Setup & Configuration

- Overview · Setup & Configuration
- Requirements & Installation
- Production Resource Structure
- Modular Configuration
- Core Configuration
- Inventory Runtime Configuration
- Framework Detection & Custom Bridge
- Admin Access & Roles
- Admin Commands
- Notify, Text UI & Target Bridges
- Locales & Translation
- Static Stashes
- Tebex / Asset Escrow Release
- Asset Escrow Map
- Production Checklist

Overview · Setup & Configuration

1 · Setup & Configuration

Install Nord Inventory, configure frameworks, permissions, locales, bridges and production packaging.

Requirements & Installation

Requirements & Installation

Production installation, dependency order and first-run checks.

Applies to: `Nord Inventory v2.0.0`

Requirements

- FiveM/FXServer using a modern artifact compatible with `cerulean`.
- `oxmysql` running before Nord Inventory.
- A supported framework, or `standalone/custom` mode.
- MySQL/MariaDB database access.

Recommended start order

```
ensure oxmysql
# framework resources here
ensure nord_inventory
# resources which register into Nord Inventory afterwards
```

If a third-party resource registers items at startup, start it **after** `nord_inventory` unless it implements its own retry/readiness logic.

Resource naming

Do not rename `nord_inventory` unless you are also prepared to change every external integration that calls `exports.nord_inventory`.

Database

Install `sql/nord_inventory.sql`. Keep database backups before migration. Never let two inventory resources simultaneously own the same player inventory state during a live migration.

First boot checklist

- no SQL migration/query errors;
- framework detected as intended;
- locale JSON loads;
- inventory opens and closes correctly;
- Admin Studio permission works;
- one test item survives reconnect;
- one stash and one vehicle session can be opened;
- System Health is reviewed before production traffic.

Production Resource Structure

Production Resource Structure

The important folders and what server owners should edit.

Applies to: `Nord Inventory v2.0.0`

Production structure

```
nord_inventory/
├── config/                         # all owner configuration
│   ├── core.lua
│   ├── commands.lua
│   ├── inventory.lua
│   ├── bridges.lua
│   ├── vehicles.lua
│   ├── world.lua
│   ├── items.lua
│   ├── ui.lua
│   ├── stashes.lua
│   ├── locales.lua
│   └── admin.lua                  # server-only access settings
├── client/
│   ├── bridge/                    # OPEN compatibility layer
│   ├── core/                      # protected runtime/UI interaction core
│   └── modules/                   # protected public integration implementation
```

```
├── server/
│   ├── bridge/                    # OPEN compatibility layer
│   │   ├── framework/
│   │   │   ├── core.lua
│   │   │   ├── standalone.lua
│   │   │   ├── qb.lua
│   │   │   ├── qbox.lua
│   │   │   ├── esx.lua
│   │   │   ├── nord.lua
│   │   │   └── custom.lua
│   │   ├── permissions.lua
│   │   ├── notify.lua
│   │   └── textui.lua
│   ├── core/                      # protected database/registry/inventory engine
│   └── modules/                   # protected admin/runtime/integration modules
├── shared/                        # definitions + protected shared helpers
├── locales/                       # owner-editable EN / PT-PT JSON
├── sql/                           # installation/reference schema
├── web/                           # NUI + item artwork
└── docs/                          # server/developer documentation
```

Design rules

- Owner configuration belongs only in `config/`.
- Framework-specific server code belongs only in `server/bridge/framework/`.
- Notify/TextUI/Target compatibility belongs in bridge files, not in the inventory engine.
- Database, transaction, registry and provider ownership rules belong in protected core modules.
- `config/admin.lua` is server-only and must never move into `shared_scripts`.
- Lua files are loaded explicitly in the manifest because configuration and bridge order is intentional.
- The Nord Lab Recommended Scripts catalog remains in protected `server/core/ecosystem.lua`.

Modular Configuration

Modular Configuration

All supported owner configuration is split by responsibility under `config/`.

