

OpenEMR Weekly Meeting Detailed Summary

October 6, 2026

Attendees: Brady Miller, Jerry Padgett, Stephen Waite, Eric Stern (OCE), Sherwin Gaddis, Simon Quigley, David Eschelbacher.
Stephen Nielson absent; expected back October 13.

Italicized indented passages are background, clarification, and commentary added after the meeting, not statements made on the call. This week's summary carries more of them than usual, because much of the discussion turned on standards and regulatory context that is worth spelling out.

Opening — AI as a Development Partner

- **Brady has been very busy with security work**, with report volume remaining high.
- **Jerry described starting a fresh AI session on the CCDA work** and asking it to compare the implementation against the C-CDA implementation guide and to find all the workarounds that had accumulated in the Node service. It took over the project to the point that he had to slow it down to keep up. Much of what it reports is beyond him, so he relies on what it shows him in tests — “I feel like I work for the AI now.”
- **Brady's strategy:** build out the tooling, and let the AI use it — automated testing, automated checks — and largely wait on the results. It still makes mistakes, but CodeRabbit catches many of them.
- **A practical tip on CodeRabbit:** if a PR receives too many commits, CodeRabbit suspends review for high activity, with a checkbox at the top of the PR to resume. It does not review draft PRs. On very large PRs the status can get garbled, and the simplest remedy is to request a full review in a comment.

The comment command is @coderabbitai full review, which re-reviews the entire PR rather than only the latest commits; @coderabbitai review triggers an incremental review. Both work on draft PRs if invoked explicitly. The observation that AI reviewers catch AI authors' mistakes is a pattern the project is now relying on structurally — which is why the 9/29 discussion about committing shared review skills to the repository matters: the reviewer's instructions become as important as the author's.

Jerry's comment that the AI's output is sometimes beyond him is worth taking seriously rather than as a joke. When a contributor cannot independently verify generated code, the tests become the specification. That puts more weight on test quality and on review by someone with domain context — exactly the bottleneck discussed last week.

Removing Node from the Runtime — CCDA Status

- **Context:** the goal is to remove the Node-based services from the OpenEMR runtime. Schematron is done. Jerry is now finishing the CCDA service in PHP, after which comes the CQM calculator.
- **Status:** the PHP CCDA implementation could have been merged already and would pass certification testing, but Jerry paused to fix known implementation-guide conformance issues first rather than merge something he knew was incomplete. Expected to merge that night or the next day, followed by a PR removing the Node path and reviewing existing tests.
- **USCDI target:** he is bringing it toward USCDI version 5, but can only test against version 4, so version 4 conformance is the realistic bar before merging.
- **Follow-up work identified:** how documents are generated and how document provenance is recorded both need fixing in a separate PR. Certification requires three document types; the mechanisms to produce them properly exist in the code but have been bypassed over the years to get through certification. Jerry wants to stop working around them.
- **On credit:** Jerry said Eric's approach — starting from the specification rather than imitating the Node service — was far better than his own original approach, and made the work easy to pick up with AI. Eric: everyone is working toward the same goal.
- **History:** Jerry introduced Node into OpenEMR several years ago, reluctantly, because at the time there was no PHP option for CCDA generation other than a paid library. Node has always needed separate installation and management.
- **Why it matters:** removing the CCDA Node dependency removes an entire installation requirement, simplifies Docker substantially, and allows the binary Docker packaging to work fully. Eric called it a huge security and deployment win.

The three document types are the C-CDA templates required for the transitions-of-care certification criterion: the Continuity of Care Document, the Referral Note, and — for inpatient settings — the Discharge Summary. Each is a defined template within the HL7 C-CDA Release 2.1 implementation guide, with its own required sections. Generating all three from one code path with switches, rather than through the template-specific mechanisms, is the sort of shortcut that passes testing but drifts from the specification over time.

