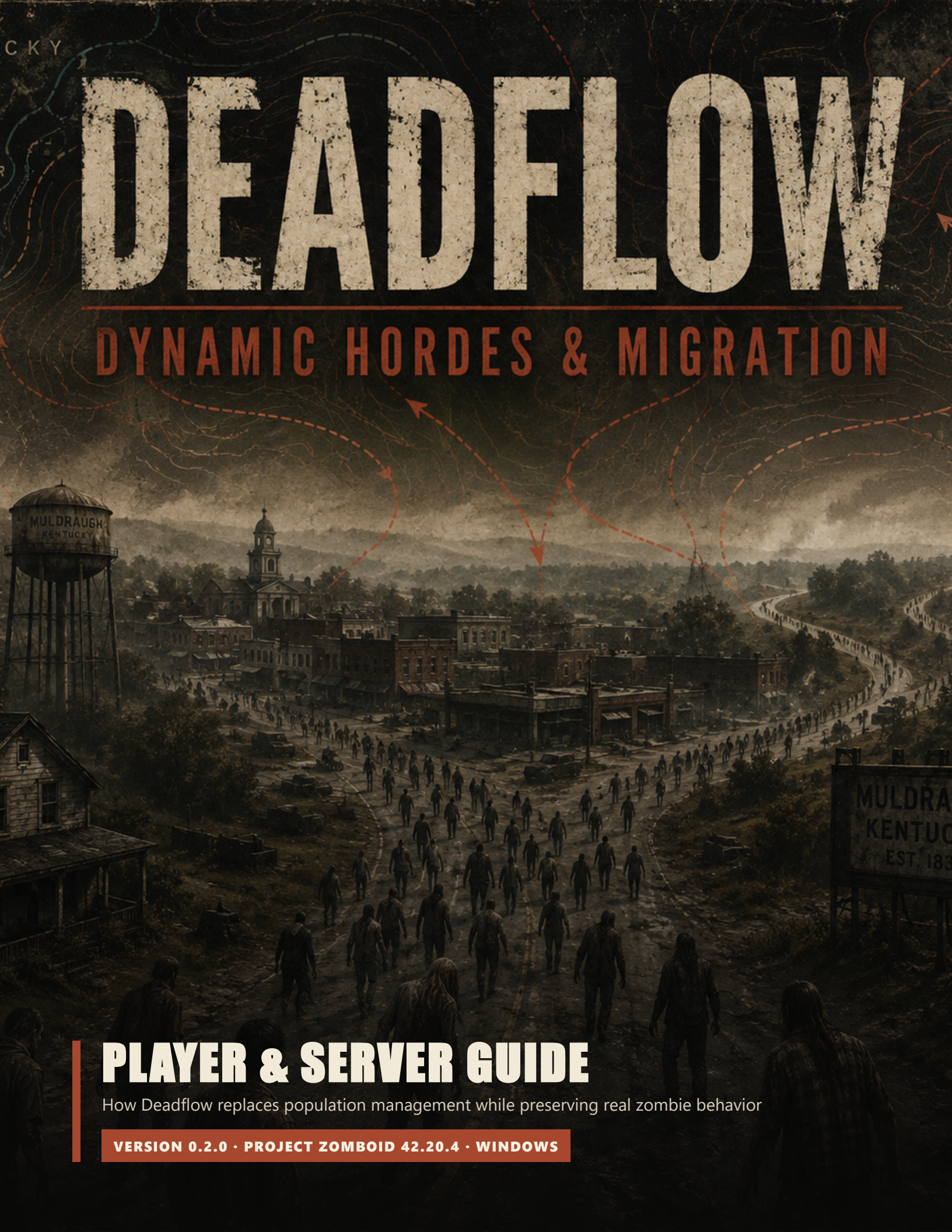


PLAYER & SERVER GUIDE

How Deadflow replaces population management while preserving real zombie behavior

VERSION 0.2.0 · PROJECT ZOMBOID 42.20.4 · WINDOWS

CONTENTS

Deadflow makes the undead population persistent, mobile and accountable. This guide explains what that means in play, how to install it safely, and how servers and compatible zombie mods work with its authority.

