Admin Studio & Item Management

- Overview · Admin Studio & Item Management
- Admin Studio Overview
- Permission Matrix
- Item Registry & Categories
- Creating & Editing Items
- Internal & External Use Behavior
- Lifecycle Editor
- Metadata Editor
- Rarity Editor
- Imports & Migration Center
- DB-Authoritative Definitions & Precedence
- Delete, Disable & Restore Items
- Vehicle Registry
- Integrations & Provider Registry
- System Health, Backups & Audit
- Recommended Scripts Modal

Overview · Admin Studio & Item Management

3 · Admin Studio & Item Management

Manage definitions, roles, imports, providers, health, backups, vehicles and the curated ecosystem catalog.

v2.0.0 extends Admin Studio with the Operations Center and aligned Integrations actions.

Admin Studio Overview

Admin Studio Overview

Server-authoritative administration for items, providers, diagnostics and vehicles.

Applies to: `Nord Inventory v2.0.0`

Admin Studio System

Admin Studio is server-authoritative. Hiding a button in NUI is only UX; every mutation is checked again on the server.

Roles

Default ACE objects:

```
add_ace group.dev nord_inventory.dev allow
add_ace group.admin nord_inventory.admin allow
add_ace group.owner nord_inventory.owner allow
```

Default capability levels:

- **Developer**: view Admin Studio, integrations, health diagnostics and item testing.
- **Admin**: Developer access plus item/category management, imports, vehicle storage, backup creation/deletion and audit log.
- **Owner**: Admin access plus backup restore.

The exact ACE names and optional license-based roles are configured in the server-only `config/admin.lua`.

Integrations page

Shows resources that register through Nord APIs, including:

- provider resource and display name
- online/offline state
- provider version/author when supplied
- items associated with the provider
- custom action and image counts
- dependency state
- registration failures/conflicts

The **Recommended Scripts** catalog opens in a read-only modal from the Integrations page. It is maintained only by Nord Lab developers in protected server source; no Admin Studio role, command, event or public export can edit its contents. Server owners can only hide/show the read-only button with `Config.Ecosystem.ShowRecommendedScripts` in `config/core.lua`.

Recommendations are curated by **Nord Lab**. Server owners, administrators and developers can never add, edit or remove catalog entries. The only public option is visibility: `Config.Ecosystem.ShowRecommendedScripts = true/false` in `config/core.lua`.

The catalog data lives in the protected server core and is updated by Nord Lab as part of inventory releases.

System Health

Health Check reports problems such as:

- item image missing from `web/images/items`
- unknown category
- durability item not unique/non-stackable
- malformed external export
- export/provider/dependency offline
- runtime item without a tracked provider
- provider registration conflict
- orphan item action
- invalid/offline action export
- orphan/offline metadata schema provider

The report is diagnostic only; it does not silently rewrite definitions.

Backups and restore

A backup contains Nord-managed configuration:

- database-authoritative custom item definitions
- disabled reference-item markers
- custom categories
- vehicle storage categories
- vehicle model overrides

Before restore, Nord automatically creates a **pre-restore safety backup**. Runtime items supplied by external resources are not persisted as authoritative database definitions; active providers are reapplied after the registry reload.

Restore is Owner-only by default.

Audit log

The server records administrative changes such as item/category/vehicle mutations, imports and backup operations. Audit data is only sent to roles with the `audit` permission.

Migration Center

The import UI supports:

- ox_inventory
- QB/QBCore
- ESX
- Quasar
- CodeM
- Nord JSON

Files are read by the NUI as text/data; imported Lua is not executed by Nord. Always review the conversion preview and warnings before applying a large migration.

Database-authoritative conflicts can be skipped or explicitly overwritten when the user's role allows imports. Reference/source items remain protected from direct source-file rewriting.

Item Templates

Admin Studio includes templates for common definitions such as food, drink, medical, reusable tool, document, container, durability repair item and external usable item.

Templates are starting points only; review the resulting definition before production use.

Testing Mode

Authorized developers/admins can select an item and exercise controlled testing flows from System Health. Testing uses the real server-authoritative inventory path, so it is suitable for verifying use handlers, metadata and durability behavior before rollout.

Source-aware item management

Nord distinguishes the source of each item:

- **Reference (`shared/items.lua` / `shared/weapons.lua`)**: deleting from Admin Studio disables the reference instead of rewriting the Lua file.
- **Reference + DB override**: delete removes the database override and leaves the reference disabled.
- **Database-only**: delete removes the custom database row.
- **Runtime provider item**: protected from Admin Studio deletion; its provider owns unregister/cleanup.