Applies to: `Nord Inventory v2.0.0`

File	Responsibility
`config/core.lua`	framework, debug, ecosystem visibility, inventory command/key
`config/inventory.lua`	capacities, sessions, ammo, durability/history
`config/operations.lua`	Operations/forensics limits, persistent placement, clothing bridge and movement policies
`config/ui.lua`	UI, Fast Slots, special slots, hotbar, item feedback
`config/admin.lua`	server-only ACE/license/Nord Staff admin access
`config/commands.lua`	Admin Studio and administration command aliases
`config/bridges.lua`	notify, TextUI, target and interaction mode
`config/locales.lua`	active language and fallback
`config/player_interactions.lua`	search/rob/hands-up rules
`config/world.lua`	ground item/drop/throw behavior
`config/vehicles.lua`	registry, trunks, gloveboxes, locks, NPC loot
`config/items.lua`	prop rules, back weapons, use animations, aliases
`config/stashes.lua`	static stashes
`config/rarities.lua`	rarity levels and UI behavior

Rule for upgrades

Merge your intentional values into the new files. Do **not** blindly copy an old `config/` directory over a newer release.

Core Configuration

Core Configuration

Current `config/core.lua` defaults.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - core configuration
-- Loaded on both client and server. Keep secrets out of this file.

Config = Config or {}

Config.Framework = 'auto' -- auto | esx | qb | qbox | nord | standalone | custom
Config.Debug = false

-- Developer ecosystem / Admin Studio extensions.
Config.Ecosystem = {
    Debug = false, -- verbose provider/export resolution logs
    MaxImageBytes = 2 * 1024 * 1024,

    -- Nord Lab curated recommendations are read-only. Server owners can only
    -- choose whether the Recommended Scripts button/catalog is visible in Admin Studio.
    -- The catalog itself remains inside the protected Asset Escrow server core.
    ShowRecommendedScripts = true
}

Config.OpenCommand = 'inventory'
Config.OpenKey = 'TAB'
```

Inventory Runtime Configuration

Inventory Runtime Configuration

Slot/weight limits, session behavior and weapon magazine defaults.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - inventory limits, weapon ammo and runtime rules

Config = Config or {}

Config.DefaultPlayerSlots = 40
Config.DefaultPlayerWeight = 80000 -- grams
Config.DefaultStashSlots = 60
Config.DefaultStashWeight = 150000
Config.TrunkSlots = 50
Config.TrunkWeight = 120000
Config.GloveboxSlots = 10
Config.GloveboxWeight = 15000

-- Weapon magazine handling. Nord keeps ONLY the currently loaded magazine on the
-- weapon metadata; reserve ammunition always remains as inventory items until a
-- reload transaction removes exactly the rounds that fit in the magazine.
Config.WeaponAmmo = {
    Enabled = true,
    DefaultMagazineSize = 30, -- used only until a weapon instance has magazineSize metadata
    MaxMagazineSize = 999,
    DetectMagazineFromGame = true, -- first equip can seed metadata.magazineSize from GTA's
```

```
native clip size

    -- Keep the weapon physically equipped when the magazine reaches 0.
    -- GTA may try to switch the ped to WEAPON_UNARMED when the last round is
    -- fired; Nord immediately restores the empty weapon. The weapon is only
    -- holstered when the player uses/clicks that same inventory item again.
    KeepEquippedWhenEmpty = true,

    -- Automatic mode is intentionally slower than pressing R manually.
    -- First Nord waits before requesting the reload, then it keeps the weapon
    -- empty for an additional preparation period before the native reload starts.
    AutoReloadDelay = 1400, -- ms after the magazine reaches zero before automatic reload is
requested
    AutomaticReloadPrepareTime = 900, -- extra ms before the automatic reload animation /
rounds enter the weapon

    ReloadCooldown = 650, -- protects against duplicate reload requests
    ManualReloadControl = 45, -- INPUT_RELOAD (R)
}

Config.InventorySessionSeconds = 300
Config.SaveIntervalSeconds = 30
Config.MaxTransferAmount = 100000
Config.MaxContainerDepth = 3

Config.EnableWeight = true
Config.EnableVolume = false
Config.EnableDurability = true
Config.EnableItemHistory = true
Config.TransactionLogRetentionDays = 14
```

Framework Detection & Custom Bridge

Framework Detection & Custom Bridge

Framework selection and how to handle unsupported/custom cores.

