OpenEMR Weekly Meeting Detailed Summary

September 15, 2026

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

New Participant: Simon Quigley

- **Returning contributor.** Was involved with OpenEMR as a developer previously and is coming back to it for a specific project. GitHub: tsimonq2. Works at Altispeed Technologies.
- **Deploys OpenEMR via Ansible rather than Docker,** on a yearly upgrade cycle. Stated goal: upstream as much as possible rather than carry a fork.
- **Has two open pull requests,** both driven by requests from a client practice.

> *Simon Quigley is a Debian Developer and long-time Ubuntu contributor — relevant context for how he framed his questions about project policy, since Debian has formal governance procedures for exactly the kind of question he raised about AI usage.*

PR 13795 — LBF Statements Module

Pull request: https://github.com/openemr/openemr/pull/13795

- **What it does:** adds a module ("Form statements" in the interface, "LBF statements" internally) that generates prose sentences from numeric values already recorded on a layout-based form, and writes the resulting paragraph into a textarea on that same form.
- **The worked example from the PR:** a value between 0.0000 and 1.0000 produces "This value is mildly recorded"; 1.0001 to 2.0000 produces the moderate sentence; above that, the severe one. An administrator defines those bands once, and the sentence is then generated consistently rather than a clinician remembering which template matches which cutoff.
- **Rule types supported:** numeric bands, ratios between two numeric fields, and list or text tokens. Overlapping bands on the same field are rejected at save time, with database-level locking so two administrators cannot create a conflict concurrently.
- **Why not Nation Notes or Text Notes,** addressed directly in the PR: both are template-insertion tools where a person reads the chart, picks a template, and writes. Neither reads the other fields on the form instance, so neither can act on the recorded value. Switching the destination field to Nation Notes would also change its data type and discard existing form data.
- **Design consequence:** the measurements stay ordinary LBF fields, so graphing, printing, and export continue to work, and the generated paragraph is written as ordinary form data — meaning stock LBF printing picks it up with no changes.
- **Scope:** roughly 5,000 lines across 33 files, entirely self-contained within the module directory with no touch points into core. Includes isolated unit tests; patch coverage reported around 67%. CodeRabbit rated it complex, roughly an hour of review.
- **Rules require the admin/super ACL; generation requires encounters/notes.**

> *Correction to the meeting notes: this PR is not an autocomplete feature and does not involve Nation Notes. The PR text explicitly argues against the Nation Notes approach. The autocomplete framing appears to have arisen during the call as participants worked out what the module did — the discussion moved from "is the main feature autocomplete?" to a narrower understanding by the end.*

PR 13852 — Window Envelope Statements

Pull request: https://github.com/openemr/openemr/pull/13852

- **What it does:** adds a Billing global, “Statement envelope,” that repositions the clinic return address and patient address on printed statements so they line up with the windows of a double-window envelope.
- **The problem:** stock OpenEMR statements print addresses where they do not align with envelope windows, so a folded statement shows neither sender nor recipient through the glass.
- **Options provided:** None (unchanged), US #9 double-window, US #10 double-window, or Custom — which accepts the measurements printed on the envelope carton in inches or centimeters, including fractions such as 3/8 and 1-3/16.
- **Also included:** payments are directed to the facility mailing address rather than the physical clinic address when a mailing address is present, with fallback to the physical address so the window is never blank.
- **Implementation note:** geometry and address fitting live in a single StatementEnvelope class shared by the statement template and the EOB PDF path. Long addresses wrap and shrink to fit. HTML, plain text, and portal statements all honor the setting.
- **Stephen Waite posted a review on this PR the same day as the meeting,** asking that the render callback be moved out of globals.inc.php into a class under src/ rendering a Twig template, following the existing telehealth config pattern, with JavaScript as a separate asset rather than inline.

> *This one is a straightforward core change rather than a module, and it is the kind of small, self-contained contribution that fits the “bring in straightforward PRs” position expressed later in the meeting.*

An Unanswered Question

- **Both PR descriptions ask whether AI assistance is welcome in OpenEMR contributions,** noting that Simon did not use it and that Debian recently passed a general resolution on the subject. The question does not appear to have been answered on either PR or during the meeting.

