Database, Security & Operations

- Overview · Database, Security & Operations
- Database Schema Reference
- Database & Persistence
- Item Definition Precedence
- Security & Server Authority
- Transaction History & Retention
- Backups & Restore
- Audit Log
- Stability Notes
- Troubleshooting Playbook
- Error Reference & Diagnostics
- Upgrade to v2.0.0
- Performance & Production Tuning
- Operations & Forensics Center — v2.0.0
- Transaction Ledger & Anti-Dupe
- Inventory Manager & Snapshots
- Persistent Placed Items & Movement Policies
- Operations Configuration & Limits

Overview · Database, Security & Operations

6 · Database, Security & Operations

Persistence, security boundaries, diagnostics, backup/restore, upgrades and production operations.

v2.0.0 adds Operations & Forensics, live/offline Inventory Manager, snapshots, Anti-Dupe, persistent placements and movement policies.

Database Schema Reference

Database Schema Reference

Persistent tables installed by the current SQL schema.

Applies to: `Nord Inventory v2.0.0`

Current schema tables:

- `nord_inventory_custom_categories`
- `nord_inventory_storage`
- `nord_inventory_custom_items`
- `nord_inventory_item_versions`
- `nord_inventory_disabled_reference_items`
- `nord_inventory_preferences`
- `nord_inventory_transactions`
- `nord_inventory_vehicle_categories`
- `nord_inventory_vehicle_models`
- `nord_inventory_audit_log`
- `nord_inventory_backups`
- `nord_inventory_admin_access`

Do not treat these tables as a replacement for the public SDK. External resources should use exports so Nord can preserve source precedence, audit/history and validation semantics.

v2.0.0 Operations tables

`nord_inventory_snapshots`

Stores inventory key/type/owner, slot and weight limits, reason/creator, full item payload and creation timestamp for recovery workflows.

`nord_inventory_forensics_alerts`

Stores Anti-Dupe findings with severity, code, identity value, item, payload, open/resolved status and timestamps.

`nord_inventory_placed_items`

Maps a persistent world placement ID to its inventory key, owner identifier, coordinates, model and timestamps. The inventory contents themselves continue to use normal Nord Inventory storage.

All three are created with `CREATE TABLE IF NOT EXISTS`; v2.0.0 does not drop or rewrite existing inventory tables during this migration.

Database & Persistence

Database & Persistence

Persistent item/storage state and safe restart behavior.

Applies to: `Nord Inventory v2.0.0`

Database item persistence

Item-definition precedence is strict and restart-safe:

1. If `nord_inventory_custom_items` contains a row for an item name, that row owns the definition.
2. `shared/items.lua` / `shared/weapons.lua` are consulted only if no database row exists for that name.
3. Resource startup is read-only for item definitions. It never seeds, updates, replaces, or deletes DB item rows.
4. A disabled DB row remains authoritative and blocks file fallback.
5. The only way to restore the reference definition is an explicit Revert action, which deletes the DB override row.

This prevents edited base items from returning to file defaults after `restart nord_inventory`.

Item Definition Precedence

Item Definition Precedence

Deterministic reference/database/provider ownership rules.

Applies to: `Nord Inventory v2.0.0`

Item definition precedence

Nord Inventory uses one deterministic rule:

1. **Database row exists for the item name** → the database definition is authoritative.
2. **No database row exists** → `shared/items.lua` / `shared/weapons.lua` is used as reference fallback.

Editing a reference/base item from Admin Studio creates or updates its database row. The reference file is never mutated. Reverting a base override removes the database row; only then does the reference definition become active again.

This prevents a base item from being simultaneously resolved from the file and the database.

Restart behaviour

Resource/server startup is read-only for item definitions. Existing rows in `nord_inventory_custom_items` are never replaced, disabled, deleted, or reseeded from the Lua reference files during boot. Disabled DB rows still own their names and block reference fallback.

Security & Server Authority

Security & Server Authority

Trust boundaries for inventory, admin, robbery, links and integrations.