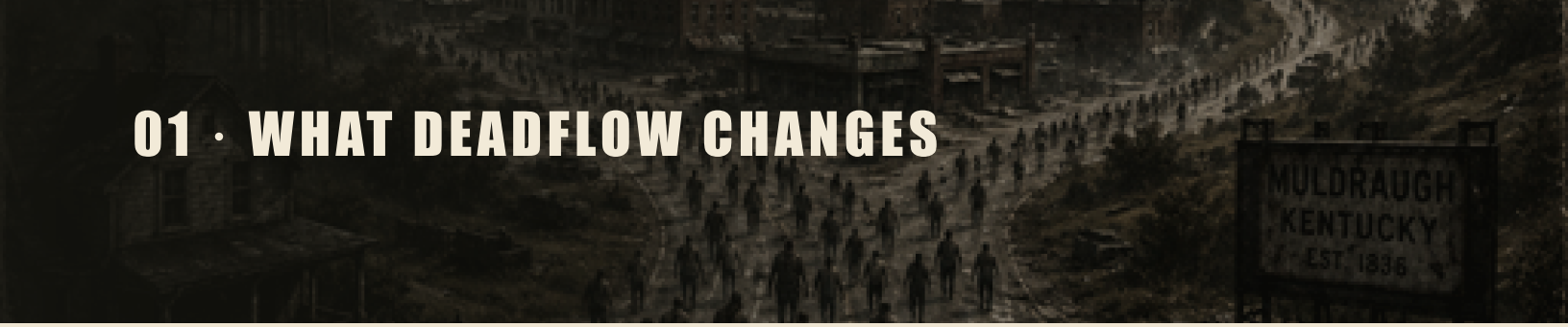
READ BEFORE ENABLING

Deadflow 0.2.0 is for new worlds. It does not import an existing vanilla population save and does not export a Deadflow world back to vanilla. Keep it enabled for the life of a world it owns.

SP + MP
Authority runs in single-player and on the multiplayer server.

FJL
FJL 0.3.4+ and FadedJavaLoaderBridge are required.

B42
Pinned to Windows PZ 42.20.4 in this release.



01 · WHAT DEADFLOW CHANGES

Deadflow becomes the population authority. It replaces vanilla's offscreen population, distribution, migration, respawn and population-save paths, while leaving each real zombie's moment-to-moment behavior in Project Zomboid.

DEADFLOW OWNS

- Regional zombie counts
- When and where population materializes
- Offscreen horde travel
- Growth and respawn birth budgets
- Population persistence and accounting

PZ AND BEHAVIOR MODS OWN

- Real-zombie AI and animation
- Combat, physics and pathing
- Network replication
- Nests, hunts and special behavior
- Mod-specific state through adapters

WHY THIS FEELS DIFFERENT

Clearing an area removes actual residents from the ledger. Deadflow does not silently replace a failed spawn or a migrating group. A quiet area can remain depleted until surviving groups arrive or the configured respawn policy becomes eligible. Noise, density pressure, terrain and mod-supplied influences can redirect movement elsewhere.

> **NO COOPERATIVE POPULATION MODE**
>
> There is one owner. If Deadflow's verified hooks cannot take control, authority does not start. It will not knowingly run beside vanilla as a second population manager.

WHAT "VIRTUAL" MEANS

A virtual zombie is still alive in Deadflow's census, but it is represented as a regional count or saved record instead of a loaded `IsoZombie`. Previously encountered individuals can retain health, outfit, inventory and serializable mod data. They do not receive individual AI updates while stored.

FROM DISTANT COUNT TO REAL ACTOR AND BACK

THE ZOMBIE LIFECYCLE

Regional population › **Loaded legal terrain** › **Real PZ zombie** › **Safe offscreen snapshot**

1. REGIONAL POPULATION

The world is divided into 64×64-tile regions, each holding eight-by-eight chunk counts, density weights and growth history. Initial population comes from the map's real zombie-density headers and the selected vanilla population multiplier.

2. MATERIALIZATION NEAR PLAYERS

When loaded terrain enters the activation area, Deadflow finds legal squares and creates ordinary engine zombies. Creation is capped per update, so a backlog is spread over time rather than delivered as one large spike. New arrivals respect player clearance. In SP, visible squares are also deferred.

3. REAL PLAY

Once materialized, the zombie uses PZ's normal AI, combat, navigation, animation and networking. Zombie behavior mods can continue to operate. Deadflow tracks its stable population identity without replacing the actor's AI.

4. STORAGE

Far enough from every living player, an eligible zombie can be serialized and removed through the normal engine/network cleanup path. Active combat, a visible target, fire, grappling, recent flesh contact, a behavior lease or an iSync lease can keep it real longer.