USCDI (United States Core Data for Interoperability) is ONC's versioned list of data elements certified systems must support. Under the HTI-1 rule, USCDI v3 became the certification baseline from January 1, 2026. Versions 4 (2023) and 5 (2024) have been published, and developers may voluntarily advance to newer versions as test tooling catches up — which is why Jerry can target v5 but only test v4. Document provenance refers to recording who authored the information and when; USCDI includes a Provenance data class, and C-CDA represents it through author participation.

Remaining Node and Dependabot Updates

- **CQM stays on Node for now** and is Jerry's next priority. It was expected to be the hardest of the three.
- **Dependabot PRs for Node packages should keep being merged** until Node leaves the runtime. Node will remain only as a build-time package loader for the UI, where the version matters much less.
- **Timing pressure:** a new production Node release would arrive with the next Alpine Linux release, so the aim is to have Node out of the runtime before then.

Jerry referred to the CQM work as “QDA” and “DQM”; the relevant formats are QRDA Category I (patient-level quality reports) and QRDA Category III (aggregate reports). The calculation itself is done by a CQL execution engine, the Clinical Quality Language processor discussed on 9/8. Alpine Linux typically releases twice a year, around May and November, rather than in December as mentioned; the practical point stands that a new base image could bring a new Node version within weeks.

FHIR Writes — Gating, Scopes, and Auditing

- **FHIR write support has landed and been well tested**, with a few bugs identified by Brady still to address. Brady has also added the ability for the OpenEMR command utility to import patients via FHIR instead of CCDA, supported in the latest Flex Docker image, which should help close remaining gaps between FHIR writes and CCDA import.
- **Eric Stern had to pause automated Rector refactoring work** because the FHIR write changes touched a shared trait and created conflicts.
- **Jerry's priority is clinical workflow.** FHIR writes matter to him because staying current with clinical workflows is how OpenEMR keeps pace with other EHRs, and ONC requirements are moving in the same direction. He has been involved with FHIR since the beginning of the project's FHIR work.

Rector is an automated refactoring tool for PHP: it applies rule sets to upgrade syntax, remove dead code, and modernize patterns across a codebase. Large automated refactors and large feature merges in the same files inevitably collide, which is an argument for coordinating when sweeping refactors run.

How Write Access Is Granted Today

- **Current scope:** FHIR writes are available only in user context — not patient context, and not system context.
- **The flow, as clarified on the call:** someone registers a client application requesting write scopes; an administrator must approve and enable that client; then an authorized OpenEMR user with the necessary permission must log in through it and choose which of the requested scopes to grant. Only then can the app write on that user's behalf.
- Brady admitted he always gets confused about confidential versus public clients — a common source of confusion worth explaining.

In OAuth and SMART on FHIR, the client type is about whether an application can keep a secret, not about who uses it. A confidential client runs somewhere its credentials can be protected, typically a server, and authenticates to the authorization server with a secret or a signed key. A public client cannot protect a secret — a browser app or a mobile app installed on a patient's phone — so it relies on PKCE instead. Patient-facing apps are usually public clients, which is why the two ideas get conflated. Separately, scope prefixes define the context: user/ scopes act on behalf of a logged-in clinician with that user's permissions, patient/ scopes are limited to one patient's record, and system/ scopes are for backend services with no user present.

FHIR write operations come in three forms: POST creates a new resource, PUT replaces an existing one in full, and PATCH changes part of one. A question raised later in the meeting — whether to allow only creating new items, or also modifying existing ones — maps directly onto whether PUT and PATCH are permitted alongside POST.

The Decision: An Off-by-Default Gate

- **Jerry proposed a configuration setting that keeps FHIR writes off unless an administrator enables them**, as the one guard he can deliver before the 8.5 release. His view: write access is a large exposure, other EHRs do not appear to be paying much attention to it, and it is safer to push the decision onto the administrator deliberately.
- **Brady initially questioned whether a gate was needed**, since OAuth already restricts writes to approved clients and authorized users.
- **Brady then came round to it on defense-in-depth grounds:** if any authorization flaw were ever found, write access would escalate the potential impact from reading data to altering clinical records — medications, allergies, orders. A separate gate limits that exposure on every installation that does not need writes.
- **Why FHIR is different from the existing API, in Brady's framing:** both can write, but the project is actively encouraging FHIR, and for ONC-certified deployments the FHIR endpoints are publicly published. Anyone can look up the list. That makes it new territory worth an extra layer.
- **Jerry floated an alternative:** instead of a global, show an alert when an administrator enables a client that requests write scopes. Brady's response: a warning is fine in addition, but the hard gate is what matters, because it stops writes regardless of what else goes wrong — just as FHIR being off entirely protects against flaws in OAuth.