> *Worth answering, given how central AI-assisted development has become to the project's own workflow — a contributor asking in good faith whether it is acceptable, and receiving no reply, is a poor signal. A short written position would settle it for everyone.*

Module Policy — What Belongs in Core

The Concern Raised

- **If new modules are directed to Packagist and no new modules are accepted into core, features arrive unreviewed.** Nobody knows their quality, nobody knows how they are built, and module authors get no feedback. As the project upgrades, it will not know which modules it has broken.
- **The worry stated plainly:** a parallel ecosystem of important features develops that the admins are too busy to look at, making it hard for OpenEMR to progress as a product.

The Clarification

- **Being a module does not mean being outside the code base.** Core modules exist and are not leaving. The test is applicability, not packaging.
- **The working rule offered:** if a feature is needed by a large share of users, it belongs in core. If it talks to a third-party service, that is a clear line against — OpenEMR should not be opinionated about which vendors a practice plugs into.
- **Applied to the case at hand:** if the same feature had been submitted embedded rather than as a module, it would have been considered for inclusion on its merits. Doing it properly as a module should not count against it.
- **Conversely,** the fax module is unlikely to be removed because clinics need it, whereas a vendor could equally build their own fax integration and distribute it independently.
- **The design intent behind the module work, restated:** the goal is to address upgradability and ease of installation — a packaging problem more than a distribution problem. Anything included by default would still get the same review as a direct contribution.
- **The resource constraint stated honestly:** not a refusal to accept features, but limited capacity. Where long-term contributors actively maintain an area, bringing code in is fine. Where something will generate ongoing development and concerns, active maintainers need to be identified first.

Bundling Third-Party Code

- **Strong position against the transitional middle ground** — extracting a module from core but continuing to ship it in the official package. For security purposes that is equivalent to including it: if a vulnerability is reported and the author is not fixing it, the project is still answerable.
- **Counter-position:** this is the same as any other Composer dependency, and the project already declines to depend on anything without an active maintainer.
- **Rejoinder:** an actively maintained package is not necessarily an actively secured one, and the exposure here is a logged-in user inside OpenEMR.
- **Resolution reached:** modules packaged with core should live under the OpenEMR organization, so the project retains the ability to fix them. A contributed module could change ownership to OpenEMR if the author wishes.
- **A fairness consideration raised:** where a contributor has volunteered to take a module out of core on the understanding it would still be distributed, changing that expectation later would reduce its reach. Not a reason to avoid the change, but a reason to provide a good installation path first.
- **The unblocking idea:** a module registry — a list of modules certified to work well with OpenEMR. With that in place, extracted modules could be removed from the package without stranding anyone, and the contributor would be comfortable.

How Hard Is Installing a Module?

- **Essentially a single Composer install command.** Anyone who has worked out the OpenEMR upgrade process can manage it, and AI assistance makes it easier still.
- **The nuance is immutable Docker images:** for those, you would build your own image on top of the OpenEMR image with the module installed at build time. Most users do not work that way.
- **The envisioned shape:** the official Docker image as a base containing core OpenEMR, with deployments layering their chosen modules on top — the base stays immutable and each site's distribution is that base plus its own additions.

The Decision on PR 13795

- **Reviewed live on the call.** Assessment: cleanly structured as a module with no touch points into core; queries reasonable in number; no major problems visible. One file flagged — dynamic JavaScript generation in the encounter toolbar, which should be a separate JavaScript file with variables injected.
- **The feature was judged a genuine value add.** Nothing else in OpenEMR does this across all layout-based forms.
- **Why LBF specifically:** layout-based forms are standardized, so one implementation reaches every LBF form in existence. Custom forms are all different, and many already do their own totalling and range logic. Simon confirmed the approach could extend to other form types but has not been generalized.
- **Suggestion raised:** abstract the logic into a library that other areas of the application could call, rather than keeping it available only within the module.
- **Argument that the module form is an advantage:** embedding it would add to the existing tangle, whereas an isolated module is easier to separate later and may provide a model for eventually extracting all the LBF machinery into a core module of its own.
- **The case for waiting:** let it run on Packagist, see whether there is demand, then bring it into core — at which point inclusion is a single line in composer.json. Hesitancy about adding new surface area while the security workload is heavy.
- **The case against waiting:** community feedback will not arrive. Experience is that forum posts get no responses; the only feedback that comes back is when something is broken. The decision has to be made on the merits by the people making it.
- **The countervailing point on review capacity, stated candidly:** review has been triaged to bug fixes and broken behavior, with feature PRs left sitting. The suggestion made was to branch out — have someone take on straightforward feature PRs, since a clean, simple PR is much easier to bring in than one doing unusual things.
- **Position on messaging:** rather than telling contributors the project is not accepting features, say that capacity is limited, the intent is to get to it, and to keep pinging.
- **Deferred to next week** given the disagreement, and because the release discussion needed the remaining time.