5. DEATH AND REMOVAL

A death reduces living population once. Administrative removal is tracked separately. Explicit births add population through a durable grant. The same identity cannot count simultaneously as a real zombie, a saved member and a travelling member.

> **FORCED CHUNK UNLOAD**
>
> If terrain must unload, Deadflow takes a durable snapshot before the engine disposes of the actor. Adapters must release transient references during this handoff.

02 · ORGANIC MIGRATION

Migration moves existing survivors. It never creates a replacement population to make a destination look full.

Crowded or disturbed region	→	Collision-checked route leg
Travelling cohort in ledger	→	Revalidated destination

WHAT SHAPES MOVEMENT

NATURAL PRESSURE

Groups prefer lower-density neighboring areas with a slowly changing drift direction. This creates motion without making every horde omniscient.

WORLD SOUNDS

Large sounds can pull population regionally. Deadflow influences population flow; it does not assign a visible target or an AI order to each real zombie.

TERRAIN

Departures and arrivals use the game's native collision service. Loaded ground connectivity prevents arrivals through sealed local geometry.

MOD INFLUENCES

Other mods can register attractors, repulsors and protected nest regions or request a specific population migration.

LONG ROUTES

Long-distance travel is divided into route legs of at most 64 tiles. Only a bounded number of collision queries and arrivals are processed at once. A route delayed by work limits waits; its zombies are not discarded. If an arrival becomes blocked, the group settles at its last confirmed location.

> **BEHAVIOR REMAINS LOCAL**
>
> Deadflow does not add a permanent offscreen player tracker, special zombie powers or its own night hunt. The Hive or another behavior mod may provide those systems and tell Deadflow when a cohort may move.

SAVED INDIVIDUALS INSIDE A HORDE

A travelling group may carry up to 32 previously encountered mobile identities plus anonymous population. Adapter-owned zombies remain anchored unless their adapter explicitly allows ambient migration.

A POPULATION THAT BALANCES

PERSISTENCE & CONSERVATION

The central rule is simple: a zombie must have a recorded origin, and movement must not change how many are alive.

```
initial population + births − deaths − removals
= regional stock + travelling stock + saved/live identities
```

DURABLE FILES

FILE	PURPOSE
`<save>/Deadflow/population.bin`	Atomic CRC32C checkpoint containing the durable population state.
`<save>/Deadflow/population.wal`	Revisioned CRC32C redo log for accepted changes since the checkpoint.

An incomplete final journal frame recovers to the last durable prefix. A complete but corrupt frame fails instead of guessing. Replayed revisions and terminal zombie identities cannot be applied twice.

WHAT IS SAVED FOR ENCOUNTERED ZOMBIES

Deadflow uses the engine's own zombie serialization plus known phenotype fields. This covers inventory, appearance, health and serializable mod data. Runtime functions, userdata and live Java/Lua object references are not portable save data; behavior adapters capture the durable equivalent.

SAVE CADENCE

- Journal writes are batched and forced about once per second.
- Live snapshots refresh every 30 seconds.
- A full checkpoint runs on a normal world save or after a 32 MiB journal threshold.

> **CRASH BOUNDARY**
>
> The ledger is not one atomic transaction with every PZ corpse, inventory and terrain file. Journal damage tests and normal restarts pass, but a forced process or power failure during a world mutation can still roll recent live state back to a durable boundary.

TUNE TRANSITION WORK, NOT TOTAL POPULATION

SANDBOX SETTINGS

SETTING	DEFAULT / RANGE	WHAT IT CONTROLS
Zombies added per update `SpawnPerTick`	8 1–128	Maximum automatic materializations per update. Lower values spread arrivals over more updates; they do not lower the population.
Distant zombie checks `ScanPerTick`	64 1–4096	Maximum active actors checked for safe offscreen storage per update.
Activation distance `ActivationRadius`	112 tiles 48–512	Loaded terrain inside this distance from any living player can restore population.
Offscreen storage distance `SleepRadius`	160 tiles 64–1024	Eligible actors farther than this from every living player can be stored. Effective value is at least activation + 16.
New arrival clearance `SpawnClearance`	32 tiles 16–128	Minimum player distance for new anonymous arrivals. It is clamped below activation distance.

RECOMMENDED STARTING PROFILES

DEFAULT

8 / 64 / 112 / 160 / 32

Start here. Balanced transition rate and hysteresis.

SMOOTHER LOAD

4 / 32 / 112 / 176 / 40

Slower arrivals and retirement checks. The world may take longer to fill nearby.

