Reference Files

- Overview · Reference Files
- Configuration Reference
- Commands & Default Keybinds
- SQL Table Map
- Production File Map
- Recommended Script Entry Schema
- Public API vs Internal Events
- Downloadable Reference Files

Overview · Reference Files

8 · Reference Files

Quick-reference configuration, schema, file map and import attachments.

Configuration Reference

Configuration Reference

Current owner-editable configuration files and defaults.

Applies to: `Nord Inventory v2.0.0`

Configuration

Nord Inventory keeps all owner-editable settings inside one `config/` directory. The former root `config.lua` and `server/config.lua` files have been removed.

Files

File	Purpose	Loaded on
`config/core.lua`	Framework, locale, debug, ecosystem controls, inventory open command/key	Client + Server
`config/commands.lua`	Admin Studio command and protected command aliases	Client + Server
`config/inventory.lua`	Player/stash/vehicle base limits, weapon ammo, sessions, durability/history	Client + Server
`config/operations.lua`	Operations/forensics, snapshots, persistent placement, policies, clothing bridge	Client + Server
`config/player_interactions.lua`	Player search/robbery/hands-up rules	Client + Server
`config/bridges.lua`	Notify, TextUI, Target and world interaction provider selection	Client + Server
`config/vehicles.lua`	Vehicle registry, trunk/glovebox rules, NPC robbery/loot and placement	Client + Server

File	Purpose	Loaded on
`config/world.lua`	Ground/drop distances, capacities and world animations	Client + Server
`config/items.lua`	Hand props, fallback props, back weapons and item-use animations	Client + Server
`config/rarities.lua`	Rarity levels, colors and presentation	Client + Server
`config/ui.lua`	Inventory UI, hotbar and received-item slot feedback	Client + Server
`config/stashes.lua`	Static stash definitions	Client + Server
`config/locales.lua`	Locale fallback/path and translation helpers	Client + Server
`config/admin.lua`	ACE names, direct license fallback and Admin Studio roles	**Server only**

Load order

The order in `fxmanifest.lua` is intentional. `config/inventory.lua` loads before `config/vehicles.lua` because the vehicle registry reuses the base trunk/glovebox limits. `config/core.lua` loads first and initializes `Config`.

Do not move `config/admin.lua` into `shared_scripts`. It may contain FiveM license identifiers and is intentionally loaded only in `server_scripts` before `server/bridge/permissions.lua`.

Recommended Scripts visibility

Nord Lab owns the contents of the Recommended Scripts catalog in protected server source. Server owners can only control whether the read-only button is visible:

```
-- config/core.lua
Config.Ecosystem = {
    Debug = false,
    MaxImageBytes = 2 * 1024 * 1024,
    ShowRecommendedScripts = true
}
```

Set `ShowRecommendedScripts = false` to hide the button and prevent the catalog payload from being sent to the Admin Studio NUI. This does not expose or allow editing the Nord Lab catalog.

Admin access

Edit `config/admin.lua` for ACE and optional license fallback:

```
Config.AdminAccess = {
    UseAce = true,
    Ace = 'nord_inventory.admin',
    LicenseFallback = true,
    Licenses = {
        -- 'license:0123456789abcdef0123456789abcdef01234567',
    }
}
```

The file remains owner-editable in Asset Escrow but is never shared with clients by the manifest.

Bridge selection

Edit `config/bridges.lua`:

```
Config.Notify = 'nord_inventory'
Config.TextUI = 'nord_inventory'
Config.Target = 'auto'
```

Changing one bridge does not change the others.

Framework bridge

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

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

Available values are `auto`, `qb`, `qbox`, `esx`, `nord`, `standalone`, and `custom`. The implementation is split under `server/bridge/framework/`; those files remain owner-editable in Asset Escrow.

For an unsupported framework, set:

```
Config.Framework = 'custom'
```

and implement the small adapter contract in `server/bridge/framework/custom.lua`. Do not modify the inventory engine for framework compatibility.

Received item feedback

Edit `config/ui.lua`:

```
Config.ItemNotifications = {
    Enabled = true,
    Receive = true,
    Duration = 1800,
    MaxVisible = 8
}
```

Source item definitions

Item, weapon, vehicle and category reference definitions remain in `shared/` because they are data registries rather than runtime settings. They are still owner-editable through `escrow_ignore`.

Complete PT / EN locale system

Language is selected in `config/locales.lua`:

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

Both `locales/en.json` and `locales/pt.json` contain:

- `notifications`: shared client/server notifications;
- `nui`: Admin Studio and player NUI text;
- `patterns`: dynamic UI phrases with counters/names;
- `server`: integration-specific server messages.

The loader checks both `notifications` and `server` for `L('key')`.

The NUI also performs a `localeReady` handshake after loading, so the selected locale is resent even if the browser missed the first early NUI message.

When adding a new visible UI string, add the English source string as a key in both `nui` tables. Keep EN/PT key parity.