Doctrine, Schema Drift, and Static Analysis

- **Hesitation expressed — not specific to this PR but to all current code:** queries and column references are plain strings, so static analysis has no visibility into them.
- **The concrete failure mode described:** a deployment where schema changes were not applied somewhere, and nothing caught it because nothing could. Code that is correct today can drift out of sync as the schema evolves, and that is a risk for the project as a whole.
- **With Doctrine, the mapping is structured and the tooling understands the real schema,** so that class of problem disappears. Hand-constructed SQL goes away, and handling of new columns and schema changes becomes largely automatic.
- **What is blocking adoption:** the core foundation is in place; what remains is the front controller and some routing work so that modules reliably use it.
- **Counter-argument on sequencing:** the harder problem is the legacy code that follows none of the current conventions, not newly written code that already follows most of them. Converting a well-structured module to Doctrine later should be largely mechanical.

> *Doctrine is the standard PHP object-relational mapper; DBAL is its database abstraction layer. The practical benefit in this context is that table and column names become part of typed, analyzable code rather than strings assembled at runtime — which is what makes both schema drift and injection-class problems visible to tooling rather than only to a human reader.*

- **Recognition of what the existing tooling has already achieved:** contributors are now producing markedly more structured PRs because PHPStan, CodeRabbit, and the enforced conventions catch problems before a human looks.

Layout-Based Forms

> *Layout-based forms (LBF) are OpenEMR's mechanism for building custom clinical forms through the administration interface rather than by writing code. An administrator defines fields, data types, and list options, and the resulting form is stored in the database and rendered generically. Values are stored as rows in a shared data table keyed by form instance and field, which is why a single implementation can reach every LBF form at once.*

- **Widely regarded as both a major strength and a maintenance burden.** Described as an awesome feature at one end and a pain source at the other.
- **Evidence of demand:** a lecture on layout-based forms given roughly four years ago remains by a wide margin the most-viewed video the project has ever published.
- **The associated technical debt:** the options file behind the layout machinery runs to around 4,000 lines and is acknowledged as needing substantial refactoring. A candidate for eventual extraction into a core module.
- **Telemetry gap:** the project does not currently know how widely layout-based forms are used, because telemetry covers menu usage rather than encounter activity.

Telemetry

- **Proposal: record which modules are installed on an instance.** That would answer adoption questions directly — publish a module, then see who installed it — rather than relying on forum feedback that does not arrive.
- **Broader point:** there is a lot that could be added to telemetry, and the current coverage is not well documented. Suggestion made that the simplest way to find out what is currently collected is to ask AI to read the code base and report.

> *This connects directly to decisions taken in earlier meetings. The fax/SMS core-versus-module question on 8/11 also stalled on the absence of usage data, and the backup.php removal on 8/18 turned on a telemetry figure of roughly 10% adoption. Module and feature telemetry would make several recurring arguments resolvable with evidence.*

Release Status

- **Two bugs under discussion for a point release.** First, the refined patient search — the field-by-field narrowing search, not the main patient finder — is broken. Confirmed as not the main finder, which reduced the severity considerably.
- **Second, the encounter form is broken,** traced to a change in how the patient identifier parameter is passed. Considered significant enough to justify a release.
- **Also outstanding:** portal fixes from Jerry restoring auditing, and a long-standing data issue where male and female values are transposed in a FHIR context. The latter has apparently existed for a very long time without anyone noticing, and Inferno does not catch it because the test suite has no way to know which patients are which.
- **Argument for doing the point release:** the release mechanism has only recently been straightened out, so exercising it on a low-stakes release is worthwhile — breakage is likely, and better to discover it now than during a genuinely critical issue.
- **Mechanics noted:** a database change requires manual handling around the dev release tag, since tagging generates a further upgrade file in master that has to be moved back. Not painful, but manual.
- **An unrelated observation on release friction:** the most tedious remaining step is manually removing unsubscribed addresses from the mailing database. Automating it was raised as worth asking about.
- **Tentative plan:** gather the available fixes and do the release over the weekend.