FAST TRANSITION

16 / 128 / 128 / 176 / 32

Moves more work per update. Test server frametime before keeping it.

Profile order: SpawnPerTick / ScanPerTick / ActivationRadius / SleepRadius / SpawnClearance. These profiles are operational suggestions, not benchmarked performance guarantees.

VANILLA POPULATION SETTINGS STILL USED

Population multiplier, population start/peak, peak day, uniform distribution and respawn hours/fraction/unseen time feed Deadflow's initial density and explicit birth policy. Respawn hours set to zero disables respawn. Vanilla redistribute and rally behavior is superseded by Deadflow's migration policy.

> **RESTART REQUIRED**
>
> Deadflow reads its five settings when the world starts. Restart the world/server after changing them.

03 · SINGLE-PLAYER SETUP

1. **Confirm the supported environment.** Use Windows Project Zomboid 42.20.4.
2. **Install FJL 0.3.4 or newer.** Deadflow's Java hooks must load before the population classes are used.
3. **Enable FadedJavaLoaderBridge.** This is Deadflow's required PZ mod dependency.
4. **Install the Deadflow folder.** Place it inside your profile's `Zomboid/mods` folder. The JAR must remain at `Deadflow/42/java/Deadflow-0.2.0.jar`.
5. **Enable Deadflow and restart the JVM.** A return to menu is not enough after changing Java extensions.
6. **Create a new world.** Configure the Deadflow sandbox page before the first load.

VERIFY AUTHORITY

Trusted Lua/debug tooling can call `Deadflow.status()`. A healthy single-player world reports `AUTHORITY` and lists `nativePopulationJNI=SUPPRESSED`. A missing Java authority message means the loader or dependency did not start correctly; do not continue that world as a Deadflow save.

CORRECT FOLDER SHAPE

```
Zomboid/mods/Deadflow/
├── mod.info
├── poster.png
└── 42/
    ├── mod.info
    ├── java/Deadflow-0.2.0.jar
    └── media/...
```

UPDATING

Back up the world, replace the complete Deadflow folder, keep FJL and the bridge compatible, then restart the game. Do not mix Java files from one Deadflow version with Lua files from another.

REMOVING THE MOD

Version 0.2.0 has no export back to vanilla. Removing or disabling Deadflow from a world it owns leaves vanilla without a valid population conversion.

THE SERVER IS THE ONLY POPULATION WRITER

MULTIPLAYER SETUP

Deadflow server authority › **Normal PZ replication** › **Deadflow client replica**

1. Install matching FJL, FadedJavaLoaderBridge and Deadflow artifacts on the dedicated server and every client.
2. Add `Deadflow` and `FadedJavaLoaderBridge` to the server's mod list using the existing FJL installation instructions and load order.
3. Keep normal Lua checksum and FJL handshake enforcement enabled.
4. Configure Deadflow's sandbox values in the server preset, then start a new world.
5. Restart both server and clients after any Deadflow Java update.

AUTHORITY ROLES

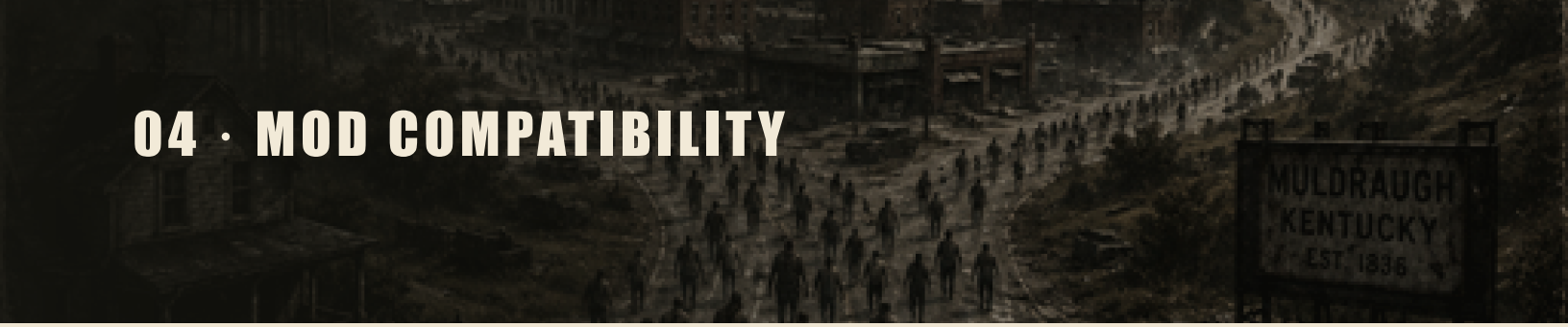
SIDE	EXPECTED STATE	RESPONSIBILITY
Dedicated/host server	AUTHORITY	Ledger, migration, materialization, snapshots and saves.
Connected client	CLIENT REPLICA	Receives ordinary PZ zombie spawn, update and removal packets.