Restoring a disabled reference returns to the original Lua definition.

Permission Matrix

Permission Matrix

Exact role hierarchy used by the v2.0.0 server permission bridge.

Applies to: `Nord Inventory v2.0.0`

Roles are ordered:

```
developer < admin < owner
```

Capability	Minimum role
View Admin Studio	developer
Integrations/provider registry	developer
System health	developer
Test items	developer
Manage/create/edit items	admin
Delete/disable items	admin
Import/migrate items	admin
Manage vehicle registry	admin
Create backups	admin
Audit log	admin
Operations / Forensics	**admin**
Inventory Manager / offline_inventory	**admin**
Create/delete snapshots	**admin**
Restore snapshots	**admin**
Restore full Admin Studio backups	owner

The permission keys added for v2.0.0 are `forensics`, `offline_inventory` and `restore_snapshots`. The older `offline_inventory` name is intentionally retained for backwards compatibility even though the manager now handles both online and offline players.

Runtime Nord Staff grants are persisted without rewriting ACE or static license lists.

Item Registry & Categories

Item Registry & Categories

Built-in categories and the provider-aware item registry.

Applies to: `Nord Inventory v2.0.0`

Built-in categories in the current shared registry include:

- `generic`
- `food`
- `drink`
- `medical`
- `weapon`
- `ammo`
- `tool`
- `document`
- `evidence`
- `container`
- `clothing`
- `key`
- `technology`

Categories can also be registered at runtime by trusted providers. Provider ownership is tracked so one resource cannot silently masquerade as another provider when using the public SDK.

Creating & Editing Items

Creating & Editing Items

Core item fields and DB-authoritative editing workflow.

Applies to: `Nord Inventory v2.0.0`

Admin Studio supports quick creation and detailed editing. Depending on the item, common fields include:

- canonical item name and label;
- category;
- weight/stack/close behavior;
- image and presentation properties;
- internal/external use behavior;
- metadata defaults/schema/display rules;
- durability and expiration/lifecycle behavior;
- container settings;
- rarity;
- provider ownership/source state.

After a DB-backed admin edit, the database definition is intended to remain authoritative across restarts instead of being overwritten by a runtime registration using the same canonical name.

Internal & External Use Behavior

Internal & External Use Behavior

Choose one clear authority for item use.

Applies to: `Nord Inventory v2.0.0`

External item use exports

Nord Inventory has two mutually exclusive usable-item modes:

- `internal`: Nord handles duration, animation, hand prop, player effects and optional removal.
- `export`: Nord validates the item/slot transaction and delegates the gameplay action to another resource export.

The Admin Studio exposes this under **Item Management → Behavior → Use handler** with **Internal (Nord)** and **External export**.

OX-style item definition

A resource can register an item and point its use directly at one of its own exports. When the export name is bare, Nord automatically binds it to the resource calling `RegisterItem`.

```
exports.nord_inventory:RegisterItem('scanner', {
    label = 'Scanner',
    weight = 250,
```

```
    client = {
        export = 'UseScanner', -- automatically <calling_resource>:UseScanner
        remove = 0
    }
})

exports('UseScanner', function(item, slot)
    -- item.name, item.label, item.metadata, item.count, item.slot, ...
    -- Return false to cancel the transaction.
    return true
end)
```

The full resource name is still accepted:

```
client = { export = 'my_scanner:UseScanner', remove = 0 }
-- or
client = { export = 'my_scanner.UseScanner', remove = 0 }
```

Server export

```
exports.nord_inventory:RegisterItem('repair_kit', {
    label = 'Repair Kit',
    weight = 500,
    close = true,
    server = {
        export = 'UseRepairKit',
        remove = 1
    }
})

exports('UseRepairKit', function(source, item, slot, definition)
    -- Return false to cancel and prevent Nord from removing the item.
    return true
end)
```

The Nord-native descriptor is also supported:

```
exports.nord_inventory:RegisterItem('repair_kit', {
    label = 'Repair Kit',
    weight = 500,
    use = {
        mode = 'export',
        export = 'my_mechanic:UseRepairKit',
        side = 'server',
        remove = 1
    }
})
```

`mode = 'export'` may be omitted when `use.export` is present.