Applies to: `Nord Inventory v2.0.0`

```
Config.Framework = 'auto'
```

`auto` lets the bridge inspect known resources. Explicit values are preferable on heavily customized servers because startup intent is unambiguous.

Supported values:

- `esx`
- `qb`
- `qbox`
- `nord`
- `standalone`
- `custom`

Custom integrations

Keep custom framework logic in bridge code instead of modifying inventory authority. A custom bridge should resolve player identity, jobs/groups where required, and any framework-owned money/account behavior while leaving item state to Nord Inventory.

Troubleshooting auto detection

If more than one framework-like resource is present, set an explicit framework value. Restart `nord_inventory` after changing framework mode so the runtime initializes against one consistent provider.

Admin Access & Roles

Admin Access & Roles

ACE, server-only license fallback and Nord Staff grants.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - SERVER ONLY admin access configuration
-- This file is loaded only through server_scripts. FiveM identifiers in this file are never sent to clients.

Config = Config or {}

Config.AdminAce = 'nord_inventory.admin'

Config.AdminAccess = {
    -- ACE is checked first. The default matches server.cfg.example:
    -- add_ace group.admin nord_inventory.admin allow
    UseAce = true,
    Ace = Config.AdminAce or 'nord_inventory.admin',

    -- If ACE does not grant access, allow an exact FiveM identifier from this list.
    -- Recommended format: license:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    -- You may also paste only the hash; Nord will automatically prefix "license:".
    LicenseFallback = true,
    Licenses = {
        -- 'license:0123456789abcdef0123456789abcdef01234567',
    },

    -- Automatic Nord Staff access integration. nord_staff detects the public
    -- HasAdminAccess/GiveAdminAccess/RemoveAdminAccess exports and can grant or
    -- revoke this role without changing ACE or the static license lists above.
    NordStaffBridge = {
        Enabled = true,
```

```
        Role = 'admin', -- developer | admin | owner
        AllowedResources = { 'nord_staff' }
    }
}

-- Optional Admin Studio role separation.
-- Legacy `nord_inventory.admin` keeps full day-to-day administration access.
Config.AdminRoles = {
    UseAce = Config.AdminAccess.UseAce ~= false,
    OwnerAce = 'nord_inventory.owner',
    AdminAce = Config.AdminAccess.Ace or 'nord_inventory.admin',
    DeveloperAce = 'nord_inventory.dev',

    -- License fallback entries may stay as strings (treated as admin) or use:
    -- { identifier = 'license:...', role = 'owner' }
    LicenseRoles = {
        -- { identifier = 'license:012345...', role = 'owner' },
    }
}
```

Role order

`developer < admin < owner`

Owner should be restricted because restore operations can replace persistent inventory state.

Admin Commands

Admin Commands

Current command aliases and access expectations.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - commands configuration
-- Command names are shared because the inventory/Admin Studio commands are registered client-side too.

Config = Config or {}

Config.AdminCommand = 'norditems'

-- All commands below require Admin access (ACE first, then the server-only license fallback in config/admin.lua).
-- Rename/remove aliases here if they conflict with another resource.
Config.AdminCommands = {
    Enabled = true,
    GiveItem = { 'giveitem', 'nordgiveitem' },
    GiveCash = { 'givecash', 'nordgivecash' },
    GiveDirtyMoney = { 'givedirtymoney', 'nordgivedirtymoney' },
    RemoveItem = { 'removeitem', 'nordremoveitem' },
    RemoveSlot = { 'removeslot', 'nordremoveslot' },
    ClearInventory = { 'clearinventory', 'clearinv' },
    OpenInventory = { 'openinventory', 'openinv' },
    SetMetadata = { 'setmetadata', 'setmeta' },
    SetDurability = { 'setdurability', 'repairitem' },
    ItemCount = { 'itemcount' },
    ItemInfo = { 'iteminfo' },
    SaveInventory = { 'saveinventory', 'saveinv' },
    ListItems = { 'listitems' }
}
```

All configured administration commands are server-authorized. Renaming the command in NUI or client code does not grant access.