Applies to: `Nord Inventory v2.0.0`

Core rules:

- never trust NUI/client item amounts, metadata or permissions without server validation;
- use ACE/server-only license data for administration;
- keep owner-only restore separate from routine admin permissions;
- validate player distance/session identity for remote inventory access;
- validate vehicle lock/movement/range continuously during trunk sessions;
- validate external recommended-script URLs as `http/https` on server and NUI sides;
- use provider ownership for runtime registrations;
- do not expose privileged exports through unrestricted net events.

Transaction History & Retention

Transaction History & Retention

Item history and transaction-log retention defaults.

Applies to: `Nord Inventory v2.0.0`

`Config.EnableItemHistory = true` and `Config.TransactionLogRetentionDays = 14` in the current build. Transaction/audit data is operational evidence, not a substitute for full database backups.

Adjust retention based on server volume and your own compliance/storage requirements.

Backups & Restore

Backups & Restore

Operational backup flow and owner-only restore.

Applies to: `Nord Inventory v2.0.0`

Admin Studio can create inventory backups. Restore is owner-gated by default because it can replace persistent state. Before a major migration:

1. take a full database backup outside Nord;
2. create a Nord backup where appropriate;
3. test the migration on staging;
4. stop conflicting inventory writers;
5. migrate and verify counts/metadata;
6. keep rollback data until the server has run cleanly through restart/reconnect cycles.

Audit Log

Audit Log

Administrative traceability for sensitive operations.

Applies to: `Nord Inventory v2.0.0`

The audit log records sensitive administrative activity in `nord_inventory_audit_log`. Use it to investigate who changed definitions or performed operational actions, but keep server/database logs as an additional source of evidence.

Stability Notes

Stability Notes

Runtime guardrails and patterns for production servers.

Applies to: `Nord Inventory v2.0.0`

Nord Inventory stability notes

Large weapons on back

- Up to two long weapons can be displayed at once.
- Cosmetic weapon props are networked so nearby players can see them.
- The equipped weapon is excluded by unique item UID.
- Back props are hidden while dead or inside a vehicle and are restored automatically.
- Inventory synchronization uses the lightweight player snapshot, so it also works with the NUI closed.
- Position/rotation is configurable under `Config.BackWeapons`.

Item-use props

- Hand prop convention corrected to bone `60309` and rotation order `0`.
- Common consumable/utility models are normalized through `Config.ItemPropAttachments`.
- `rotOrder` is now preserved by server sanitization.
- Admin-created hand props use known model calibration when available.

Static validation

- `web/app.js` passed Node syntax validation.
- EN/PT JSON parsed successfully.
- Every static Lua `L('...')` notification key exists in both EN and PT.
- No duplicate HTML IDs detected.
- Referenced manifest files validated.

Native compatibility guards

Nord Inventory no longer requires `NetworkGetEntityIsNetworked` or `SetEntityCompletelyDisableCollision` to exist as Lua globals.

Vehicle control uses normal network-control natives directly. Attached props and back weapons use `SetEntityCollision(false, false)` as the compatible baseline, with complete collision disabling only when the current artifact exposes it.

Network IDs are also read through guarded helpers because ambient NPC vehicles can disappear while an inventory or trunk session watchdog is running.

v2.0.0 runtime hardening

- networked long-weapon back props are removed reliably after ownership migration;
- stale/legacy equipped weapon snapshots fall back to weapon-name matching so the back prop does not remain until relog;
- pending ground-model loads are invalidated when a drop leaves range or is removed, preventing ghost props;
- metadata-aware `CanCarryItem` avoids incorrect stack acceptance;
- `stack = false` requires one free slot per item and creates count-1 instances;
- legacy malformed unique stacks and missing UID/serial/container identities are repaired safely during load;
- server-owned identities cannot be erased/replaced by `SetMetadata`/`UpdateMetadata`;
- failed robbery transfers no longer remove Fast Slot pins;
- same-slot moves are no-op operations;
- transient SQL save failures keep inventories dirty for a later autosave retry;
- player disconnect and resource-restart synchronization include fallback/resnapshot paths.

Troubleshooting Playbook

Troubleshooting Playbook

Fast diagnosis by symptom.

