Nord Ecosystem

- Overview · Nord Ecosystem
- Nord Staff Integration
- Nord Inventory + Nord CarPlay
- Nord Inventory + Nord Crafting
- Nord Inventory + Nord Restaurant Creator
- Nord Inventory + Nord Keys
- Recommended Scripts · Current Catalog
- Recommended Scripts · Tabs, Search & Links

Overview · Nord Ecosystem

7 · Nord Ecosystem

Native Nord integrations and the protected recommended-script catalog.

Nord Staff Integration

Nord Staff Integration

Automatic admin-access bridge plus shared item/provider APIs.

Applies to: `Nord Inventory v2.0.0`

Nord Staff Integration

Admin access bridge

Nord Inventory now exposes the exact access-export pattern detected automatically by current `nord_staff` builds:

```
exports['nord_inventory']:HasAdminAccess(playerId)
exports['nord_inventory']:GiveAdminAccess(playerId)
exports['nord_inventory']:RemoveAdminAccess(playerId)
```

No change is required in `nord_staff`. When **Dar acesso** is used, Nord Inventory grants the configured role and all server-side Admin Studio permission checks use it immediately. License-based grants are persisted automatically in `nord_inventory_admin_access`, so they survive resource/server restarts. Removing access removes only the Staff-managed grant; ACE and static license roles are never changed.

Configure the bridge in `config/admin.lua`:

```
Config.AdminAccess.NordStaffBridge = {
    Enabled = true,
    Role = 'admin', -- developer | admin | owner
    AllowedResources = { 'nord_staff' }
}
```

Nord Inventory exposes a canonical API intended for Nord Staff and other admin resources. The integration never uses a visible item label as the spawn ID.

Canonical identity

Every external item object follows:

```
{
    name = 'ammo-rifle2',       -- internal/spawn name
    id = 'ammo-rifle2',
    internalName = 'ammo-rifle2',

    label = 'Rifle Ammo Mk II', -- visible text only
    title = 'Rifle Ammo Mk II'
}
```

`name` is always the real Nord Inventory identifier.

Recommended Nord Staff exports

Integration information

```
local info = exports['nord_inventory']:GetIntegrationInfo()
```

Resolve any incoming item value

```
local internal = exports['nord_inventory']:ResolveItemName('mg_ammo')
-- ammo-rifle2
```

The resolver accepts canonical IDs, configured aliases, and unique labels.

Aliases are owner-editable in:

```
config/items.lua
```

under `Config.ExternalItemAliases`.

Item catalogue

```
local items = exports['nord_inventory']:GetItemCatalog()
```

Each entry includes canonical item identity plus `image`, `imageUrl` and `imageFile`.

Item image

```
local url = exports['nord_inventory']:GetItemImage('bandage')
local file = exports['nord_inventory']:GetItemImage('bandage', 'file')
local data = exports['nord_inventory']:GetItemImage('bandage', 'data')
```

Local item URLs use:

```
https://cfx-nui-nord_inventory/web/images/items/<file>
```

Remote HTTP/HTTPS item images are returned unchanged.

Inventory snapshot

```
local snapshot = exports['nord_inventory']:GetInventorySnapshot(playerId)
```

The snapshot is a slot-sorted array designed for Staff/Admin UIs. Every item uses the canonical internal `name`.

Add / remove

```
exports['nord_inventory']:AddItem(playerId, internalName, amount, metadata, slot)
exports['nord_inventory']:RemoveItem(playerId, internalName, amount, metadata, slot)
```

The Nord Staff bridge resolves the item through Nord before either operation.

Why this fixes Staff spawn errors

Older bridges commonly mixed:

```
name = item.label
id = item.name
```

That makes a Staff UI send a display title instead of the spawn identifier. The new contract keeps visible text in `label`/`title` and reserves `name` for the canonical Nord ID.

Nord Inventory + Nord CarPlay

Nord Inventory + Nord CarPlay

Recommended integration pattern for device/item ownership.

Applies to: `Nord Inventory v2.0.0`

Use canonical Nord item IDs and metadata to represent device identity where required. CarPlay should query/add/remove items through public exports and keep technical identifiers hidden from player-facing metadata details where appropriate.

If the integration registers runtime items, register an integration manifest/provider so Admin Studio can identify ownership.

Nord Inventory + Nord Crafting

Nord Inventory + Nord Crafting

Native crafting integration principles.

Applies to: `Nord Inventory v2.0.0`

`nord_crafting` should use Nord Inventory as the material/output authority:

- resolve canonical item names;
- check counts/carry capacity server-side;
- remove ingredients only as part of a validated craft transaction;
- add outputs through `AddItem`;
- register custom categories/catalogs at runtime when useful;
- keep metadata/lifecycle/rarity in the resulting item instance.