Vendor Upgrade Experience

The General Problem

- **Discussion of how vendors keep large deployments current.** The recurring difficulty is that customizations made directly to core code must be rebased onto each new version, and each release starts a clock.
- **Speaking generally rather than about any specific deployment:** the motivation behind simplifying module development and making things easier to disentangle is precisely so that deployments can pull in patches without rewriting half their code.
- **Performance work has been contributed upstream** where it was not specific to any one installation. A synthetic test with roughly 100,000 calendar events across a year rendered the calendar page with about 15,000 queries; after the fixes it ran about eight. That class of problem surfaces only at scale.
- **On the proposed module system:** real-world deployment experience is largely aligned with the design. The base — roughly three hooks covering command line, database, and API registration — remains a good starting point, with further layers to be added as modules become more sophisticated.
- **Repeated request for feedback:** the module design is currently informed by three or four people's views, which is a limited view of how OpenEMR is actually used. More real-world input would make the design more comprehensive.

Event Listeners vs. Bootstrap Registration

- **Two different approaches were described.** The existing custom modules use event listeners. The newer approach uses a low-level bootstrap — a three-line change in the file every request passes through, which registers the modules.
- **Why this requires the front controller:** a single registration point is needed for modules to function reliably. Every request must run the modules it is supposed to run, and without one entry point there is no way to guarantee that.
- **The two are orthogonal rather than competing.** Single-point registration is a requirement for any reasonable approach; whether customization happens through events or through a defined API is a separate, more philosophical question.
- **That deeper question framed:** OpenEMR today can be customized so extensively that a heavily modified installation is nearly a different piece of software. That is powerful, but it makes it hard for core to offer any guarantees to modules. The alternative — defined, largely additive extension points — trades flexibility for predictability. There is no technically correct answer; it is a decision about what OpenEMR should be.

> *This is the same tension the project has returned to repeatedly: the fax/SMS extraction, the backup.php removal, and the module registry all turn on how much the project is willing to own and guarantee. The front controller keeps appearing as the prerequisite because it is what makes any guarantee enforceable.*

Sherwin Gaddis on Real-World Upgrades

- **Test run completed upgrading 8.3 to 8.4** with core modifications in place, working toward a documented playbook. The challenge is rebasing existing modifications onto the new code base.
- **Key caution:** you cannot fall far behind. Too many versions back and rebasing stops working — files change completely or disappear, and modifications may no longer apply at all. A five-to-eight jump was called too much.
- **Longest migration performed:** version 3 to version 7, successfully, though complicated by a site that had installed version 3 twice and copied patients between the two instances. Almost nightmarish to unpick, with essentially no documentation of why anything was done as it was. The client was happy with the result.
- **Current plan:** move every client to 8.4 before the end of the year, starting with unmodified installations where the upgrade is just the SQL step. Holiday season provides the window. Motivation is not falling behind on security and features.
- **On modules and upgrades:** everything was moved into modules roughly four years ago, and anything needed in core gets pushed upstream so it is available to everyone. This helps, but modules still misbehave between versions — they have to be installed rather than simply moved, and quirky issues occur even when the database records them as installed.
- **Refactoring a module between major versions is described as mild,** mostly globals and session handling — bring in the global bag, fix the sessions, and it works. Written to standard, going from 7 to 8 should not be a large problem. Their published module list states which versions each module supports.

Payment Processors

- **Question raised about supporting additional payment processors,** prompted by interest in lower-cost alternatives to the processors currently supported.
- **Advice given:** do not follow the pattern of the processors currently in the code base. Build a module that ties into what exists rather than baking another integration into core.
- **The gap identified:** there is no generalized payment interface in core that individual processor modules could hook into. Agreement that one is needed — a generic mechanism in core, with per-processor modules.
- **Prior art recalled:** a package existed that built a gateway abstraction across credit card processors, so integrating code only had to speak to that gateway and the gateway negotiated each processor's differing API. Believed to have been looked at years ago.

