Handz V1 PRD Bundle

CH28 — Offline Behavior & Sync

Doc ID: **HZ-V1-CH28_Offline_Behavior_And_Sync_R1**

Revision: **R1** (2026-01-02)

Status: **Draft**

Depends on: **CH04, CH07, CH08, CH12, CH15, CH18, CH29, CH30, CH31**

Related: **CH09-CH11, CH20-CH24, CH33, CH34, CH36, CH37**

Supersedes: —

1. Scope and purpose

This chapter specifies exactly how Handz behaves when the device has limited or no connectivity, how data is cached locally, how changes are synchronized to the server, and how conflicts and failures are handled without losing user work. It also defines user-facing UX patterns for offline states (banners, disabled actions, draft indicators, and recovery messaging).

2. Definitions and terminology

Canonical definitions live in CH03. This chapter uses:

- **Online**: device can reach backend services reliably (auth + database).
- **Offline**: device cannot reach backend services, or connectivity is intermittent enough that network calls fail/retry.
- **Degraded**: network exists but is slow/unreliable; app must avoid blocking the user and must queue work.
- **Local cache**: locally stored read copies of server data (e.g., flows, moves, folders).
- **Local drafts**: locally stored user changes that are not yet synced to server.
- **Sync queue**: ordered list of pending operations to replay to server when online.
- **Conflict**: same entity modified in two places since last successful sync.

3. Goals and non-goals

Goals

- Never lose user work due to connectivity issues.
- Make offline state obvious but non-alarming.
- Keep flow-builder and practice usable in typical gym connectivity.
- Keep behavior consistent across screens.
- Prevent abuse/runaway storage via soft caps + warnings (CH30).

Non-goals

- Full offline access to share links opened outside the app.
- Offline collaboration in V1.
- Perfect automatic merges; V1 prioritizes no-loss conflict copies.

4. Offline capability matrix (by feature)

Read = view using cache; Create/Edit = draft locally; Sync = server; Blocked = action cannot proceed.

Feature	Online	Degraded	Offline
Guest mode	Browse demo flows; build sandbox flow; cannot save (CH08).	Same; show connectivity banner.	Browse cached demos if previously cached; sandbox is ephemeral; cannot save.
Auth / account	Sign up / log in / manage account.	Use if token valid; login may fail; retry.	Cannot sign up/log in; app usable only if session already cached.
Moves library	View/search; create/edit custom moves.	Create/edit as drafts; queue sync.	View cached; create/edit as drafts (account required).
Flow builder	Create/edit flows; autosave; practice gating elsewhere.	Edits stored as drafts; queue sync.	Edit cached flows; create as drafts; sync later.
Inbox / imports	View items; save to library (caps CH18).	Queue save operations; avoid duplicates.	View cached items only; cannot fetch new; cannot resolve share links.
Share links	Create/revoke links (CH17/CH25).	May take time; retry; avoid duplicates.	Blocked: cannot create/revoke.
Practice mode	Start sessions; log history; gating per CH25.	Allow if content cached; queue logs.	Allow only if entitlements recently validated and content cached; queue logs.
Media (video)	Pro uploads (CH29). Links shareable; uploads private.	Queue uploads; show pending; allow cancel.	Select/record to attach locally; upload later; sharing uses links only.
Export / deletion	Export data; delete account.	Exports may queue; deletion requires online.	Export cached only; account deletion blocked.

5. Global UX patterns for connectivity and sync

5.1 Connectivity indicator (global)

A persistent, non-blocking connectivity banner appears at the top of the app when the app is in Degraded or Offline state. It must not cover navigation bars or push important controls off-screen.

- **Degraded banner** copy: “Connection is unstable — saving locally.”
- **Offline banner** copy: “Offline — changes will sync when you’re back online.”
- Banner includes a small “Details” link that opens the Sync Status sheet (see §5.3).

- Banner disappears automatically after 5 seconds only if the state returns to Online; otherwise it remains visible.

5.2 Save status (per editable object)

Any screen where users can edit must show a compact save status near the primary title or in the header area.

- States: **Saved** (synced), **Saving…** (local write in progress), **Saved locally** (draft pending sync), **Sync error** (needs attention).
- The status must update immediately when the user makes an edit (optimistic).
- On “Sync error”, show a tappable affordance: “Retry” and “Details”.
- Never show placeholders; copy must be human-readable and consistent across the app.