The current curated catalog lists `nord_crafting` as **Verified / Native**.

Nord Inventory + Nord Restaurant Creator

Nord Inventory + Nord Restaurant Creator

Native restaurant item/stash/recipe integration.

Applies to: `Nord Inventory v2.0.0`

The restaurant system can register food/ingredient catalogs, dynamic stashes and usable items through the SDK. Startup ordering should allow Nord Inventory to be ready before restaurant item registration, or the restaurant should retry registration safely.

The current curated catalog lists `nord_restaurants` as **Verified** / **Native**.

Nord Inventory + Nord Keys

Nord Inventory + Nord Keys

Vehicle lock state used by live trunk session security.

Applies to: `Nord Inventory v2.0.0`

`Config.VehicleInteraction.LockIntegration.Provider = 'auto'` supports automatic lock-provider resolution. The current lock bridge recognizes `nord_vehiclekeys` and can also read the entity state key `nordkeys:locked` where available.

A custom lock provider should expose a reliable server-side lock state. Trunk UI access alone must not bypass a subsequently locked vehicle.

The curated catalog entry is named **Nord Keys** with resource `nord_keys`; the lock bridge can also detect the vehicle-key provider used by the server build.

Recommended Scripts · Current Catalog

Recommended Scripts · Current Catalog

The protected Nord Lab curated catalog included in this build.

Applies to: `Nord Inventory v2.0.0`

Script	Resource	Developer	Status	Integration
Metal Detector	`metaldetector`	Renovax Scripts	Verified	Compatible
Outfit Bag	`outfitbag`	Renovax Scripts	Verified	Compatible
Outfit Bag V2	`outfitbag_v2`	Renovax Scripts	Verified	Compatible
Drug Sell V2	`drugsell_v2`	Renovax Scripts	Verified	Compatible
Sell Drugs	`selldrugs`	Renovax Scripts	Verified	Compatible
Dustbin Search	`dustbin_search`	Renovax Scripts	Verified	Compatible
Trucker Job	`trucker_job`	Renovax Scripts	Verified	Compatible
Warehouse System	`warehouse_system`	Renovax Scripts	Verified	Compatible
Delivery System	`delivery_system`	Renovax Scripts	Verified	Compatible
Crafting System	`crafting_system`	Renovax Scripts	Verified	Compatible
Nord Crafting	`nord_crafting`	Nord Lab	Verified	Native
Nord Restaurant Creator	`nord_restaurants`	Nord Lab	Verified	Native
Nord Keys	`nord_keys`	Nord Lab	Verified	Native

Server owners can hide/show the modal with `Config.Ecosystem.ShowRecommendedScripts`; they cannot edit the protected catalog through Admin Studio.

Current developer destinations

- **Nord Lab** — Store: `https://nord-lab.tebex.io/` · Docs: `https://documentation.lunarocketeer.com/books/`
- **Renovax Scripts** — Store: `https://renovax-scripts.tebex.io/` · Docs: `https://renovax-scripts.gitbook.io/renovax-scripts-docs`

In v2.0.0 these destinations are inherited from the developer-level link registry and appear automatically on all current recommendations from the same developer.

Recommended Scripts · Tabs, Search & Links

Recommended Scripts · Tabs, Search & Links

How compatibility grouping and optional browser links work.

Applies to: `Nord Inventory v2.0.0`

Automatic tabs

The NUI derives tabs from unique `integration` values. This means adding a future curated value such as `Bridge` can automatically create a new tab without changing the HTML structure.

Search

Search covers script name, resource, developer, status, integration, notes and available link labels/URLs.

One external link

```
{
    name = 'Example',
    resource = 'example_resource',
    developer = 'Example Dev',
    status = 'Verified',
    integration = 'Compatible',
    notes = 'Example integration.',
```

```
    url = 'https://example.com'
}
```

Multiple links

```
links = {
    { label = 'Store', url = 'https://example.com/store' },
    { label = 'Documentation', url = 'https://example.com/docs' },
    { label = 'Discord', url = 'https://example.com/community' }
}
```

Limits: `http/https` only, max URL length 1024, max label length 48, deduplicated URLs, maximum four links per entry. The browser action uses FiveM's native URL opener when available with a normal browser-window fallback.

All visible labels/states must remain represented in both EN/PT locale JSON files.

Current developer destinations

- **Nord Lab** — Store: `https://nord-lab.tebex.io/` · Docs: `https://documentation.lunarocketeer.com/books/`
- **Renovax Scripts** — Store: `https://renovax-scripts.tebex.io/` · Docs: `https://renovax-scripts.gitbook.io/renovax-scripts-docs`

In v2.0.0 these destinations are inherited from the developer-level link registry and appear automatically on all current recommendations from the same developer.