> *This is almost certainly Omnipay, a long-established PHP payment abstraction library that provides a consistent interface across a large number of gateway drivers. It remains actively maintained and is the closest existing match to what was described.*

- **Current state:** a recent processor addition sits alongside the older integration in the same connections area, which was noted as the current project norm but also as accumulating duplicate code. The suggestion was that the connections section will need to be broken out into its own tab as vendors keep being added.
- **Also noted:** the card reader supported by the older integration is outdated and possibly no longer compliant.
- **A prior conversation with a major processor did not progress** because the arrangement proposed did not fit how the foundation and independent clinics relate to one another. Worth revisiting.

Security Workload — Context Given to the New Contributor

- **Asked how long the current security situation would last,** the answer given was that it is an industry-wide phenomenon affecting essentially all projects, not something specific to OpenEMR.
- **An infrastructure indicator:** CVE identifiers that were once issued within a day or two are now taking considerably longer across the board, pointing to a backlog in the underlying process rather than anything specific to this project. Releases go out ahead of identifier assignment, with identifiers attached afterward.
- **Reason for cautious optimism:** there is now noticeable duplication in reports — different people using automated tooling find and report the same issues. That implies a finite set rather than an endless stream. Rough estimate offered: perhaps a year, with the rate levelling off and beginning to decline in three to six months.
- **The realistic counterweight:** OpenEMR is a high-value, source-available target. It will attract more scrutiny than a hobby utility because the payoff is higher. The volume should become much smaller, but it will not stop.
- **The compensating benefit:** the pressure has driven real infrastructure improvement — the release mechanism, the demo farm, and much that previously required manual work are now automated, which puts the project in a position to focus on the backlog rather than the pipeline.
- **Refactoring work has second-order security benefits,** not just maintainability ones: understanding what data is where is what makes it possible to verify that data is safe in a given context. As that improves across the system, the report volume should fall.
- **Centralization compounds:** as logic is consolidated, one fix replaces what would previously have been many.

Housekeeping

- **Last week's forum post implied the meeting time change applied to the weekly call.** It applies to the monthly Foundation board meeting, which moves to 10am Pacific / 1pm Eastern next month. The weekly call is unchanged. The forum post was corrected during the meeting, and the minutes already had it right.
- Noted that the forum software records and displays edits, so corrections are visible rather than silent.

Items Discussed and Proposed

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

Area	Discussed / Proposed
PR 13795	Decision on whether the LBF statements module warrants inclusion in core deferred to next week. Disagreement acknowledged; no objection to the code quality itself.
Module policy	Clarified: the test is applicability, not packaging. A module that delivers a broadly useful feature can still live in the code base. Third-party and vendor-specific modules go to Packagist.
Bundled modules	Position taken that modules shipped in the official package should live under the OpenEMR organization, so the project can fix security issues in anything it distributes.
Transitional packaging	Consensus against a long-term middle ground where a module is extracted from core but still bundled. Preference is for a proper post-install path.
Module registry	Revisited — a list of modules known to work well with OpenEMR, which would let extracted modules be removed from the package without stranding their users.
Telemetry	Proposal to record which modules are installed on an instance, so adoption questions can be answered with data rather than guesswork.
Release	Patient search (refined search, not the main finder) and the encounter form are both broken. A point release was discussed for the weekend, partly as a rehearsal of the process.
Front controller	Still the blocker for the module system. An up-for-grabs demo remains outstanding.
Payment processors	No generalized gateway abstraction exists in core. Suggestion that core define a generic interface with per-processor modules.
Forum correction	Last week's post implied the time change applied to the weekly call; it applies to the monthly Foundation meeting. Corrected.

Notes

- **The central tension of the meeting, stated cleanly by one participant:** the project is telling contributors to build modules, and then declining to review modules. Whether or not that is the intent, it is how the policy reads from outside, and the clarification reached — that applicability rather than packaging decides inclusion — is worth stating publicly.
- Recurring theme: several threads ended at the same place — telemetry would answer the question. Module adoption, LBF usage, and feature demand were all discussed without data, and in each case someone observed that community feedback rarely arrives on request. Community members who do have views on these questions would be unusually useful right now, precisely because so little feedback comes in unprompted.