5.3 Sync Status sheet

A bottom sheet that summarizes sync health and offers recovery actions without forcing users into Settings.

- Shows: current connectivity state, last successful sync time, number of pending operations, and any blocking errors.
- Actions: “Retry now”, “Copy debug info”, “Open storage manager”, “Contact support”.
- If pending operations exceed the soft cap, show a warning with guidance (see CH30 Warning Ladder).

6. Local drafts and autosave

Handz uses a local-first autosave model for content creation/editing. Users should never need to tap 'Save' to avoid losing work. Explicit save buttons may exist for clarity, but they commit to local draft immediately and enqueue sync.

6.1 Draft creation rules

- When editing begins (first change), create or update a local draft record for the object.
- Drafts are keyed by stable object IDs (server ID when known, otherwise a locally generated UUID).
- Drafts store a **draftRevision** counter that increments on every meaningful edit (used for conflict detection and recovery UI).
- Drafts are persisted immediately (no in-memory-only drafts).

6.2 Autosave cadence

- Flow builder: autosave local changes within 250ms after the last edit gesture ends (debounced).
- Move editor: autosave local changes within 500ms after last field change (debounced).
- Practice session: log events instantly; update per-set completion at event time.
- If the app is backgrounded or terminated, flush pending local writes before suspension.

7. Sync queue model

All server mutations are represented as idempotent operations in a local sync queue. When online, the queue is replayed in order. When offline, operations accumulate and the UI reflects “Saved locally”.

7.1 Operation types

- **UPSERT** entity: create or update an entity (moves, flows, folders, settings).
- **DELETE** entity: delete an entity (reversible until synced, see §11).
- **APPEND_LOG**: append practice log events or session summaries.
- **UPLOAD_MEDIA**: upload a local media file to cloud storage and attach its server URL to an entity.
- **LINK_OP**: create/revoke share links.
- **INBOX_OP**: accept an import, save to library, or dismiss.

7.2 Operation record schema (conceptual)

Exact DB schema is implementation-dependent, but the operation record must include these fields:

```
op_id, op_type, entity_type, entity_id, payload_json, created_at, attempt_count,
next_retry_at, last_error_code, last_error_message, depends_on_op_id (optional),
device_id, user_id
```

7.3 Retry and backoff

- Use exponential backoff with jitter for network-related failures.
- Do not retry immediately in a tight loop; respect next_retry_at.
- Show non-blocking toasts for transient failures; escalate only if failures persist or queue grows (CH30).
- Users can always tap “Retry now” from the Sync Status sheet (§5.3).

7.4 Queue ordering and dependencies

- Operations are processed FIFO *within the same entity*, but the engine may interleave entities for throughput.
- If an operation depends on another (e.g., UPLOAD_MEDIA depends on UPSERT Move), store depends_on_op_id and enforce ordering.
- If dependency is missing or failed permanently, mark dependent ops as blocked and surface error state (CH31).

8. Pull sync: keeping local cache fresh

When online, Handz must keep a local cache of core entities so users can work smoothly and so offline mode is meaningful.

- On app launch: perform a lightweight sync summary check. If changes exist, pull deltas.
- On foreground resume: if last sync > 5 minutes ago and network is available, pull deltas.
- While actively editing: do not interrupt the user with forced refresh; apply remote changes in background and surface conflicts only if needed.
- Deltas must be bounded to avoid huge payloads on cellular; paginate if needed.

8.1 What is cached

- Moves (default + custom) and their user-specific overrides.
- Flows (metadata + nodes/edges + sequence metadata) in the user’s library, plus pinned/offline-designated flows.
- Folders and flow organization metadata.
- Inbox item metadata (but not necessarily full payloads if large).
- Practice history summaries (recent window) and streak state.

Media caching follows §10 and CH29 caps.

9. Conflict detection and resolution

V1 prioritizes no data loss and clear user recovery over perfect automatic merges. Conflicts can happen if the same flow/move is edited on multiple devices before syncing.

9.1 Conflict detection signals

- Each entity stores **serverUpdatedAt** and a monotonic **serverRevision** (or equivalent).
- Local drafts store **baseServerRevision** captured when editing started.
- When pushing a draft, if serverRevision has changed since baseServerRevision, treat as a conflict.
- Clock time alone is not sufficient; use serverRevision where possible.