Register an external handler for an existing item

Compact form, bound to the calling resource:

```
exports.nord_inventory:RegisterUseExport('repair_kit', 'UseRepairKit', 'server')
```

Explicit resource form:

```
exports.nord_inventory:RegisterUseExport('repair_kit', 'my_mechanic', 'UseRepairKit',
'server')
```

Descriptor/string forms through `RegisterUsableItem` are also accepted:

```
exports.nord_inventory:RegisterUsableItem('repair_kit', 'UseRepairKit', 'server')
exports.nord_inventory:RegisterUsableItem('repair_kit', 'my_mechanic:UseRepairKit', 'server')
exports.nord_inventory:RegisterUsableItem('scanner', { export = 'UseScanner', side = 'client'
})
```

A normal server Lua callback remains supported:

```
exports.nord_inventory:RegisterUsableItem('repair_kit', function(source, item, definition,
slot)
    return true
```

```
end)
```

Register the item and handler in one call

The third `RegisterItem` argument can be a function, export string or export descriptor. The optional fourth argument selects `server` or `client` for string handlers.

```
exports.nord_inventory:RegisterItem('repair_kit', {
    label = 'Repair Kit',
    weight = 500,
    use = { remove = 1 }
}, 'UseRepairKit', 'server')
```

Bulk registration

`RegisterItems` registers multiple runtime items from one resource and applies the same caller-resource binding rules.

```
local ok, result = exports.nord_inventory:RegisterItems({
    scanner = {
        label = 'Scanner',
        weight = 250,
        client = { export = 'UseScanner', remove = 0 }
    },
    repair_kit = {
        label = 'Repair Kit',
        weight = 500,
        server = { export = 'UseRepairKit', remove = 1 }
    }
})
```

Transaction behavior

Nord validates that the player still owns the same item/slot before the external action is committed. If an export is unavailable, errors, or returns `false`, the use is cancelled and Nord does not remove the item.

`remove` stays server-authoritative:

- `remove = 0`: keep the item.
- `remove = 1`: remove one item after a successful external action.
- Higher values remove that amount, limited by the current stack.

External mode is exclusive: Nord does **not** also run its internal duration, animation, hand prop, hunger, thirst, health or armor effects. The external resource owns those effects.

Hands-only animations (`prop = false`)

An internal usable item can explicitly disable every hand-prop fallback while keeping its animation.

```
use = {
    duration = 2500,

    animation = {
        dict = 'missheistdockssetup1clipboard@idle_a',
        clip = 'idle_a',
        flag = 49
    },

    prop = false
}
```

`prop = false` is authoritative for the use action. Nord Inventory will not fall back to:

- `worldModel`;
- keyword prop rules;
- category prop rules;
- the generic fallback prop.

The item's `worldModel` can still be kept for ground/drop presentation.

The default `bandage` uses this mode so the player performs a hands-only bandaging animation with no object attached.

Lifecycle Editor

Lifecycle Editor

Durability, shelf life and repair behavior in item definitions.

Applies to: `Nord Inventory v2.0.0`

Lifecycle settings are definition-level defaults applied to item instances where enabled. Use the lifecycle editor instead of custom client-side timers when the state must persist.

Public durability exports are documented in the Developer API chapter. Integrations should modify lifecycle state through those exports rather than editing raw metadata/storage rows.

Metadata Editor

Metadata Editor

Structured metadata defaults, display rules and provider schemas.

Applies to: `Nord Inventory v2.0.0`

Metadata can be free-form, but the SDK also supports `RegisterMetadataSchema` so an integration can describe expected fields and display behavior. Schemas improve consistency without giving the browser authority over values.

Use hidden metadata display for technical identifiers that scripts need but players do not.

Rarity Editor

Rarity Editor

Opt-in rarity configuration in Admin Studio.

Applies to: `Nord Inventory v2.0.0`

Rarity is not forced onto every item. An item can opt into a configured level and an individual instance can override the base level through metadata when `AllowMetadataOverride = true`.

Current levels: Common, Uncommon, Rare, Epic and Legendary.

Imports & Migration Center

Imports & Migration Center

Preview-first item-definition migrations and compatibility imports.

Applies to: `Nord Inventory v2.0.0`

The migration workspace is for item **definitions**, not a blind database replacement. Preview mappings first, resolve name/category/image/use differences, and keep canonical Nord names stable after import.

For live servers, migrate player item state only after testing a complete mapping in staging and taking a database backup.

DB-Authoritative Definitions & Precedence

DB-Authoritative Definitions & Precedence

Why Admin Studio edits survive runtime registration and restart.

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

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.

Delete, Disable & Restore Items

Delete, Disable & Restore Items

Source-aware item removal without losing track of reference/runtime ownership.

