ChatGPT

CRDT-Disconnected (Offline) Mode

ANVCS is designed for a **local-first** workflow: developers and AI agents can work fully offline, with the system falling back to a local CRDT session whenever the CRDT "room" or network is unavailable [1] [2]. In offline mode, **every** `ai-vcs commit` **writes to the local DAG and Git repository just as if connected**. In practice this means agents collect local CRDT operations and, on commit, snapshot the state into a new DAG node *and* create a corresponding Git commit (content-addressed, with AI metadata) with no difference in format. For example, the design specifies a "dual-write" on commit: the DAG is updated and a Git commit (with AI metadata in notes) is created [3]. All AI metadata (prompts, model, test results, etc.) is captured and stored in the same schema (e.g. in a sidecar `.ai/` folder or Git notes) whether online or offline [4] [5]. In short, **offline commits are indistinguishable from online commits**: the `.git/` history grows immutably with new SHA-hashed commits and the `.ai/` metadata is recorded in identical form [3] [4].

- **Local-First Workflow:** By principle, ANVCS operates fully locally by default, with optional sync. The design explicitly calls out "Local-First, Cloud-Ready: Fully local dev loop" [1]. If the CRDT server or peer network is unreachable, agents fall back to a local CRDT "room" on the same machine (or local network). CRDT operations continue to commute locally, and commits append to the DAG as normal [3].
- **Dual Writing:** On every commit, ANVCS **simultaneously updates the in-memory DAG and writes a Git commit** (with AI metadata) to disk. This means even offline commits appear in `.git/objects` and `.git/refs` as in a normal Git repo [3] [6]. The commit metadata schema (prompt text, model/version, test report, etc.) is unchanged [4]. Tools that read the `.ai/` folder or Git notes see exactly the same format regardless of connectivity.
- **Seamless User Experience:** There is no special "offline-commit" command. The CLI behaves identically: `ai-vcs commit` always creates a local commit, online or offline. When disconnected, the CLI simply skips any network sync steps and finishes the commit locally [2]. Developers do not need to change their workflow. An optional `ai-vcs rejoin-room` (or simply reconnecting) can be used later to resume sharing ops, but is not required at commit time.

Fallback Mode Equivalence

In offline fallback mode, **all metadata and snapshots use the same structure as connected mode**. For example, each commit still carries the full AI context (prompt, model, test results, timestamp) in a JSON/YAML schema [4] [5]. The commit graph structure is identical – new commits attach to parent(s) in the DAG, and no metadata fields are omitted or altered. As the design notes, every commit is content-addressed and append-only [6] [7], so offline commits are fully Git-compatible. In practice, the `.ai/` sidecar directory remains mirrored in the local repo during offline work; those files are simply updated alongside code changes and committed in the same way. This ensures **feature parity**: any tooling that reads `ai-vcs log` or inspects metadata will see no difference between an offline commit and an online one [4] [3].

CLI Flow & `rejoin-room`

The CLI is explicitly designed for **implicit fallback**. In normal use, `ai-vcs init` and `ai-vcs commit` do not require a running server: the local CRDT engine and Git mirror handle actions. If a CRDT room URL is configured, the client will attempt to connect, but failure to reach the room simply leaves the client in "offline" mode (often a local room already exists) without error. In other words, **no separate fallback-commit command is needed** – the same `ai-vcs commit` works whether online or not [2]. When connectivity is restored, `ai-vcs rejoin-room` (or re-running the agent/IDE) will publish all locally buffered operations and commits to peers. During rejoin, ANVCS reconciles by merging DAGs/ops as usual (see below).

- **Pure CLI + Local Files:** ANVCS explicitly supports a workflow with no long-running daemon. All state is on disk (`.git/` + `.ai/`) and in-memory (for the local CRDT session). The documentation states: "Pure CLI + local files: `.git/` + `.ai/` on disk; no always-on daemon required" [2].
- **Rejoin Command:** While commits happen offline, an optional command (e.g. `ai-vcs rejoin-room` or simply restarting with the correct room URL) triggers pushing local commits/ops to the network. This step uses the same DAG merge logic as normal sync. If multiple offline clients rejoin simultaneously, their commits become separate branches (or merge commits) in the DAG, to be reconciled by the semantic merge engine.

Airgapped / No-Egress Environments

ANVCS supports running in completely isolated networks (e.g. corporate airgaps) by treating the local repository as the source of truth. In such setups, no external network calls are needed. The design provides for **local or private storage** of artifacts and metadata [8]. For instance, a local MinIO or on-prem S3 can be used for blob storage with versioning; these are addressable without public egress. The documentation even includes test matrices for such scenarios: for a "Local laptop" or "Codespaces + MinIO", the system works "offline with local cache; sync later" and versioning is supported [8].

- **Testing Offline Modes:** CI/test setups should include cases where the network is disabled. Agents should still pass tests and commit successfully. Any attempt to contact cloud services should fallback to local caches or fail gracefully. ANVCS's design plans mention "minio sidecar" and VPC endpoints so that in closed environments artifacts are still stored/versioned without public internet [9] [10].
- **Expected Behavior:** In an airgapped network, users can `commit` and run `ai-vcs graph`, `ai-vcs log`, etc., with all functionality intact. Sync commands (`push` / `pull`) simply operate against private remotes (or be skipped), and artifact upload/download operations use local endpoints. The behavior matches connected mode as closely as possible: commits include the same "VersionId" references for artifacts, but in this case those point to local/private stores [9].