9.2 Default conflict policy

- Do not attempt deep automatic merges in V1.
- Create a **Conflict Copy** so both versions are preserved.
- Naming: “ (Conflict Copy —)”.
- Surface a banner: “We found edits from another device. Review your conflict copy.”

9.3 Conflict resolution UI

When a user opens an entity that has conflicts, show a Conflict Resolution screen or modal with these options:

- **Keep Mine**: keep local draft as main; save server version as a copy.
- **Keep Theirs**: discard local draft as main; keep server; save local draft as a copy.
- **Keep Both**: keep both versions as separate items (default).
- Include: “View differences” (basic metadata diff) and “Undo this decision” (see §11).

Error-state visuals and routing must align with CH31.

10. Offline media behavior (video, thumbnails, links)

Uploads are Pro-only and governed by CH29 (2GB total cap, private-only, not shareable). Offline behavior must still avoid losing selected videos.

10.1 Attaching media while offline

- Pro user may select/record a video while offline. Store a local file reference and mark **Pending Upload**.
- If user is Free, show paywall before allowing selection and do not store media.
- Pending uploads show status: “Waiting for connection”.
- Users can remove a pending upload at any time.

10.2 Upload resume and failure handling

- When online resumes, begin uploads automatically in background.
- Failures retry with backoff (see §7.3).
- If user edits while upload pending, upload attaches to latest draft unless removed.
- If cap exceeded, block upload and route to Storage manager.

10.3 Shareability rule reminder

- Links are shareable. Uploaded videos are not shareable and do not travel through share links/imports.
- Label wherever user adds media: “Uploaded videos stay private and won’t be included when you share.”

11. Undo, revert, and recovery

Offline and sync-heavy apps must allow users to recover from mistakes without fear.

11.1 Local undo vs. version revert

- **Local undo**: undo/redo inside Flow Builder for last N actions. Works offline.
- **Version revert**: revert a move/flow to a prior saved version; requires version cached locally or online to fetch.

11.2 Deletion behavior while offline

- Offline delete marks entity as **Pending Delete**, not immediate destruction.
- Pending Delete items appear in 'Recently Deleted' and can be restored until synced.
- After delete sync succeeds, remove from cache except minimal tombstone metadata.
- If restored before sync, remove delete op from queue.

12. Guest limitations (offline-specific)

Guest gating rules are owned by CH07/CH08; this section specifies offline implications and required messaging.

- Guests cannot save flows (not even locally). Flow building in guest mode is sandbox only.
- Offline guest may explore cached demos if previously cached; edits are lost on exit.
- Before entering Flow Builder as guest, show interstitial: “Guest mode doesn’t save your flows. Create an account to save.” Buttons: “Continue as guest” / “Create account”.
- If guest attempts any action that would create a draft (e.g., custom move), block and route to sign-up.

13. Storage and offline access management UX

Users need a simple way to understand what is stored on device and to control it.

13.1 Storage manager screen (Settings)

- Path: Settings > Storage & Offline.
- Show: Local cache size (non-media), Offline media pending uploads, Cached thumbnails, and 'Pinned for offline' flows.
- Actions: “Clear cache”, “Clear thumbnails”, “Remove pending uploads”, “Manage offline flows”.
- Confirmations explain consequences (clearing cache may require re-download).

13.2 Offline pinning for flows

- Users can mark flows as “Available offline”.
- Pinned flows ensure nodes/edges/sequence details are cached and refreshed when online.
- Link-based videos are stored but playback still requires network.
- Pinned flows count toward the offline cache budget (placeholder).

14. App lifecycle and background sync (iOS)

iOS may suspend background execution; Handz must be resilient.

- On background: flush local draft writes; pause non-critical network operations; persist sync queue state.
- On foreground: re-check connectivity; resume queue; refresh deltas if stale.
- Optionally notify when long upload finishes/fails (CH27).
- Do not rely on always-on background sync; users are never blocked waiting for background work.