Applies to: `Nord Inventory v2.0.0`

Inventory does not open

- confirm `nord_inventory` is started and named correctly;
- check framework detection/config value;
- inspect NUI/client console errors;
- verify locale JSON is valid;
- verify no other resource owns the same key command unexpectedly.

Items do not persist

- inspect oxmysql/database errors;
- confirm SQL schema is installed;
- ensure the target inventory ID/player identifier is stable;
- check that a third-party resource is not overwriting state directly.

Runtime item registration returns false

- start the provider after Nord Inventory or implement retry;
- canonicalize the item name;
- inspect Provider Registry/System Health;
- verify source precedence/provider ownership.

Trunk closes immediately

- check lock provider state;
- player/vehicle distance;
- vehicle movement threshold;
- network entity/plate identity;
- target vehicle despawn/replacement.

Admin Studio opens but mutation fails

- UI visibility is not permission: check server role/ACE;
- confirm `config/admin.lua` and Nord Staff grant state;
- inspect audit/server logs for rejected action.

Error Reference & Diagnostics

Error Reference & Diagnostics

Where to look before changing code.

Applies to: `Nord Inventory v2.0.0`

Use these in order:

1. FXServer console error and resource/line;
2. client F8/NUI console if the issue is presentation/input;
3. Admin Studio > System Health;
4. Provider Registry for runtime registration ownership/failures;
5. `DebugItem(itemName)` for definition/provider inspection;
6. database table/schema existence and oxmysql query errors;
7. current config value and locale JSON validity.

Enable debug logging only while diagnosing; verbose integration logging is not intended as permanent production noise.

Upgrade to v2.0.0

Upgrade to v2.0.0

1. Replace the resource with the complete v2.0.0 build while keeping the resource folder named `nord_inventory`.
2. Keep your existing database. Auto-SQL remains backwards-compatible and creates any missing Operations tables without deleting existing inventory data.
3. Review `config/operations.lua` before restart. Existing defaults remain backwards-compatible.
4. If you customized `Config.UI.specialSlots`, merge your slot definitions into the v2.0.0 `config/ui.lua`. Category matching and explicit `items` allowlists remain supported.
5. Restart `nord_inventory` (or the server) and open **Admin Studio → Operations** and **Admin Studio → Integrations**.
6. Run one Anti-Dupe scan and verify the result before resolving historic alerts.
7. Create a manual snapshot of a test player, perform a small Inventory Manager edit and restore the snapshot to validate database permissions.
8. Place a disposable test item persistently, restart the resource and confirm the item returns at the same coordinates.
9. Confirm long weapons disappear from the back immediately when equipped/used and that ground drops stream in/out cleanly.
10. Confirm connected integration providers appear in the Integrations registry and that **Recommended scripts** / **Refresh** actions remain responsive.

No destructive database migration is required for v2.0.0.

Optional physical clothing

`Config.Operations.ClothingBridge.Enabled` remains `false` after upgrade. Enable it only when the configured appearance resource exposes the matching client export. This keeps v2.0.0 backwards-compatible with existing clothing/appearance setups.

Config merge checklist

Do not copy an old `config/` folder over v2.0.0. Explicitly review at least `config/operations.lua`, `config/ui.lua`, `config/admin.lua`, `config/player_interactions.lua` and any owner-customized bridge/vehicle/item settings.

Post-upgrade verification

After the resource starts, check for missing tables, provider failures and locale errors before allowing players in. Then perform the v2.0.0 Operations smoke tests from the Production Checklist.

Performance & Production Tuning

Performance & Production Tuning

Practical tuning without weakening authority.

Applies to: `Nord Inventory v2.0.0`

- Keep `SaveIntervalSeconds` reasonable; the current default is 30 seconds.
- Avoid polling full inventory snapshots every frame in external resources.
- Cache static item definitions/catalogs where appropriate and refresh on provider changes.
- Use client read-only exports for display-only checks; use server exports for mutations.
- Keep ground prop draw/spawn distances bounded.
- Avoid oversized item images; ecosystem image registration has a 2 MiB default cap.
- Keep transaction/audit retention appropriate for server volume.
- Do not disable trunk session guards to solve a UI symptom; fix the underlying lock/network/distance integration.