Empty weapon equip behavior

By default, an equipped firearm stays in the player's hands when its magazine reaches zero.

```
Config.WeaponAmmo.KeepEquippedWhenEmpty = true
```

FiveM/GTA may automatically select `WEAPON_UNARMED` after the last shot. Nord Inventory immediately restores the same empty weapon and keeps its inventory equip state active.

The weapon is only holstered when the player **uses/clicks the same weapon item again**. This preserves the normal inventory equip/unequip toggle.

Set the option to `false` to restore the older automatic unarmed behavior.

Default weapon reload mode

The default reload mode for new/reset player preferences is **Manual**.

```
R -> request a reload
```

Automatic reload is still available in Personal Studio under the weapon reload setting. Existing players who already saved `automatic` keep that preference; this change only changes the default/fallback.

Automatic reload timing

Automatic reload is intentionally slower than manual reload.

```
Config.WeaponAmmo.AutoReloadDelay = 1400
Config.WeaponAmmo.AutomaticReloadPrepareTime = 900
```

The automatic flow is:

```
magazine reaches 0
→ wait 1.4 s
```

```
→ server approves/removes compatible ammunition
→ keep weapon empty for another 0.9 s
→ play the native weapon reload
→ rounds become available
```

Manual `R` reload and directly using an ammunition item do **not** use the extra automatic preparation delay.

Default Personal Studio layout

The default player inventory layout now matches the compact transparent preset used by the project reference layout:

```
Accent: cyan (#59c7ff)
Columns: 5
Panel opacity: 62%
Header opacity: 68%
Slot opacity: 54%
Border opacity: 7%
World shade: 12%
Panel gap: 22
Slot gap: 6
Item scale: 61%
```

These are the defaults for players without saved Personal Studio preferences and for reset/default states. Players with existing saved preferences keep their own layout until they reset/change it.

Metadata label override

Nord Inventory follows the OX-style per-instance label behavior.

When an item instance contains:

```
metadata = {
    label = 'teste'
}
```

the player sees `teste` as that item's name everywhere in the player inventory:

- inventory grid;
- Fast Slots / hotbar;
- Special Slots;
- drag preview;
- item information modal;
- context menu;
- receive/use/give item pulse;
- ground label for a single dropped item.

The base item definition label is preserved internally as `baseLabel`, while `name` remains the canonical internal/spawn identifier.

Example:

```
name       = nr_food
baseLabel  = Comida de Restaurante
metadata.label = teste


Visible player label = teste
```

If `metadata.label` is missing or empty, Nord falls back to the normal item definition label.

Hide metadata details from players

Each item definition can hide all metadata information cards from the player:

```
hideMetadataDetails = true
```

This never deletes or changes the metadata itself. Scripts and integrations keep receiving the complete metadata normally.

Presentation metadata is separate:

```
metadata = {
    label = 'Fresh Burger',
    image = 'fresh_burger.png',
    recipeId = 'secret_481',
    restaurant = 'cute_cafe'
}
```

With `hideMetadataDetails = true`, the player still sees `Fresh Burger` and `fresh_burger.png`, but `recipeId`, `restaurant` and the remaining metadata are not shown in the item information modal.

`metadata.label` and `metadata.image` are never rendered as metadata detail cards. The existing per-field Player Info switches still work whenever the master hide option is disabled.

Metadata privacy default

Metadata details are now **hidden by default for all items**.

No definition is required to hide metadata:

```
['my_item'] = {
    label = 'My Item'
}
```

To make metadata detail cards visible for a specific item:

```
['my_item'] = {
    label = 'My Item',
    hideMetadataDetails = false
}
```

`metadata.label` and `metadata.image` are presentation metadata and keep working in both modes.

Player capacity self-repair

Player inventories cannot use a zero slot/weight capacity. If an old integration or database row contains `slots <= 0` or `max_weight <= 0`, Nord automatically restores:

```
Config.DefaultPlayerSlots
Config.DefaultPlayerWeight
```

For maintenance/debugging:

```
local ok, result = exports['nord_inventory']:RepairInventoryCapacity(playerId)
```

The export also cleans invisible invalid/zero-count runtime rows and immediately saves the repaired persistent inventory.

Current correction: item feedback

For `{VERSION}`, `Config.ItemNotifications` uses `Duration` and `MaxVisible`; older `QueueGap/MaxQueue` examples are obsolete.

Operations / Forensics

New in v2.0.0, `config/operations.lua` controls the Operations Center limits and behavior. Missing movement-policy fields mean allow. `ClothingBridge.Enabled` is `false` by default for backwards compatibility. See **Database, Security & Operations → Operations Configuration & Limits** for the complete current structure.

Commands & Default Keybinds

Commands & Default Keybinds

Quick operational reference.