15. Security and privacy for offline storage

Offline caches and drafts may contain sensitive training notes and private media. Minimum requirements:

- Auth tokens in iOS Keychain only.
- Encrypt local database at rest (recommended) or encrypt sensitive fields at minimum.
- Protect local media files with iOS file protection.
- Do not log PII or raw content in analytics; use redacted debug logs (CH33).
- On logout, clear local drafts and caches by default (unless user explicitly opts to keep).

16. Edge cases and failure scenarios

- Crash during edit: recover unsynced drafts on next launch and show a “Recovered draft” banner.
- Rapid online/offline toggling: prevent duplicates via op_id idempotency; ensure uploads don't duplicate.
- Clock drift: never rely on client time for ordering; use server revisions.

- Large flows: autosave must remain responsive; do not block UI while persisting.
- Dangling paths allowed; draft validity does not require all branches to terminate.
- Merge nodes with multiple incoming edges allowed; validations tolerate this.
- If user exceeds soft limits offline (too many pending ops), warn but do not delete work; offer export draft JSON as escape hatch.

17. Telemetry and diagnostics

Record minimal health metrics for troubleshooting (CH33).

- Sync queue length (bucketed), retry counts, permanent failures.
- Average time from local save to successful sync.
- Upload success/failure rates and average upload size (no content).
- Conflict events count and chosen resolution option.
- Storage usage snapshots (bucketed).

Users can copy a redacted debug string from the Sync Status sheet to paste into support.

18. Acceptance criteria and test cases

The following tests must pass before V1 ships:

- **Offline edit retention**: signed-in user edits a saved flow offline; edits persist after app restart; status shows “Saved locally”.
- **Queue replay**: offline edits sync when online returns; status becomes “Saved” without duplicates.
- **Conflict copy**: same flow edited on two devices; both sync; neither set of edits lost; conflict copy exists.
- **Pending delete**: offline delete then restore before online; item returns; no server delete occurs.
- **Pro offline media**: Pro attaches video offline; upload completes on reconnect; cap exceeded routes to storage manager.
- **Guest no-save**: guest builds a flow; leaving builder does not save; interstitial prompts account creation.
- **Sync error clarity**: persistent auth failure shows “Sync error” with re-login path; no infinite spinner.

19. Replit / Vibe Coding build prompt for CH28

Paste the following into your Vibe Coding agent. It is behavior-first and must be implemented exactly.

Paste-able prompt:

```
ROLE: Build Handz V1 (iOS-only). Implement CH28 Offline Behavior & Sync exactly.
CONNECTIVITY: Online / Degraded / Offline; Degraded when requests intermittently fail.
GLOBAL BANNER: Degraded: “Connection is unstable — saving locally.” Offline: “Offline —
changes will sync when you’re back online.” Include “Details” opening Sync Status sheet.
SAVE STATUS: Saved / Saving… / Saved locally / Sync error on all editable screens.
DRAFTS: Persist immediately; store draftRevision and baseServerRevision; never keep
```

```
drafts only in memory.
SYNC QUEUE: Idempotent ops with op_id; retry with exponential backoff + jitter; user can
“Retry now”.
PULL SYNC: On launch + foreground resume, pull bounded deltas if stale.
CONFLICTS: Detect revision mismatch; create Conflict Copy; show Conflict Resolution UI
with Keep Mine/Theirs/Both + Undo decision.
MEDIA: Pro-only uploads; allow attach offline (Pending Upload); resume on reconnect;
label uploads as private/non-shareable.
GUEST: Guests cannot save flows even locally; show guest interstitial before builder;
saving routes to sign-up.
SETTINGS: Build Storage & Offline screen with clear cache, manage pinned flows, remove
pending uploads.
SECURITY: Tokens in Keychain; protect local files; clear cache on logout by default.
TESTS: Run acceptance tests from §18; document how to simulate offline/degraded.
```

20. Troubleshooting guide

- **Saved locally forever** → still Offline/Degraded or auth expired → open Sync Status, Retry; re-login if auth error.
- **Duplicate flows** → missing idempotency/conflict rules → ensure op_id and upsert by entity_id; enforce conflict copies.
- **Upload stuck** → background restrictions or cap exceeded → check Storage & Offline; retry on Wi■Fi.
- **Data missing after logout/login** → cache cleared by default → expected by design; confirm settings.
- **Offline copy inconsistent** → enforce exact banner strings in §5.1 across app.

End of CH28.