- **Eric Stern's counterpoint:** any weaknesses in the permission layer probably exist in the existing endpoints too, just in less well-formed shape, so the real security posture may not change much by routing writes through FHIR. The better long-term answer is fixing the lower-level layers once, centrally. He suggested treating the gate like a feature flag for an area still in development.
- **Brady's view on that:** it need not be labelled experimental — administrators who genuinely need writes can turn it on. The point is that installations that do not need writes should not carry the exposure.
- **Agreed:** add a global that turns FHIR writes off by default, an administrator warning is optional on top, and verify the API log captures every call. It can be removed later once lower-level protections and regression testing make the project more confident.

Brady's point about public endpoints is accurate and important. Under the 21st Century Cures Act certification rules (45 CFR 170.404(b)(2)), any developer certified to the standardized FHIR API criterion — 170.315(g)(10) — must publicly publish the FHIR service base URLs for all of its customers in a machine-readable format, at no charge, through ONC's Certified Health IT Product List. ONC's Lantern project checks those endpoints daily. The policy goal is patient access without special effort, but the side effect is that every certified deployment's FHIR endpoint is discoverable by anyone. That is a meaningful difference from an internal API that is only reachable if you already know where it is.

The off-by-default gate is a textbook defense-in-depth control: it does not fix any particular flaw, but it caps the damage if a flaw elsewhere is exploited, and it costs nothing for the installations that do not need the feature. It also follows a pattern OpenEMR already uses — FHIR itself is off by default, and system scopes and the password grant have their own separate settings.

Should Some Resources Stay Read-Only?

- **The standard API already allows writing to almost everything**, with only around three resources strictly disallowed. Writes through the standard API arrived largely because users requested them.
- **Procedure writes are currently not allowed through FHIR**, a decision Stephen Nielson made and Jerry agrees with until auditing is in place. The workaround is ServiceRequest: to record labs or vitals and close the loop, there must be an existing order to attach to.
- **Brady's challenge:** the CCDA import already writes procedures, and if writes to medications and allergies are allowed, why draw the line at procedures? A SMART app placing orders on behalf of a user is a reasonable thing to want.
- **Stephen Waite reframed it:** the real question may be whether to support creating new items only, or also modifying existing ones — and noted that if everything goes to the audit log, and the global gate is there, that addresses much of the concern.
- **Observations:** vitals writes are in. Lab observations are the next PR. Within Observation, categories are scoped separately — a general Observation grant, with each category allowed or not — so vitals can be permitted while labs are not.
- Left for Stephen Nielson's input when he returns.

FHIR Observation categories come from a standard HL7 code system and include vital-signs, laboratory, social-history, imaging, procedure, survey, exam, therapy, and activity. "Surgery," mentioned on the call, is not one of them; surgical events are generally represented as Procedure resources. Scoping writes by category is exactly what SMART v2 granular scopes allow, as discussed on 9/29 — a scope like `user/Observation.c?category=vital-signs` grants create access only for vitals.

The ServiceRequest workaround mirrors FHIR's own workflow model, where a request (ServiceRequest) is fulfilled by events (Procedure, Observation, DiagnosticReport) that reference it. Requiring an existing order before results can be attached is safer, because a result cannot appear in the record without a corresponding clinical decision. It is also why procedure writes are pivotal: without them, a SMART app cannot complete a lab or vitals workflow end to end on its own.

