OpenEMR Weekly Meeting Detailed Summary

September 8, 2026

Attendees: Brady Miller, Stephen Nielson, Stephen Waite (departed early), Eric Stern (OCE), Asher Densmore-Lynn, David Eschelbacher.

Note: a small meeting, attributed to the holiday week. The call ran on Google Meet for the first time, with the Zoom room left open in parallel for anyone joining out of habit — nobody did. Live captions were noticed to be transcribing the whole meeting, though no way to export them was found.

Release 8.4

- **Branch cut targeted for that night, the following day at the latest.** The full release expected over the weekend, likely Saturday, given work schedules.
- **A large PR needed to land first** — something already released into the wild that had to be in. It went through many iterations on the public repo, with Inferno fixes and automated review corrections repeatedly working against each other.
- **Must-haves for the release:** the outstanding security work and the major calendar break.
- **A number of smaller fixes are also outstanding** and would be good to include. Position taken: if something is deemed critical, bring it over rather than blocking the cut.
- **The three-day window after a cut** gives time to bring items over if people judge them necessary.
- **Intent stated:** review what the large PR actually changed before cutting, given the volume of automated code changes.

Inferno Test Suite

- **Inferno is reporting false passes** — it says it works when it is not working. This has been masking real breakage.
- **Initial diagnosis:** some tests take too long and hit a timeout, which appears to kill other tests in the same set. A full run takes 10–15 minutes, roughly 20 with the round trip.
- **Expected to be several layers deep,** and once it starts working properly there will likely be genuine failures to fix, given how much has changed.
- **A PR is in flight using Stephen Waite's earlier commits,** which is unmasking the underlying breakage rather than fixing it outright. Baseline comparison still worked well enough to confirm no new breakage was introduced.
- **Practical note for admins:** Inferno testing can be run on a PR by putting it on a branch of the OpenEMR repo rather than a fork.
- **Coverage confirmed:** the suite uses the Inferno test suites directly with the US Core profiles, testing 8.0 with backwards compatibility to earlier versions. Profiles are pinned, so they may need updating to pick up upstream fixes. An AI-generated claim that 8.0 was not being tested was incorrect.
- **Broader question raised:** with CI now very large — Docker and acceptance testing added — whether other jobs are also reporting true passes deserves checking.

Meeting Time and Structure

- **The move to Google Meet happened this week.** Zoom was left running in parallel; audio carried faintly between the two but nobody joined the old room.
- **Time change proposed for next month:** Wednesday at 1pm Eastern (10am Pacific, noon Central). Wednesday at 2pm was the alternative but would have collided with the release call. Deliberately staged — change the location first, then the time.
- **Asher flagged a possible conflict** given employer commitments, but agreed to try.
- **Consensus that the separate release meeting is no longer needed.** The mechanism is automated enough that the meeting has lost its purpose. A security discussion may occasionally be warranted, but that happens on Signal.
- To be confirmed at the following night's meeting.

Vendor Hooks for Container Upgrades

- **Asher's work:** a designated point in the Docker build where a vendor-supplied script runs, so customizations can be restored after a container is thrown away and replaced with an updated one.
- **The design is deliberately minimal.** It does not copy state from the old container to the new one — the script pulls changes from wherever the vendor stores them. Asher wanted a bare-bones version rather than over-engineering, so early adopters can say what is missing.
- **He came to the call specifically for feedback,** uncertain whether this is something people want, something they will not realize they want until they try it, or something nobody will ever use.
- **Response: clearly useful.** With releases moving to roughly monthly, vendors and groups need a way to take on upgrades without losing their work. Keeping customizations intact across upgrades is precisely what the project is asking people to do.
- **The concrete test case suggested:** custom modules. There is a directory holding installed modules, with the installation state in the database — those files disappear on upgrade. Demonstrating a script that restores them would show the pattern.
- **Asher's scope:** a Dockerfile with the barest hint of a script, on the reasoning that showing a vendor it is possible is enough — they know their own environment. Not planning to build out full testing.
- Noted candidly that he does not use OpenEMR himself, which is part of why he wants adopter feedback.