Notify, Text UI & Target Bridges

Notify, Text UI & Target Bridges

Independent bridge selection for notifications, world prompts and target interactions.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - bridge and interaction providers

Config = Config or {}

-- Bridges: one resource name per system.
-- Keep each integration independent so changing Notify never affects TextUI/Target.
Config.Notify = 'nord_inventory' -- nord_inventory | ox_lib | qb-core | es_extended | resource name | false
Config.TextUI = 'nord_inventory' -- nord_inventory | ox_lib | qb-core | es_extended | resource name | false
Config.Target = 'auto' -- auto | ox_target | qb-target | nord_target | resource name | false

-- World interaction mode only. Provider selection lives above in Config.TextUI / Config.Target.
Config.Interaction = {
    Mode = 'textui', -- textui | target
    Key = 'E',
    Control = 38 -- INPUT_CONTEXT (E)
}
```

`Config.Target = 'auto'` checks supported target resources and falls back according to the configured interaction mode. Notify, TextUI and Target are deliberately independent so swapping

one provider does not force the others to change.

Locales & Translation

Locales & Translation

PT/EN locale architecture and the rule for adding visible UI strings.

Applies to: `Nord Inventory v2.0.0`

The build ships with:

```
locales/en.json
locales/pt.json
```

Selection is controlled in `config/locales.lua`:

```
Config.Locale = 'pt'
Config.LocaleFallback = 'en'
```

Locale data is grouped into `notifications`, `nui`, `patterns`, and `server` namespaces. The Lua `L()` helper resolves active language first and fallback second. NUI waits for locale data before rendering localized text.

Adding a new visible string

1. Add the English key/value to `locales/en.json`.
2. Add the matching Portuguese key/value to `locales/pt.json`.
3. Use the existing translation helper in Lua/NUI instead of hard-coding a visible label.
4. Verify both locales after changing dynamic components such as modals/tabs/search states.

This is especially important for the Recommended Scripts modal: tabs are derived from `integration` values, but visible compatibility labels and UI actions remain locale-aware.

Static Stashes

Static Stashes

Static configuration plus dynamic registration options.

Applies to: `Nord Inventory v2.0.0`

```
-- Nord Inventory - static stash definitions

Config = Config or {}

-- Stashes use Config.Interaction (Nord Text UI / ox_lib / target).
-- Dynamic stashes can also be registered from other server resources through exports.
Config.Stashes = {
    -- ['police_evidence'] = {
    --     label = 'Police Evidence',
    --     coords = vector3(474.6, -996.8, 26.27),
    --     slots = 100,
    --     maxWeight = 300000,
    --     jobs = { police = 0 }
    -- }
}
```

Dynamic stashes should normally use `RegisterStash` or `RegisterInventory` from a trusted server resource. Avoid creating storage by writing directly to Nord database tables.

Tebex / Asset Escrow Release

Tebex / Asset Escrow Release

Production packaging and distribution notes.

Applies to: `Nord Inventory v2.0.0`

Tebex / Asset Escrow release

This package is structured as the production distribution of Nord Inventory.

Resource name

Keep the installed resource folder named exactly:

```
nord_inventory
```

Start `oxmysql` before the inventory:

```
ensure oxmysql
ensure nord_inventory
```

What stays open

Nord Inventory intentionally keeps only the parts a server owner or integration developer may reasonably need to edit outside Asset Escrow.

Owner configuration

Everything under `config/` stays open:

```
config/
├── core.lua
├── commands.lua
├── inventory.lua
├── bridges.lua
├── vehicles.lua
├── world.lua
├── items.lua
├── ui.lua
├── stashes.lua
├── locales.lua
└── admin.lua
```

There is no root `config.lua` and no `server/config.lua`.

Compatibility bridges

Framework/provider compatibility stays open so a server can adapt Nord Inventory without modifying its protected core:

```
server/bridge/
├── framework/
│   ├── core.lua
│   ├── standalone.lua
│   ├── qb.lua
│   ├── qbox.lua
│   ├── esx.lua
│   ├── nord.lua
│   └── custom.lua
├── permissions.lua
├── notify.lua
└── textui.lua
```

The client bridge layer is open as well:

```
client/bridge/
├── common.lua
├── notify.lua
├── target.lua
└── textui.lua
```

Use `server/bridge/framework/custom.lua` for unsupported frameworks instead of changing the inventory core.

Data intended for owner edits

These files also remain open:

- `shared/items.lua`
- `shared/weapons.lua`
- `shared/vehicles.lua`
- `shared/categories.lua`
- `locales/en.json`
- `locales/pt.json`
- item images under `web/images/items/`

What stays protected

The inventory engine, database/registry implementation, Admin Studio server actions, provider ecosystem implementation, runtime transaction logic and the main NUI source remain protected by Asset Escrow.

In particular, do **not** add the following to `escrow_ignore`:

```
server/core/ecosystem.lua
```

That file contains the Nord Lab curated Recommended Scripts catalog. Server owners can choose whether the read-only button is visible through `Config.Ecosystem.ShowRecommendedScripts`, but cannot edit the catalog itself.

Framework bridge layout

Set the framework in `config/core.lua`:

```
Config.Framework = 'auto'
```

Supported values:

```
auto
qb
qbox
esx
nord
standalone
custom
```

`auto` detects Qbox, QB-Core, ESX and Nord Core in that order. If an explicitly configured framework is unavailable, Nord falls back safely to standalone identifiers and logs the reason.

Client download footprint

Documentation, changelogs and SQL files are included in the Tebex ZIP for server owners, but are deliberately **not** listed in the manifest `files` block. Connecting players only download the runtime NUI/locales/item artwork required by the resource.

Encryption note

A source ZIP is not itself Cfx.re-encrypted. FiveM Asset Escrow protection is applied when Nord Lab publishes the resource through the Cfx.re/Tebex Asset Escrow pipeline.

Before publishing, verify that:

1. `fxmanifest.lua` contains the intended `escrow_ignore` list.
2. `server/core/ecosystem.lua` is not ignored.
3. No protected `server/core/`, `server/modules/`, `client/core/` or `client/modules/` file was accidentally added to the ignore list.
4. The resource folder is named `nord_inventory`.
5. The production checklist passes on a staging FXServer.

Asset Escrow Map

Asset Escrow Map

Which surfaces are intended to stay configurable and which remain protected.

Applies to: `Nord Inventory v2.0.0`

Asset Escrow map

Nord Inventory uses a deliberately small open surface for server configuration and compatibility work.

Open in production

```
config/**
server/bridge/**
client/bridge/**
shared/items.lua
shared/weapons.lua
shared/vehicles.lua
shared/categories.lua
locales/en.json
locales/pt.json
web/images/items/*
```

These files are listed in `escrow_ignore` because server owners need to configure providers, frameworks, permissions, reference items, translations and artwork.

Protected in production

```
server/core/**
server/modules/**
client/core/**
client/modules/**
shared/utils.lua
web/index.html
web/style.css
web/app.js
```

The protected area contains Nord Inventory business logic, transaction rules, database registry behavior, Admin Studio backend, integration ownership implementation and the curated Nord Lab recommendation catalog.

Recommended Scripts security

`server/core/ecosystem.lua` must remain protected. The only server-owner control is:

```
Config.Ecosystem.ShowRecommendedScripts = true
```

Setting it to `false` hides the Recommended Scripts button. It does not expose or alter the Nord Lab catalog.

Production Checklist

Production Checklist

Pre-release checklist for server owners and Nord Lab builds.

Applies to: `Nord Inventory v2.0.0`

Production Checklist

Use this checklist before replacing an inventory on a live server.

Tebex / Asset Escrow package

- Confirm the ZIP contains a single resource folder named `nord_inventory`.
- Confirm all owner settings are under `config/`; there must be no root `config.lua` or `server/config.lua`.
- Confirm `server/bridge/` and `client/bridge/` are listed in `escrow_ignore` as intended.
- Confirm `server/core/ecosystem.lua` is **not** in `escrow_ignore`.
- Confirm protected `server/core/`, `server/modules/`, `client/core/` and `client/modules/` files were not accidentally exposed.
- Confirm docs/SQL/changelog are not listed in the manifest `files` block, so clients do not download release documentation.
- Publish through Cfx.re/Tebex Asset Escrow before calling the distributed build encrypted/protected.

Files and dependencies

- Keep a copy of the currently deployed inventory resource and database.
- Start `oxmysql` before `nord_inventory`.
- Start any target/notification/TextUI providers in the order required by your configuration.
- Start `nord_inventory` before resources that call its registration exports.

- Confirm `fxmanifest.lua` and all listed files are present.

Database

- Let Nord create its tables automatically on a staging database, or import `sql/nord_inventory.sql` manually.
- Confirm `nord_inventory_custom_items`, `nord_inventory_disabled_reference_items`, audit and backup tables exist.
- Create an Admin Studio backup before a migration/import.
- Keep a full database backup outside Nord before the first production deployment.

Permissions

- Verify Developer/Admin/Owner ACE roles with test accounts.
- Confirm Developer cannot save/delete/import/restore through direct NUI/server event attempts.
- Confirm backup restore is restricted to Owner unless you intentionally change the permission bridge.
- If `Config.AdminAccess.UseAce = false`, verify only configured license fallback entries receive access.

Item registry

- Open System Health and resolve errors before launch.
- Review missing images and invalid/offline exports.
- Verify reference-item disable/restore behavior.
- Verify a DB-authoritative definition cannot be overwritten by `RegisterItem`.
- Verify runtime provider items disappear when their provider stops and re-register when it starts.

External integrations

For each native integration:

- Confirm the provider appears on **Integrations**.
- Confirm dependencies are online.

- Confirm item/provider ownership is what you expect.
- Test client and server external use handlers separately.
- Test return `false` and stopped-resource failure paths.
- Test item actions and metadata schema extensions.
- Stop/restart the provider and confirm cleanup/re-registration.

Player inventory

- Add/remove/transfer/split/give/drop items.
- Open Ground, stash, trunk, glovebox and item containers.
- Test overweight and slot-limit rejection.
- Test reconnect persistence.
- Test two players interacting with the same secondary inventory.

Lifecycle

- Test normal durability loss.
- Test durability repair items with missing, full and damaged targets.
- Test `GetDurability`, `SetDurability`, `AddDurability`, `RemoveDurability` and `RepairItem` from a trusted test resource.
- Test expiration on online and offline-loaded items.

Weapons

- Equip/unequip weapons.
- Verify serial/durability/ammo metadata.
- Verify manual and automatic magazine reload modes.
- Verify reserve ammunition remains inventory-authoritative.

Vehicles

- Test trunk and glovebox capacities by model/category.
- Test addon front/rear trunk placement.
- Test vehicles with no trunk.
- Test locked/unlocked/NPC vehicle behavior according to config.

Admin Studio

- Create/edit/delete/restore a database and reference item.
- Run an import preview before committing.
- Test templates.
- Create a backup, make a harmless change, restore it, then verify registry reload.
- Confirm audit records the administrative changes.

Static validation included with this package

The packaged build should pass:

- Lua compile/syntax validation for all `.lua` files.
- JavaScript syntax validation for `web/app.js`.
- JSON parsing and EN/PT locale-key parity.
- duplicate HTML-ID check.
- NUI callback and Nord client/server event mapping checks.

Static validation cannot reproduce a real FXServer, framework, OneSync, database driver, target provider or third-party resource. Run the checklist on a staging FXServer before swapping a live production server.

v2.0.0 Operations smoke test

- Confirm `nord_inventory_snapshots`, `nord_inventory_forensics_alerts` and `nord_inventory_placed_items` exist.
- Open all Operations tabs and confirm Ledger/Instance filters remain usable on your Admin Studio resolution.
- Run Anti-Dupe once; verify it records diagnostics but does not delete items.
- Create and delete a manual snapshot.
- Restore a disposable test snapshot and verify a pre-restore snapshot is created.
- Edit an online player's inventory and confirm the live client refreshes immediately.
- Test a simulated persistence failure on staging if possible; runtime state should roll back rather than diverge from SQL.
- Place a disposable persistent world item, restart the resource/server and verify it returns.
- Configure one test movement policy and verify server-side rejection for the blocked destination/action.

- Verify Recommended Scripts shows Nord Lab/Renovax Store and Documentation buttons and that only `http/https` URLs open.