tractable at current volume.
FHIR writes	Wanted, but patient-context writes need a confirmed/unconfirmed data model and audit trail. Jerry's questionnaire work awaits review.
Extension order	Prefer SMART on FHIR and FHIR APIs, then internal APIs, then a module, then a core contribution — unless the core change adds the missing API.
Permission model	Correct authorization means merging user permissions, patient consent, facility policy, and legal holds. Cerbos-style policy engine raised.
7 to 8 upgrades	Straightforward with minimal customization. Rehearse on a clone, back up keys and documents, avoid MySQL 5.6.
Testing	Overhaul end-to-end tests; proposal that new features be API-based so they can be tested.
Deprecations	Revisit multisite, subdirectory installs, and the Docker push at an upcoming meeting — possibly October 13.
Wiki	Last week's summary link broken; suspected Cloudflare setting.

Notes

- Recurring theme: the bottleneck has moved from writing code to deciding what should be accepted. AI now makes contributions cheap and review of correctness faster, but judgments about project direction, patient safety, and legal liability — as in the FHIR writes discussion — still depend on a few people with deep context. Contributors who help turn those judgments into written guidance, shared review skills, or better test coverage would ease the pressure directly.