Operations & Forensics Center — v2.0.0

Operations & Forensics Center — v2.0.0

Nord Inventory v2.0.0 includes an admin-only Operations Center built around the existing server-authoritative inventory model.

Included systems

- **Ledger / Forensics** — searches `nord_inventory_transactions` by transaction id, identifier, item, inventory, UID, weapon serial or container UID. Movement transactions now preserve identity fields in their payload.
- **Anti-Dupe Center** — on-demand bounded scan of persisted + live cached inventories. UID, weapon serial and container UID are checked independently. A unique identity with multiple copies in one stack is also flagged.
- **Item Instance Explorer** — locates item instances across player, stash, container, vehicle and ground inventories and exposes owner, slot, storage key and sanitized metadata.
- **Inventory Manager** — inspects both connected and offline player inventories. Connected players are read from the authoritative live `InventoryCore` state; offline players are read from `nord_inventory_storage`. Every admin edit creates a safety snapshot before mutation when enabled.
- **Snapshots & Recovery** — manual and automatic per-inventory snapshots with owner-level restore permission. A pre-restore snapshot is always created first.
- **Persistent Placed Items** — players can deliberately place an item in the world. The drop, inventory payload and coordinates survive resource/server restarts.
- **Item Policies** — per-item or per-category movement restrictions for giving, dropping, throwing, persistent placement and destinations such as trunk, glovebox, stash, container, player and ground.
- **Dependency Graph** — reads the Nord provider registry and shows active provider/resource relationships in Admin Studio.

- **Equipment Slots 2.0** — special slots can now match item categories or explicit item-name allowlists. Existing servers retain the previous slot layout unless extra equipment slots are configured.

Permissions

The normal Nord Inventory permission bridge is used:

- `forensics` — Operations Center, ledger, instance explorer, anti-dupe and alert resolution.
- `offline_inventory` — Inventory Manager lookup, inspection, editing and manual snapshots. The permission name is kept for backwards compatibility.
- `restore_snapshots` — snapshot restore. Default minimum role is `admin`.
- `offline_inventory` — snapshot creation and deletion, alongside offline inventory management. Default minimum role is `admin`.

Nord Staff/ACE/database access continues to flow through the existing permission bridge.

Inventory Manager source of truth

The manager deliberately uses different sources depending on player state:

- **Online player:** `InventoryCore.GetPlayerInventory(source)` / live runtime state is authoritative. The UI never silently falls back to stale SQL for a connected owner.
- **Offline player:** `nord_inventory_storage` is authoritative.
- Online players are listed first and stale persisted rows for the same owner are de-duplicated.
- Legacy persisted player keys are redirected to the current canonical live inventory key when that owner is connected.
- Manual and automatic snapshots of connected players serialize the current live item table.
- Snapshot restore refreshes the connected player's inventory session immediately.
- Admin edits persist immediately; if the SQL transaction fails, the runtime mutation is rolled back in memory.
- While the Inventory Manager tab is open, selected live inventories receive event-driven updates through the normal inventory watcher path. The subscription is removed when the tab/Studio closes and also expires automatically.

Auto-SQL

The runtime creates these tables with `CREATE TABLE IF NOT EXISTS`:

- `nord_inventory_snapshots`
- `nord_inventory_forensics_alerts`
- `nord_inventory_placed_items`

The same schema is included in `sql/nord_inventory.sql` for manual installations. Existing inventory data is not deleted or rewritten by the Operations migration included with v2.0.0.

Item policies

Edit `config/operations.lua`. Missing properties mean **allow**, so updating from an older build does not block existing item movement.

```
Config.Operations.ItemPolicies = {
    ByName = {
        police_evidence = {
            allowGive = false,
            allowDrop = false,
            allowThrow = false,
            allowTrunk = false,
            allowGlovebox = false,
            allowedJobs = { police = 0 }
        }
    },
    ByCategory = {
        identity = {
            allowDrop = false,
            allowThrow = false
        }
    }
}
```

Supported policy fields: `allowGive`, `allowDrop`, `allowThrow`, `allowPersistentPlace`, `allowTrunk`, `allowGlovebox`, `allowStash`, `allowContainer`, `allowPlayer`, `allowGround`, `allowedJobs`.

