adaptcx

ARCHITECTURE GUIDE

Audit-by-Design

A blueprint for AI in regulated work, with architectural patterns designed for serious audit and assurance review.

A technical blueprint for making AI activity reconstructable, attributable and defensible. AdaptCX builds on an earlier, smaller version. The expanded platform has not yet been deployed in full at a customer site.

GUIDE MAP

Contents

What auditors actually check 3. Hash chain and time anchor Pattern: vendor-owned export Read access to logs is separated from write access by IAM

SECTION

What auditors actually check

Most AI vendors talk about "audit logging" as if it's a feature you turn on. Real auditors don't think that way. When an Inspector General, OIG, state auditor, or external SOC 2 assessor reviews your audit trail, they evaluate seven specific properties. Bolt-on logging fails most of them consistently. The seven things auditors actually check - Completeness - does every customer-facing interaction produce a log entry, and can the system process a request that doesn't get logged? - Tamper-evidence - can the system prove that an entry hasn't been altered after the fact, in a way that holds up to a second auditor? - Time-stamp integrity - do timestamps come from a trusted source, are they monotonic, and can someone backdate an entry? - Identity binding - does each entry name the human, service account, or system that caused it, and is that identity verifiable from outside the log? - Retention - are logs retained for the regulatory period and deleted on schedule, with deletion itself logged? - Access control - who can read, write, and delete logs, and are those roles separated? - Replayability - given any log entry, can you reconstruct the exact prompt sent to the model, the retrieval set, and the response? The three architectural commitments Audit-by-design starts with three commitments that have to be made before any code is

SECTION

What auditors actually check (continued)

written, not after. Each one rules out a class of failure modes that bolt-on logging cannot prevent 1. Write before display Every customer-facing response must be written to the audit log before it is exposed to the user. Not after. Not concurrently. Before. If the audit write fails, the response is not delivered. This sounds extreme. It is the only posture that survives audit pressure. The alternative - writing concurrently or after - means there is a window where a response was delivered without being logged. A skilled adversary or a system bug can exploit that window. The auditor will ask whether you can prove the window doesn't exist. If your architecture allows it, you cannot prove that. Implementation: serialize the response, hash it, write the hash and body to a tamper-evident store, wait for confirmation, then return the response. The latency cost is real - usually 30-80ms for high-throughput log paths. The latency is worth it. 2. Customer-owned sink

SECTION

3. Hash chain and time anchor

The audit log lives in a sink the customer owns and the vendor cannot write to outside the prescribed write path. A proposed deployment can use a customer-controlled logging sink. Write, read and administrative permissions must be designed, configured and tested for that environment. This separation is what lets a customer's auditor trust the log. If the vendor writes to its own sink and exports a copy to the customer, the auditor has to trust the export is faithful. If the vendor writes directly to the customer-owned sink, the auditor only has to trust the cloud provider's write semantics. Each log entry includes the hash of the previous entry, forming an append-only chain. The chain head is anchored periodically (at an interval selected and verified for the deployment) by signing the head with a key escrowed in the customer's KMS. Tampering with a historical entry would require recomputing every subsequent hash and re-signing every subsequent anchor - and the original signed anchors live in the customer's audit storage, where the vendor cannot reach them. This is overkill for most enterprise SaaS. It is the bare minimum for regulated AI. The cost is one cryptographic hash per write and one signing operation per anchor interval - negligible at any reasonable throughput. What gets logged The temptation is to log selectively -

SECTION

3. Hash chain and time anchor (continued)

only the "interesting" interactions. This is the bolt-on failure mode in disguise. The audit chain logs everything customer-facing, end to end - The inbound request: channel, identity, timestamp, raw payload (or hash if sensitive) - The retrieval set: which documents and chunks were retrieved, with similarity scores - The deterministic-rule match: which playbook (and version) matched, if any - The model invocation: model name, version, prompt template version, token counts, request fingerprint - The model response: full text, with PII redaction tokens replaced by reversible references - The post-processing: redaction applied, citations attached, guardrail checks run - The delivered response: the exact bytes shown to the user - The user action, if any: did the agent accept, override, or edit the suggestion If you cannot reconstruct exactly what happened in any past interaction - including the model prompt and the retrieval set - you do not have an audit chain. You have a history view. Patterns we explicitly avoid Pattern: log to the application database

SECTION

Pattern: vendor-owned export

The application database is operational. It gets restored from backups. It gets schema-migrated. It accumulates index writes. Treating it as an audit store conflates two failure modes - operational and forensic - and the operational concerns always win in a tradeoff. The audit store has to be dedicated, append-only, and out-of-band of the application database. Some platforms write logs to the vendor's sink and export to the customer on a schedule. This breaks tamper-evidence: the customer's copy is downstream of the vendor's, and the vendor's copy is the canonical record. The customer's auditor is reviewing a derivative artifact. The fix is direct write to the customer-owned sink, with the vendor's observability needs handled by a separate, less-sensitive telemetry stream. What this gets you In the proposed AdaptCX architecture, an audit record supports review of AI Engagement Hub activity. Every coaching cue, every knowledge suggestion, every memory-graph lookup, every wellness signal, every root-cause cluster, and every playbook execution lands in the same chain - with the same hash, the same anchor, the same customer-owned sink. One audit log across six AI capabilities, all of which run inside your tenancy The point of audit-by-design is not to satisfy the auditor. The point is that when something goes wrong - a bad model

SECTION

Pattern: vendor-owned export (continued)

response, a customer complaint, a regulator inquiry - you can answer in hours instead of weeks. You can pull every interaction touched by a specific prompt template version and see what changed. You can prove that a specific user's data was never seen by a specific feature. You can show a hearing officer that the AI suggestion they overrode was logged with the override and the reasoning. Bolt-on logging cannot do any of this. By the time you need the audit chain, it is too late to build it. An implementation checklist If you are designing or evaluating AI for regulated work, walk through this checklist. Each item produces a yes/no answer. Any 'no' is a place to harden before going live - Every customer-facing response writes to the audit log before being delivered, with confirmation gating delivery - The audit log lives in a sink the customer owns, not the vendor - The vendor cannot read, modify, or delete log entries - only append - Each entry includes the hash of the previous entry, forming a chain - The chain head is signed at a periodic interval with a customer-controlled key - Timestamps come from a trusted source, are monotonic, and cannot be backdated

SECTION

Read access to logs is separated from write access by IAM

- Each entry binds to a verifiable identity - service account, user UID, or system identifier - Logs are retained for the regulatory period and deleted on schedule, with deletions logged - - Given any log entry, you can reconstruct the exact prompt, retrieval set, and model response - Logs persist independently of operational database backups - PII in logs is redacted but reversibly - references can be resolved by an authorized party for investigation

NEXT STEP

Continue the conversation

If this guide surfaced questions about your architecture, compliance posture, deployment model or AI operating model, talk with an AdaptCX engineer or solutions architect.

Talk to AdaptCX →