The client does not maintain a competing population ledger. Deadflow keeps the normal network protocol for real actors. An actor stored by the server is first removed through standard network and engine cleanup; restoration creates a new real object/online ID while retaining the same server-side Deadflow identity.

OPERATIONAL ADVICE

- Back up before mod, PZ or FJL upgrades.
- Do not combine Deadflow with another population-distribution replacement.
- Keep activation and clearance distances realistic for the server's loaded-area behavior.
- Monitor save time and transition spikes when many previously encountered zombies have rich inventory/mod data.

> **CURRENT MP EVIDENCE**
>
> A real dedicated server and one real PZ client passed spawn, deletion, restoration, strict FJL handshake, iSync lease coordination and census checks. Multi-client stress, latency/jitter, split-screen, hosted-launcher flow and busy-server bandwidth measurements remain outstanding.



04 · MOD COMPATIBILITY

Deadflow owns population. Other zombie mods can continue to own behavior, provided their persistent and transient state has a safe handoff.

INTEGRATION	STATUS	HOW IT WORKS
iSync 1.3.0	OPTIONAL	Deadflow observes successful authority/recovery leases and delays soft storage while iSync needs the actor real. It does not alter iSync's result or private tables.
The Hive 2.2.1	OPTIONAL	Resident identity, roles, roost records and wounds survive handoff. Active hunts veto soft storage; forced unload cleans transient assignment and path references.
Other behavior mods	API	Serializable modData survives. External runtime state or object references require an adapter.
Population replacements	INCOMPATIBLE	Two systems cannot safely own the same population ledger and native entry points.

ISYNC COORDINATION

iSync is not a dependency. When its reviewed authority class is present, Deadflow adds an optional hook. A successful recovery or sticky-horde lease is treated as a reason to retain the real zombie. Deadflow does not revoke a legitimate continuing lease to meet its own storage budget.

THE HIVE COORDINATION

The Hive adapter marks Hive-owned members as anchored by default. It refreshes existing roosts as protected migration regions; this does not manufacture residents. Forced terrain disposal still snapshots the resident and asks Hive to release transient movement and loaded-object references without recording a false death.

HOW TO JUDGE ANOTHER ZOMBIE MOD

- ✓ Pure serializable fields in zombie modData generally survive.
- ✓ Cached zombie objects, active Java tasks, Lua closures or userdata need lifecycle cleanup and restoration.
- ✓ A mod that constructs zombies outside the standard factory must adopt them through Deadflow's API.
- ✓ A mod that wants anchored residents should claim behavior or provide an adapter that disallows ambient migration.

> **FAILURE BEHAVIOR**
> Required adapter/capture failures halt authority rather than silently discarding the mod's state.

FOR ZOMBIE-MOD AUTHORS

INTEGRATION API

Lua uses the global `Deadflow` wrapper; Java uses `dev.deadflow.api.DeadflowAPI`. API version is 2. Mutation calls are trusted local server/SP operations and must run on the world thread.

OPERATION	USE
`identity(zombie)`	Get the stable owner-side Deadflow identity.
`queryPopulation(chunkX, chunkY)`	Occasional regional census using 8-tile chunk coordinates.
`requestMigration(owner, seq, x, y, tx, ty, count)`	Move existing population asynchronously. It never creates births.
`addPopulation(owner, seq, x, y, count)`	Add an explicit, durable birth grant without immediately creating actors.
`adoptZombie(owner, zombie)`	Account for an actor created outside the standard factory.
`upsertAttractor / upsertRepulsor`	Add a temporary regional pull or push.
`registerNest / removeNest`	Protect an area from another owner's ambient transfer.
`claimBehavior / releaseBehavior`	Temporarily keep an owned real actor from soft storage.