AWS Image Strategy

- **The Standard project is being allowed to lapse** — an image plus a CloudFormation stack, with a persistently poor AWS Marketplace experience that never improved.
- **Express Plus CloudFormation template stays** and remains in use.
- **Express stays too** — a single monolithic image, easy to support, serving as a toehold in the marketplace. Not certified-compatible, and that is made clear up front.
- **ECS is the first multi-node offering,** using Aurora Serverless, with EKS above it as the Kubernetes version.
- **Nothing announced yet** — Asher wants the vendor hook work in place and documented first, then will sweep the wiki and put more weight on ECS as the multi-node recommendation.
- Telemetry confirms the AWS images are still being used, though visibility is limited.

Front Controller, Performance, and Tooling

Front Controller

- **OCE has been testing the front controller internally.** A few deployment-specific issues surfaced; the code itself had no problems.
- **Caveat:** they are not running every module that could be run, so third-party compatibility remains unverified.
- **Still wanted — Docker packaging, front controller, the things that enable downstream cleanup** — but hard to prioritize with so much in flight. The original estimate has slipped as enterprise support work took precedence.
- Refactors are being sneaked in where possible rather than tackled as a dedicated effort.

Performance Work

- **N+1 query problems are being found and fixed,** with fixes contributed back to core. Driven by work on a large install where some pages were taking 45 seconds to load.
- **The calendar was the worst case:** with roughly 100,000 synthetic events generated for testing, building the page ran about 14,000 queries — re-querying the database for every appointment rather than fetching up front. Believed knocked out on the calendar homepage; the find-availability piece still has a bottleneck with large numbers of long-running repeating events.
- **A similar case:** a pharmacy list of around 10,000 entries doing N+3.
- **The DBAL migration would enable query logging** — not audit logging but debug console logging, so a page generating thousands of queries becomes immediately visible. Named as a priority once there is capacity.
- Twig migration continues opportunistically — pages get swept over rather than left inline where practical.

Deptrac

Configuration example: https://github.com/Firehed/php-lsp/blob/main/deptrac.yaml

- **Raised as possibly subsuming a number of custom PHPStan rules.** Rather than blanket-forbidding a function, it constrains which locations may call it — so instead of banning file access outright, all such calls must live in one place and everything routes through it.
- **Expected performance benefit:** syntax-tree pruning rather than identifying every one of thousands of call sites. Supports baselining in the same way.
- **Directly relevant to the FlySystem work** — core components are in place, but migrating the many remaining call sites is the same class of problem.
- **Reservations expressed:** the volume of tooling already in place is getting hard to manage, and adding more mid-marathon may not be the right move versus picking one thing and driving the numbers down. Agreement that this is worth watching rather than adopting now.
- Candid note: the exploration was entirely AI-driven, so familiarity with the actual configuration is limited.

Static Analysis and Code Quality Enforcement

- **An idea first discussed months ago was revisited:** using PHPStan to enforce that database queries are constructed only from literal strings defined in code, rather than assembled dynamically. Any dynamic value interpolated into a query string would then be flagged for review.
- **Why it stalled:** the existing PHPStan baseline is very large — on the order of 140,000 items — so a flagged item of real substance is hard to distinguish from a routine code style entry. That is why the change was not landed earlier.
- **The proactive proposal:** capture the current violations as a baseline, then use AI to work through that baseline and separate genuine concerns from code that is merely not in the preferred style. Enforcement for new code would then be straightforward.
- **Acknowledged limits:** a flag indicates risk rather than a confirmed problem, some patterns would be false positives, and some existing code could not be converted without a substantial rewrite.

- **The trade-off discussed:** the current approach is largely reactive, and general agreement was expressed that a proactive mechanism would be preferable at least for new code, even if retrofitting the whole code base is not realistic soon.
- **This connects to the wider refactoring effort** — the same structural work on the database layer and templating that has been discussed in previous meetings addresses much of this class of problem as a byproduct.