Applies to: `Nord Inventory v2.0.0`

Purpose	Default
Open inventory	`TAB` / `/inventory`
Quick hotbar	`Z` / `/hotbar`
Fast Slots	`1`–`5`
Hands up	`X` / `/handsup`
Search interaction	`E` (default interaction)
Rob interaction	`G`
Manual weapon reload	`R`
Admin Studio	`/norditems`

Admin command aliases are fully configurable in `config/commands.lua`.

SQL Table Map

SQL Table Map

Fast lookup of persistent table responsibilities.

Applies to: `Nord Inventory v2.0.0`

Table	Main purpose
`nord_inventory_storage`	persisted inventories/storage
`nord_inventory_custom_items`	DB-authoritative item definitions
`nord_inventory_item_versions`	definition version/history support
`nord_inventory_disabled_reference_items`	disabled built-in/reference items
`nord_inventory_custom_categories`	custom categories
`nord_inventory_preferences`	player preferences
`nord_inventory_transactions`	transaction/history records
`nord_inventory_vehicle_categories`	vehicle category capacity overrides
`nord_inventory_vehicle_models`	model-specific vehicle overrides
`nord_inventory_audit_log`	administrative audit trail
`nord_inventory_backups`	Nord backup metadata/data
`nord_inventory_admin_access`	persisted runtime admin grants

| `nord_inventory_snapshots` | Operations inventory snapshots and restore payloads | | `nord_inventory_forensics_alerts` | persisted Anti-Dupe/forensics alert history | | `nord_inventory_placed_items` | persistent world-placement registry/coordinates |

Production File Map

Production File Map

Where to look for configuration, integrations and public reference material.

Applies to: `Nord Inventory v2.0.0`

```
nord_inventory/
├── config/                owner configuration
├── locales/               EN/PT locale JSON
├── shared/                public/shared registries and definitions
├── client/bridge/         client compatibility providers
├── server/bridge/         server compatibility/access providers
├── client/core/           client runtime
├── server/core/           authoritative server core
├── client/modules/        client public API modules
├── server/modules/        admin/commands/vehicles/public SDK
├── web/                   player + Admin Studio NUI
├── docs/                  source reference notes
├── sql/                   schema
├── fxmanifest.lua
└── server.cfg.example
```

Recommended Script Entry Schema

Recommended Script Entry Schema

Developer-only schema for Nord Lab curated entries before publishing a build.

Applies to: `Nord Inventory v2.0.0`

```
{
    name = 'Script Name',
    resource = 'resource_name',
    developer = 'Developer',
    status = 'Verified',
    integration = 'Native', -- or Compatible / future category
    notes = 'Short explanation.',
    url = 'https://example.com', -- optional shorthand
    links = { -- optional, max four safe links after sanitization
        { label = 'Store', url = 'https://example.com/store' },
        { label = 'Documentation', url = 'https://example.com/docs' }
    }
}
```

The catalog is intentionally local to the protected server ecosystem module. It has no runtime write endpoint.

Developer-level inherited links

In v2.0.0 the protected ecosystem module also maintains a developer link registry. Current profiles:

```
['Nord Lab'] = {
    { label = 'Store', url = 'https://nord-lab.tebex.io/' },
    { label = 'Documentation', url = 'https://documentation.lunarocketeer.com/books/' }
}

['Renovax Scripts'] = {
    { label = 'Store', url = 'https://renovax-scripts.tebex.io/' },
    { label = 'Documentation', url = 'https://renovax-scripts.gitbook.io/renovax-scripts-docs'
}
}
```

A recommendation whose `developer` matches a profile inherits those destinations automatically. Entry-specific links remain optional and are sanitized/deduplicated together.

Public API vs Internal Events

Public API vs Internal Events

What third-party integrations are expected to call.

Applies to: `Nord Inventory v2.0.0`

Supported integration surface

- documented exports;
- configuration files intended for server owners;
- provider SDK registration functions;
- explicit bridge/custom provider points.

Do not depend on

- NUI callback names;
- internal net events not documented as public;
- direct writes to Nord tables;
- internal module globals/functions;
- protected implementation details.

This separation lets Nord change internals without forcing every integrated script to update.

Downloadable Reference Files

Downloadable Reference Files

Files attached to this BookStack page after import.

Applies to: `Nord Inventory v2.0.0`

This portable BookStack package includes these attachments:

1. **nord_inventory_v2.0.0_owner_reference.zip** — open/owner-editable configuration, locales, shared registry, documentation references, manifest and release validation files. It intentionally excludes protected runtime client/server/web implementation.
2. **nord_inventory.sql** — current database schema shipped with v2.0.0.
3. **server.cfg.example** — ACE/startup example shipped with the resource.
4. **COMPLETE_EXPORT_INDEX.txt** — generated export/reference index for the documented API surface.
5. **MANIFEST.json** — documentation package manifest for v2.0.0.

The reference ZIP intentionally does **not** contain the protected server runtime implementation.