Persistent placement

`Config.Operations.PersistentPlacedItems` controls the feature. Persistent items are intentionally different from ordinary drops:

- ordinary drops keep the configured lifetime;
- persistent placed items are database-backed;
- both sides are saved immediately when a persistent item is transferred to reduce crash-window duplication/loss;
- empty persistent placed inventories are removed from both the placed-item registry and inventory storage.

Scan safety

Forensics scans are on-demand and bounded by:

- `MaxLedgerRows`
- `MaxInstanceRows`
- `MaxStorageScanRows`
- `MaxOfflinePlayers`

The scanner overlays live cached inventories on top of database rows. That prevents stale database state and live state from being counted twice.

Equipment Slots 2.0

Existing category rules remain valid. An equipment slot may additionally use `items = { ... }`:

```
{
    key = 'phone',
    slot = 8,
    items = { 'phone', 'nord_phone' },
    label = 'Phone',
    icon = 'fa-mobile-screen'
}
```

Extra equipment slots are opt-in in `config/ui.lua` so an update does not unexpectedly reserve more player slots.

Optional clothing bridge

`Config.Operations.ClothingBridge` provides an opt-in hand-off for physical clothing items. It is disabled by default so existing servers keep their current appearance flow.

When enabled, an item whose category matches `Category` is treated as usable through the configured client export, for example:

```
ClothingBridge = {
    Enabled = true,
    Category = 'clothing',
    Resource = 'nord_appearance',
    ClientExport = 'UseInventoryClothingItem'
}
```

The target export receives `(itemData, slot)` and should return `false` to reject the equip action. Item-specific `use.mode = 'export'` definitions always take priority over this generic bridge.

Exports

Server exports added by the Operations module:

```
exports.nord_inventory:GetItemPolicy(itemName)
exports.nord_inventory:CreateInventorySnapshot(invKey, reason)
exports.nord_inventory:GetInventoryLedger(filters)
exports.nord_inventory:FindItemInstances(filters)
```

Production notes

The anti-dupe scan is diagnostic: it records alerts but never deletes items automatically. Snapshot restore requires an elevated permission and creates a safety snapshot first. Offline admin edits are audited through the existing Nord ecosystem audit system.

Transaction Ledger & Anti-Dupe

Transaction Ledger & Anti-Dupe

Applies to: `Nord Inventory v2.0.0`

Ledger

The Operations Ledger searches `nord_inventory_transactions`. Filters can target transaction ID, identifier, item, inventory, UID, weapon serial and container UID. Identity fields are preserved in transaction payloads to make investigations possible without mutating item state.

`Config.Operations.MaxLedgerRows` bounds the result size.

Clear records

Admins with the `forensics` permission can clear the transaction ledger through the Operations toolbar. The action:

- requires confirmation in NUI;
- deletes transaction rows only;
- does **not** reset the table primary key;
- does not touch live/persisted inventories;
- writes an entry to the separate Admin Studio audit log containing the deleted-row count.

New movements continue to be recorded immediately after cleanup.

Anti-Dupe Center

The scan is diagnostic and never deletes items automatically. It checks these identities independently:

- unique item UID;
- weapon serial;
- container UID.

A supposedly unique identity appearing in a stack with `count > 1` is also suspicious. The scanner overlays live cached inventories over persisted storage so the same online player is not counted twice from stale SQL + live runtime state.

Alerts are persisted in `nord_inventory_forensics_alerts` until resolved. Resolving an alert changes its status; it does not modify the underlying item automatically.

Safe use

Run scans on demand rather than polling them continuously. Review the matching inventories and item identity before deciding whether an alert represents a real duplication, a legacy malformed item or a legitimate historic record.

Inventory Manager & Snapshots

Inventory Manager & Snapshots

Applies to: `Nord Inventory v2.0.0`

Source of truth

The v2.0.0 manager has one strict rule:

- **online player:** authoritative `InventoryCore` runtime state;
- **offline player:** `nord_inventory_storage`.

A connected player is never silently edited from stale SQL. Legacy persisted keys for a connected owner are redirected to that owner's current live inventory key. Online players are listed first and duplicate stale storage rows are hidden.