Auditing — Jerry's Three-Phase Plan

- **Jerry has opened an epic issue describing a three-phase plan** and asked people to review and comment on it, since he is working on it largely alone and does not want to be solely responsible for something this important.
- **Phase 1 — fix the existing portal audit.** Jerry wrote it about eight years ago; it stores mostly serialized data, and both he and Stephen Nielson dislike the backend. It needs new tables and refactoring. This is the proof of concept.
- **Phase 2 — hold-and-review for patient-facing SMART apps.** Writes from patient-facing apps — questionnaires especially — would go through the audit engine for clinician approval before reaching the chart. Not intended for all FHIR writes, which would not be feasible.
- **Phase 3 — FHIR write permissions in scopes and endpoints,** along with logging specific to writes.
- **Progress:** an initial PR is up with services, plan services, new database tables, and a restructured version of the existing activity table.
- **The motivating workflow:** a referral to, say, a physical therapist goes out with a questionnaire to track the therapist's work, reported back. At some point the patient also completes a questionnaire — through the portal via login or a one-time token, or ideally through a SMART app. That requires a patient-facing grant type and monitoring of what comes in.
- **Jerry's underlying point:** auditing and logging are the only real protection once writes are allowed. Administrators must be given tools to monitor writes, and must accept that responsibility. His questionnaire manager — letting practices control which questionnaires appear on the dashboard — is on hold behind this.
- **Brady's caveat on review burden:** for user-context writes, the logged-in user is already accountable, exactly as when they edit through the normal interface. The audit trail lets you identify misuse after the fact. Hold-and-review makes most sense at the patient level.
- **Jerry's counter:** administrators are already overloaded, will not read logs proactively, and may approve a client app without researching what it does. Legally, responsibility for PHI falls on the practice, and could fall back on OpenEMR — Stephen Nielson's long-standing point.

The workflow Jerry describes closely resembles HL7's closed-loop referral standards — the Bidirectional Services eReferrals (BSeR) implementation guide, and the Gravity Project's SDOH Clinical Care guide, both built on ServiceRequest, Task, Questionnaire, and QuestionnaireResponse. It also connects to two other threads in these meetings: Char Miller's SDOH referral project (7/14) and Stephen Nielson's closed-loop referral work in Ireland. All three are trying to solve the same problem, and could plausibly share infrastructure.

FHIR has standard resources for this layer that the audit design could align with. AuditEvent records security-relevant activity — who did what, when, from which application. Provenance records where a piece of data came from and who is responsible for it, and can mark patient-reported data as such. The Structured Data Capture (SDC) implementation guide governs how Questionnaires are rendered and how QuestionnaireResponses can be extracted into discrete chart data — which is precisely the point at which clinician review belongs.

On the regulatory side: HIPAA's Security Rule requires audit controls that record and examine activity in systems containing electronic PHI (45 CFR 164.312(b)), and the ONC certification criteria require recording auditable events in a tamper-resistant way and producing audit reports (170.315(d)(2) and (d)(3)). So a reliable, complete API log is not only good practice — it is part of what certification and HIPAA already assume is in place.

Checking the API Log

- **While the discussion continued, Brady had AI inspect the API log.** An initial check suggested every FHIR write in a Synthea test with three patients was captured, including request and response bodies.
- **Closer inspection turned up some issues in how the API log records requests,** which Brady will address. The broader conclusion: the log adheres to the spirit of the requirements but is more complicated than it needs to be, and is due a revisit.
- **Brady's observation:** any time AI looks at something, it finds bugs. Jerry likened chasing AI-found bugs to chasing a cat on crack.
- **A related limitation raised:** logging database activity requires a database connection, which creates awkward circular dependencies.
- **A question was asked about automatically detecting suspicious database activity** — bulk access across many patients, or deletes — and shutting it down. Brady's answer: very hard in practice, because legitimate users and authorized apps also access and modify many records in the course of normal work.