COMMAND RULES

Use one persistent, strictly increasing positive sequence stream per owner across migration and birth grants. Persist the sequence in your own world data. Accepted high-water marks survive restart, so retrying the same sequence acknowledges it without applying the effect twice.

```
local ok, reason = Deadflow.registerBehaviorAdapter(
  "example.behavior", 1, {
    owns = function(z) return ownsZombie(z) end,
    canVirtualize = function(z) return not isBusy(z) end,
    allowAmbientMigration = function(z) return mayRoam(z) end,
    capture = function(z) return durableState(z) end,
    restore = function(z, saved, schema) restoreState(z, saved) end,
    onLifecycle = function(event, z, id) releaseIfNeeded(event, z) end,
  })
assert(ok, reason)
```

Adapters receive lifecycle events for creation, materialization, preparation, virtualization, unload, death and removal. Keep callbacks fast and non-reentrant. Do not create/delete/migrate actors from capture or restore callbacks. Full bounds and return values are documented in `docs/API.md`.

WHAT TO CHECK WHEN SOMETHING GOES WRONG

TROUBLESHOOTING

SYMPTOM	LIKELY CAUSE AND RESPONSE
"Authority unavailable"	FJL or FadedJavaLoaderBridge did not load before Deadflow. Verify versions, enable both components and fully restart the JVM. Do not continue the affected world.
Unsupported build/hash error	PZ classes or the Windows population DLL differ from audited 42.20.4. Use the supported build or wait for a reviewed fingerprint update.
Area fills slowly	A low SpawnPerTick spreads materialization over time. High SpawnClearance or terrain legality may also defer arrivals.
Distant zombies stay real	They may be inside SleepRadius, recently engaged, on fire/grappling, owned by a behavior adapter, or retained by an iSync lease.
Area remains empty after clearing	This can be correct. Survivors must migrate in, or eligible configured respawn must occur. Deadflow does not mint replacements for failed spawns.
MP client mismatch	Ensure server and clients have matching Deadflow, FJL and bridge files. Restart all JVMs and retain checksum/handshake validation.
Adapter restoration stops authority	The owning mod is missing, its schema changed, or capture/restore failed. Restore the matching mod version and inspect logs before loading again.

SAFE RECOVERY HABITS

- ✓ Keep versioned world backups before every PZ, FJL, Deadflow or behavior-mod update.
- ✓ Copy the complete Deadflow package; do not mix its Lua and Java versions.
- ✓ Preserve `<save>/Deadflow` with the rest of the world save.
- ✓ Record sandbox settings when comparing population behavior across restarts.
- ✓ Capture the full server log and Deadflow status line when reporting a problem.

> **DO NOT DELETE POPULATION.BIN OR POPULATION.WAL AS A "RESET"**
>
> Those files are the authority's durable census. Deleting them does not safely convert or repair a world.

05 · EVIDENCE & LIMITS

Deadflow is an implemented takeover, but this release is deliberately narrow about what its tests prove.

42
Native population methods suppressed in transformed-bytecode audit.

1,000
Complete saved individual payloads retained in the empty-server cohort test.

1
Real MP client used for the connected integration cycle.

PASSED INTEGRATION COVERAGE

- ✓ Actual SP loading, player creation, materialization, death accounting and save.
- ✓ Dedicated-server real → stored → restored cycle across three process launches.
- ✓ Collision-checked asynchronous migration with retained inventory and mod data.
- ✓ Real client/server spawn, delete acknowledgement and restoration.
- ✓ iSync 1.3.0 enforce-mode recovery lease coordination.
- ✓ The Hive 2.2.1 resident, role, roost record, hunt veto and forced cleanup.
- ✓ Million-zombie ledger conservation, 10,000 lifecycle operations and damaged-journal recovery.

NOT YET ESTABLISHED

- A percentage FPS, CPU or bandwidth improvement on a busy multiplayer server.
- Multi-client contention, driving/chunk churn, latency/jitter or hosted-server flow.
- Linux/macOS or PZ builds other than the pinned Windows 42.20.4 build.
- Existing-save import, vanilla export, automatic uninstall conversion or full crash atomicity across every PZ save file.
- Every third-party zombie mod without a purpose-built adapter.

> **HOW TO READ THE 1,000-ZOMBIE RESULT**
>
> All 1,000 added survivors were preserved as complete stored payloads with zero real zombie objects left in the empty server. The paired idle sample showed no measurable main-thread CPU gain because vanilla already issued no zombie update callbacks in that scenario. It demonstrates storage and conservation, not a gameplay FPS multiplier.

RELEASE IDENTITY

Deadflow 0.2.0 · Author **?** · Mod ID `Deadflow`
Java artifact SHA-256: 22e6c72b0bc5cb59a412374f8c72e9962cfdc3e01c99331dd841041bfff75e2a

For Java Features Click Here.