Live updates

While an inventory is selected in Admin Studio, normal inventory watcher updates refresh the manager without destroying the visible snapshot list. Subscriptions are removed when the tab/Studio closes and automatically expire after `Config.Operations.LiveManagerWatchSeconds`.

Admin edits

Add/remove/clear operations create a safety snapshot first when `AutoSnapshotBeforeAdminEdit = true`. If persistence fails, the live mutation is rolled back in memory instead of leaving runtime and SQL inconsistent.

Snapshots

Snapshots store inventory key/type/owner, slot and weight limits, the item payload, reason, creator and timestamp. Connected-player snapshots serialize current live state.

A restore:

1. requires `restore_snapshots`;
2. creates a pre-restore safety snapshot;
3. restores the selected payload;
4. targets the canonical live key when the owner is online;
5. refreshes that player's inventory session immediately.

Retention is controlled with `Config.Operations.SnapshotRetentionDays`.

Persistent Placed Items & Movement Policies

Persistent Placed Items & Movement Policies

Applies to: `Nord Inventory v2.0.0`

Persistent placed items

Persistent placement is deliberately separate from ordinary ground drops. Ordinary drops still use the configured lifetime; persistent placements are backed by `nord_inventory_placed_items` plus normal inventory storage and survive resource/server restarts.

```
Config.Operations.PersistentPlacedItems = {
    Enabled = true,
    MaxPerPlayer = 30,
    MaxDistance = 8.0,
    DefaultModel = 'prop_cs_cardbox_01'
}
```

When an item is placed persistently, both inventory sides are persisted together through an atomic transaction where applicable. Empty persistent placed inventories are removed from the registry and storage. Arbitrary client-provided models are not trusted.

Item movement policies

Policies live in `config/operations.lua`. Missing fields mean **allow**.

```
Config.Operations.ItemPolicies = {
    ByName = {
        police_evidence = {
            allowGive = false,
            allowDrop = false,
            allowThrow = false,
            allowTrunk = false,
            allowGlovebox = false,
            allowedJobs = { police = 0 }
        }
    },
    ByCategory = {
        identity = {
            allowDrop = false,
            allowThrow = false
        }
    }
}
```

Supported fields: `allowGive`, `allowDrop`, `allowThrow`, `allowPersistentPlace`, `allowTrunk`, `allowGlovebox`, `allowStash`, `allowContainer`, `allowPlayer`, `allowGround`, `allowedJobs`.

Use item-specific rules for exceptional items and category rules for broad groups. Item rules and server-side validation are authoritative; disabling a NUI action alone is not a security boundary.

Operations Configuration & Limits

Operations Configuration & Limits

Applies to: `Nord Inventory v2.0.0`

`config/operations.lua` is shared configuration and is included in `escrow_ignore` for server owners.

```
Config.Operations = {
    Enabled = true,
    MaxLedgerRows = 250,
    MaxInstanceRows = 1200,
    MaxStorageScanRows = 5000,
    MaxOfflinePlayers = 80,
    SnapshotRetentionDays = 30,
    AutoSnapshotBeforeAdminEdit = true,
    LiveManagerWatchSeconds = 300,
    PersistentPlacedItems = {
        Enabled = true,
        MaxPerPlayer = 30,
        MaxDistance = 8.0,
        DefaultModel = 'prop_cs_cardbox_01'
    },
    ClothingBridge = {
        Enabled = false,
        Category = 'clothing',
        Resource = 'nord_appearance',
        ClientExport = 'UseInventoryClothingItem'
    },
```

```
    ItemPolicies = { ByName = {}, ByCategory = {} }
}
```

Why the limits exist

Forensics reads can touch many persisted inventories, so v2.0.0 bounds Ledger, instance, storage and offline-player result sets. Increase them only after measuring on your database. Do not turn Admin Studio scans into high-frequency polling.

Clothing bridge

The clothing bridge is disabled by default. When enabled, items in the configured category are handed to the specified **client export** as `(itemData, slot)`. Returning `false` rejects the equip action. Explicit item `use.mode = 'export'` behavior has priority over the generic clothing bridge.