Anomaly detection of this kind does exist in large health systems — often called privacy monitoring or user behavior analytics — typically flagging patterns like a user viewing records of patients with whom they have no care relationship, or accessing records of coworkers or public figures. It works on the audit log rather than blocking in real time, precisely because of the false-positive problem Brady described. It depends on a complete, trustworthy audit log as its input, which is another argument for getting the API log right first.

Deprecating Multisite and Subdirectory Installs

- **From last week's notes:** deprecation decisions on multisite, subdirectory installs, and pushing toward Docker were to be revisited once Stephen Nielson returns on October 13.
- **Stephen Waite posted to the multisite forum thread the night before**, the first comment there since April. In looking into it he found a background services bug in which code assumed a single site and broke under multisite — Jerry noted this was a new bug, apparently from the Symfony command work.
- **Jerry likes multisite** and has seen as many as four multisite installations in production without real problems, but recognizes security is the main argument against it. He does not oppose removal — likened it to `backup.php`, which works for small sites but breaks badly at scale — and said he would stay out of the decision.
- **Stephen Waite:** a soft decision had been made previously, but the only follow-through was a plan to gather more feedback. The reasons: it was limiting long-term development, and any additional security boundary is valuable while so much depends on application code.
- **Eric Stern:** the sooner a clear decision is made, the sooner foundational work is unblocked. Once authentication and authorization are enforced in consistent middleware — in one place rather than 300 — the project can be far more confident exposing features like FHIR writes. That ties the deployment work directly to the security concerns raised earlier.
- **Who is affected:** probably mostly smaller vendors and managed service providers hosting several practices, plus practice groups or do-it-yourself installs using separate logins per facility — sometimes specifically so providers at one site cannot see another site's patients.
- **The migration path is not hard:** copy each site into its own path or container, set its database credentials, and give each its own domain. Ten sites means nine copies. With Docker, it is several containers instead of one.
- **Subdirectory installs may matter more than multisite from a security standpoint**, and will affect the demo farm. Most of the problems are solved by one rewrite rule in the Apache or Nginx configuration, and the code layout on disk does not need to change.
- **Avoid making people migrate twice:** a single plan should take multisite users directly to separate domains rather than separate paths first and domains later.
- **Historical comparison:** the 3.0-era change that embedded access control affected nearly every installation and required awkward manual steps. This would affect a much smaller group.
- **Telemetry does not track multisite**, which is part of why the issue has sat uncommented for six months. Jerry has telemetry through August and may be able to infer multisite use from click data.

The security argument against multisite is about isolation boundaries. Multiple sites in one installation share a single codebase, a single web server process, and often a single filesystem, so a flaw in one site's access control, or a misconfiguration, can expose another site's data. Separate containers or separate domains put an operating-system or browser-level boundary between tenants — separate domains matter because browsers scope cookies by domain, so sessions cannot leak across sites the way they can across paths on the same host. That is the same reasoning behind the front controller's assumption of running at web root.

Agreed Approach

- **Eric Stern will draft a migration guide** for the wiki: if you are running multisite, outside Docker, or from a subdirectory, here are the steps to reach the intended deployment shape. To be refined with Stephen Nielson next week.
- **Deprecation timeline:** announce it as deprecated in one release with warnings, state it is going away in the following release, so nobody is blindsided. The guide gives forum responders something to point to.
- **A newsletter announcement** to the mailing list once the guide is ready, giving a date after which these configurations are no longer supported.
- **Automation is not realistic**, but clear documentation is.

Telemetry — Users Moving Off XAMPP

- **Jerry has noticed in telemetry over recent releases that Windows users are moving from XAMPP to WSL and running Docker**, often reaching production through a proxy. He meant to report it to the board and will include it in his next report.
- **A possible cause:** Stephen Waite's forum post on migrating off XAMPP. It was meant to remain unlisted as a draft since the writing was AI-generated, but was briefly live, and the email notification went out with a direct link. He plans to bump and update it for 8.4.1.
- **Background:** XAMPP stopped at PHP 8.2, as noted on 8/18. Jerry still maintains a XAMPP environment specifically to keep up with users on it, and finds Docker on Windows very easy. XAMPP was always intended as a local development tool, though people ran it in production anyway.
- **Jerry's broader telemetry point:** several areas need tracking that currently are not, and updating telemetry has been on his list. Eric: the absence of that data is exactly why some decisions have stalled.