Git Immutability (Local-First Principles)

ANVCS **preserves Git's core guarantees** even offline [11]. All commits are content-addressed (SHA-identified) and **append-only** [6] [7]. Offline work does *not* introduce any mutable state into the repo. This was a core design principle ("Git-Compatibility First" and "Immutability & Reproducibility" [12]). In practice:

- Every commit (online or offline) includes a pointer to its parent(s) and a hash of the tree/content, ensuring DAG immutability [13] [6].
- The `.git/` folder is never rewritten: offline commits simply add new objects and update refs. Old commits remain intact and immutable.
- When offline changes are merged later, they become new merge commits rather than rewriting history. Conflicting edits produce new DAG branches or merge nodes; nothing is ever force-pushed over existing history.

By following Git's model strictly (dual writing to `.git/` [3] and never altering past commits), ANVCS retains all standard notions of provenance and integrity in an offline setting.

Git and .ai Metadata Compatibility Offline

ANVCS's `.git` data and AI metadata files are kept fully in sync in offline scenarios. The `.ai/` folder (or commit notes) is simply versioned alongside the code. The design specifies that metadata should live in sidecar JSON/YAML files that correspond to commit IDs [4] [5]. In offline mode:

- The CLI still generates the same metadata schema (prompt, model, test results, etc.) and writes it to `.ai/` files at commit time.
- The mapping from commit ID to `.ai/metadata` is maintained locally. On reconnect, those metadata files are included when syncing (via `git push` or CRDT merge).
- Any large artifacts are handled by the same storage adapters. For offline use, adapters cache uploads locally (e.g. storing in a local MinIO) and record version IDs. When rejoining, those version IDs ensure the correct artifact versions are identified [9].
- Checkouts (`ai-vcs checkout`) in offline mode can still "hydrate" the workspace by reading the pinned content-hash and metadata from the local repo [14]. This means users can branch or roll back offline just as in a regular Git repo.

Overall, the **development environment remains consistent**. Files under `.git/` and `.ai/` are simply the authoritative offline snapshot; nothing is lost. ANVCS even envisions an optional "local blobstore" mode so that even without network, artifact versions are stored locally and later synchronized [8].

Rejoin: Visibility & Conflict Resolution

When offline changes are later shared, ANVCS handles them via its CRDT and semantic-merge machinery. Because all offline operations were CRDT-based (insert/delete ops, file renames, etc.), rejoining is an **eventual-consistency merge of ops** [15]. Concretely:

- **CRDT Merge:** On rejoin, the client publishes all buffered ops and DAG nodes. Other clients (or a central server) union these ops into their local CRDT state. Because CRDTs are conflict-free,

concurrent edits converge: “all ops are commutative and mergeable” [15] . Even if two agents edited the same file offline, their operation logs will merge into a consistent sequence.

- **Semantic Merge:** If both agents happened to create commits on overlapping lines, the semantic merge engine is invoked as usual [16] . This might produce an automatic merged commit or flag a conflict. In any case, the resolution process is identical to online: AST-/LLM-assisted merging produces a new DAG node merging the branches. The key point is that **being offline does not change merge logic**; it just delays it. The same DAG-based merging and optional LLM assistance apply once ops meet.
- **Visibility:** Offline branches become visible as soon as merged. For example, if Alice and Bob both made offline commits on “feature/foo”, when they rejoin their two separate branches appear in the combined DAG. A reviewer sees two parent nodes and a merge commit (or two independent streams) exactly as if they had diverged online. All AI metadata (prompts, tests) from each offline commit are preserved, aiding reconciliation.

In essence, **disconnected changes rejoin the DAG smoothly**. The eventual DAG is the conflict-free union of all offline work, augmented by the same semantic-merge strategies already in the design [15] [16] . No special “reconciliation mode” is needed beyond what ANVCS already provides for concurrent branches.

Sources: ANVCS’s design documentation explicitly envisions a local-first, Git-compatible workflow [1] [11] . It describes dual-writing commits to Git and a CRDT DAG [3] , storing AI metadata in sidecar files [4] [5] , and running purely via CLI without daemons [2] . These principles extend naturally to disconnected operation. Additionally, CRDT literature affirms that operation-based CRDTs are *meant to work independently (including offline mode)*, with all changes merging later without loss [15] [16] . The updates above integrate these ideas into the ANVCS conceptual model and CLI, ensuring that offline usage is formally supported.

---

[1] [2] [3] [6] [7] [8] [9] [10] [11] [12] [14] anvcs_ai_native_version_control_system_problem_design_vision_v_0 (2).md
file://file-RbeWrgh2jDsM1fzq8CFar2

[4] [5] [13] [15] [16] AI-Native Version Control System (ANVCS) – Architecture and Prototype Design.pdf
file://file-WDw4PaToMK4RNWhR1FUJju