Node to PHP Conversions

Schematron

Service repository: https://github.com/openemr/oe-schematron-service

- **A PR converts the Node Schematron service to PHP,** done by pointing AI at it. Described as the small, tractable one of the three Node dependencies — CCDA, CQM, and Schematron.
- **It removes a set of package dependencies** and does not break Inferno — though Inferno does not cover Schematron, so that is not strong evidence.
- **Testing guidance sought and given:** importing a CCDA triggers a Schematron check, but the server has to be set up for it. No official Inferno testing exists; the only CCDA-related addition to the Inferno suite was a check against a known-good CCDA. Nothing at all on the CQM side. 2022 is where to look for reference material.
- Candid admission from more than one participant that they do not know exactly what Schematron does beyond it being a validator.

CCDA

- **The CCDA rewrite branch stalled.** It reached the point of matching Node's quality, then review surfaced issues that were likely present in the Node implementation too, or at least not covered by its tests. A couple of weeks of back and forth followed and the effort lost momentum.
- **Open offer:** anyone who wants to get the CCDA branch over the line is welcome to — no attachment to authorship, just a desire to get it out of the flight path. Perfect should not be the enemy of better.
- **Scale:** mechanical work, but roughly 7,000 lines with essentially no test coverage.
- **It was the CCDA dependency that stalled the earlier Docker rework,** because of reluctance to pull Node into the runtime.

CQM

- **Assessed as the harder port.** CQM uses its own XML format unrelated to CCDA and can be done independently, but it runs its own Node-based Clinical Query Language processor built by MITRE. Porting it would mean reproducing that processor, not just translating a service.
- **Some uncertainty was noted about the ongoing maintenance status** of the upstream MITRE tooling.
- **Counter-observation:** all the heavy data querying and processing already happens in OpenEMR — the service is a thin transformer applying the query language. Described as a tiny layer doing very little, which argues it may be more tractable than feared.

Separate Service vs. Single Image

- **The alternative raised:** these do not need to be one image, only to run in Docker. CCDA already spins up as a service; it could simply be part of Docker Compose, running alongside without being in the PHP image.
- **The objection:** reluctance to require a second Docker container for ONC compatibility, which would effectively make Docker mandatory. The project points people toward Docker but has not made it a requirement.
- **The maintenance argument, stated strongly:** with a small team, keeping tools and language frameworks homogeneous matters. A separate service will not get updates and will not be looked at as much as it should be. Retrospective regret expressed about the original decision to build these in Node for speed rather than PHP.
- **The counter:** any production deployment already needs separate services — database, cache, and file-based sessions are unsuitable for multi-machine deployments anyway. If accepting this got Docker to being the only supported way to run OpenEMR, enabling rapid releases and genuinely workable upgrades, that might be worth it.

- **Practical resolution:** convert what can be converted — CCDA and Schematron look mechanical and do not need a separate service. If CQM genuinely requires it, cross that threshold then, but not before. Schematron is also around ten times faster in PHP.
- **Wish expressed:** bring in the ONC edge testing tool suite the way Inferno was brought in, then let AI iterate against it until results pass.

Docker Binary

- **Jake's Docker binary** runs OpenEMR from a binary, built against a version. With more frequent releases he has to keep up more, and it currently breaks on CCDA.
- **Framed as another argument for removing the Node dependencies:** making CCDA work would turn the package into something it is not meant to be. Removing these dependencies opens the door to the binary and other options, and makes upgrades, installs, and infrastructure maintenance simpler and more reliable.