Jerry referred to “WSL3”; the current version is WSL 2, the Windows Subsystem for Linux, which runs a real Linux kernel in a lightweight virtual machine and is what Docker Desktop on Windows uses under the hood. The trend he describes is encouraging for the deprecation plan: if the Windows XAMPP population is already migrating to Docker, the number of installations stranded by dropping non-Docker paths will be smaller than feared. It is also the first concrete evidence these meetings have seen that a forum post measurably changed user behavior.

Wiki Accounts

- **Wiki accounts are created manually by admins** since open registration was turned off after a flood of spam. No email connection; the password is sent by email. Eric will request one to draft the migration guide.

Manual account creation is the most common anti-spam measure on small MediaWiki installations. If request volume grows, the ConfirmAccount extension offers a middle ground: users submit a request form, and an administrator approves it, without opening free registration.

Jerry's Focus — The Clinical Workflow

- **Jerry was candid about his motive:** everything he does on writes and audits leads back to the clinical workflow — the patient questionnaire side in particular. He apologized repeatedly for going off on tangents.
- **What bothers him is not look and feel** but whether a provider finds OpenEMR useful from start to finish of an encounter. Encounters tend to end at documentation. Even with clinical notes and care plans, there is no good way to record whether a care plan goal is being met when the provider does not see the patient again for six months.
- Next steps for him: merge the CCDA work, then the Node removal for CCDA.

Jerry's care-plan point names a real gap in most EHRs, not only OpenEMR. Goals are documented at a visit but progress happens between visits, when the patient is at home or with another provider. Patient-reported outcomes collected through questionnaires — the very capability his audit work enables — are the standard way to fill that gap, which is why his work threads together as it does: patient-facing writes require auditing, auditing enables questionnaires, and questionnaires make longitudinal care plans measurable.

Items Discussed and Proposed

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

Area	Discussed / Proposed
CCDA in PHP	Implementation-guide fixes in progress; merge expected within a day, followed by the Node removal PR. Document generation and provenance to be fixed in a separate PR.
CQM	The last Node service. Jerry's next priority.
FHIR write gate	Agreed: a global that turns FHIR writes off by default for 8.5. An admin warning on enabling write-scoped clients is optional on top.
API log	Verify every API call is logged correctly. Issues found during the call to be fixed.
Audit epic	Jerry's three-phase plan: fix portal audit, hold-and-review for patient-facing apps, write permissions and logging. Review and comments requested.
Procedure writes	Currently disallowed. Whether to allow them, and whether to allow modifying existing records, left for Stephen Nielson.
Deprecations	Eric to draft a wiki migration guide for multisite, non-Docker, and subdirectory installs. Refine on 10/13, then deprecate over two releases with a newsletter announcement.
Telemetry	Users moving from XAMPP to WSL and Docker. Multisite usage not currently tracked; telemetry updates needed.
XAMPP post	Stephen Waite to update and re-promote the XAMPP-to-Docker forum post for 8.4.1.
Wiki access	Accounts are created manually; Eric to request one.

Notes

- **The connecting thread:** the FHIR write discussion and the deprecation discussion turned out to be the same discussion. Writes feel dangerous because authorization is enforced in many scattered places; consolidating deployment shape is what makes centralized enforcement possible; and centralized enforcement is what would let the project relax the write gate later. Jerry wants clinical capability now, Eric wants the foundation first, and the off-by-default gate is the compromise that lets both proceed.
- Community members who run multisite installations, or who host several practices on one OpenEMR instance, are the people most affected by the coming deprecation and would be especially valuable reviewers of the migration guide once drafted. Likewise, anyone with experience of FHIR auditing or patient-facing SMART apps could usefully comment on Jerry's audit plan.