AI Sandboxing and Containment

OpenShell reference policy: https://docs.nvidia.com/openshell/get-started/tutorials/github-sandbox#full-reference-policy

The Problem

- **Working through a large backlog with AI assistance means granting it access to project data,** which warrants care about what that access actually permits.
- **GitHub's token model makes narrow scoping awkward.** Fine-grained tokens do not cover every case, while the broader token type grants read and write across everything — more access than anyone wants to hand an automated tool.
- **The current approach:** confine the tool to a Linux appliance with access only to a working directory and no personal data, run it permissively inside that boundary, and scope its credentials to ordinary-user rather than administrative capability. A further step would be an unprivileged container so a breakout does not reach the host.

OpenShell

- **A sandboxing layer with network-level enforcement.** It spins up a sandbox, restricts filesystem access to nominated folders, and applies application-layer inspection to outbound calls — including query parameters. Static firewall rules plus dynamic rules that can be swapped while running.
- **Described as doing what Claude Code's permissions do not reliably enforce,** with audit logs suitable for providing to customers.
- **In use already:** a workflow that sets up the working environment and operates within a firewall permitting only the relevant repository endpoint.
- **Limitation:** the sandbox is ephemeral — once torn down it is gone, so the repository has to serve as the persistence layer.
- **Offer made to share or open source the policy profiles** once they are in good shape. Received positively as another layer to stack.

Threat Model

- **The primary concern named: supply chain attack and data exfiltration** — an AI pulling a Composer or Node package that modifies its own instruction files and starts sending data out.
- **Read-only restrictions are insufficient on their own.** Outbound data can leave through channels that a read-only permission model does not close, which is why network-layer inspection matters.
- Documented real-world techniques exist that use entirely ordinary tool capabilities as a transport, which is why network-layer controls were preferred over restricting the tool set.
- **Concern is proportionate to what is at risk,** with client and patient data rated well above project data. One participant now uses a separate machine for financial work.
- **Closing observation:** the workflow has become AI reviewing AI, with humans directing attention — look at CI, look at the review bot. A different role than before: more problem-solving manager than designer and builder, with the same pressure, since there is always more to do.

Items Discussed and Proposed

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

Area	Discussed / Proposed
Release 8.4	Branch cut targeted for that night or the following day; release expected over the weekend. A three-day window after the cut allows late items to be brought over.
Inferno	Test suite is reporting false passes. Investigation under way — initial finding is that slow tests exceed a timeout and take others down with them. Several layers likely.
Meeting time	Proposal to move the weekly call to Wednesday at 1pm Eastern, effective next month. Location change to Google Meet already done; one thing at a time.
Release call	Consensus that the separate release meeting is no longer needed now that the mechanism is automated.
Asher Densmore-Lynn	Vendor hook for Docker containers — a designated point where a vendor-supplied script restores customizations after a container is replaced. Looking for early adopters to say what is missing.
AWS images	Standard project to lapse. Express stays as a marketplace toehold, Express Plus as the practical option, ECS as the multi-node offering, EKS above it. Wiki sweep to follow once vendor hooks land.
Front controller	Internal testing at OpenCoreEMR found deployment-specific issues but no problems with the code itself. Third-party module compatibility still unverified.
Deptrac	Raised as possibly subsuming custom PHPStan rules by restricting which locations may call a function. Reservations expressed about adding more tooling mid-marathon.
Schematron	PR converting the Node service to PHP. Removes dependencies; Inferno does not cover it, so real-world testing guidance was sought.
CCDA / CQM	CCDA branch stalled and open to anyone who wants to finish it. CQM assessed as the harder port because of its own query language processor.
Performance	N+1 query fixes going upstream from OCE work — calendar and user pages especially. DBAL migration would enable query logging.

Notes

- Smallest attendance in some time, attributed to the holiday week.
- Recurring theme: reducing what has to be maintained. Retiring the AWS Standard project, converting Node services to PHP, consolidating file access behind one path, removing the CCDA dependency that blocks binary packaging — all the same move. The argument that carried was not elegance but that a small team cannot keep heterogeneous pieces current, and anything not kept current eventually becomes a problem. Contributors willing to take on one of these conversions — the CCDA branch in particular — would make a direct difference.