Applies to: `Nord Inventory v2.0.0`

Nord separates built-in/reference definitions, database definitions and provider/runtime definitions. Deletion behavior is source-aware so an admin can disable a reference item, remove a database override or unregister a provider definition without pretending all sources are identical.

Disabled reference entries are persisted in `nord_inventory_disabled_reference_items`. Historical versions are stored in `nord_inventory_item_versions` where applicable.

Vehicle Registry

Vehicle Registry

Vehicle model/category capacity management in Admin Studio.

Applies to: `Nord Inventory v2.0.0`

The vehicle workspace can combine framework vehicle data with GTA model scanning and Nord overrides. Capacity is resolved from model/category/default rules rather than hard-coded only by plate.

`Config.VehicleRegistry.AutoImportFramework = true` and `ScanGameVehicles = true` in the current build. Use model-specific overrides only when category defaults are insufficient.

Integrations & Provider Registry

Integrations & Provider Registry

Runtime provider ownership, health and registration visibility.

Applies to: `Nord Inventory v2.0.0`

The Provider Registry shows resources that register items, categories, actions, schemas or images through the developer SDK. Provider ownership is useful for diagnostics: server owners can see where a runtime item came from instead of guessing which resource injected it.

Use `GetProviderRegistry()` and `GetIntegrationInfo()` for trusted programmatic diagnostics.

v2.0.0 header actions

Recommended scripts and **Refresh** are presented as one aligned action group in the Integrations provider header. Their callback/provider behavior is unchanged; the change is layout/responsiveness only.

The dependency graph in Operations uses provider registry data to show active provider/resource relationships.

System Health, Backups & Audit

System Health, Backups & Audit

Operational visibility and recovery controls.

Applies to: `Nord Inventory v2.0.0`

System Health summarizes provider/database/runtime issues. Backups and the audit log support controlled operations and traceability.

Relevant persistent tables include:

- `nord_inventory_backups`
- `nord_inventory_audit_log`
- `nord_inventory_transactions`

Backup creation is an admin operation. Restore requires **owner** role by default.

Recommended Scripts Modal

Recommended Scripts Modal

Read-only curated catalog with automatic compatibility tabs, search and safe browser links.

Applies to: `Nord Inventory v2.0.0`

Recommended Scripts security

The Recommended Scripts catalog is a Nord Lab-owned, read-only catalog.

Who can edit it?

Only Nord Lab developers editing the source before a release. There is no runtime editing path for server owners, Admin Studio administrators, or third-party developers.

Where is it stored?

The catalog is a local table inside `server/core/ecosystem.lua`. It is not in Config or the database.

Production protection

`server/core/ecosystem.lua` is intentionally not listed in `escrow_ignore`. When the package is published through Cfx.re/Tebex Asset Escrow, the file is protected by Asset Escrow.

The raw development/source ZIP is not Asset-Escrow encrypted. Encryption is applied by Cfx.re/Tebex during the publishing process.

Release rule

Never add `server/core/ecosystem.lua` to `escrow_ignore`.

Server owner visibility setting

Server owners cannot edit the Nord Lab curated catalog. They can only hide or show the read-only button in Admin Studio from `config/core.lua`:

```
Config.Ecosystem.ShowRecommendedScripts = true -- true = show, false = hide
```

When disabled, the button is hidden and the curated catalog is not included in the Admin Studio payload. The protected Nord Lab catalog itself remains unchanged inside the Asset Escrow server core.

Optional external links

Nord Lab can attach browser links to a curated recommendation. Links remain part of the protected catalog and are still read-only for server owners.

For one link, use the shorthand:

```
{
    name = 'Example Script',
    resource = 'example_script',
    developer = 'Example Developer',
    integration = 'Compatible',
    url = 'https://example.com/script'
}
```

For several destinations, use `links`:

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

```
    { label = 'Documentation', url = 'https://example.com/docs' },
    { label = 'Discord', url = 'https://example.com/discord' }
}
```

The server only exposes `http://` and `https://` links to the NUI, removes duplicates and limits each recommendation to four external buttons. The NUI validates the URL again before asking FiveM to open it in the user's external browser.

Current UI behavior

- tabs are created automatically from the `integration` values in the catalog;
- `Native`, `Compatible`, and future values such as `Bridge` can appear without hard-coding a fixed tab list;
- each tab shows its result count;
- search matches name, resource, developer, status, integration, notes and link metadata;
- cards show status, resource, developer and compatibility clearly;
- all new visible labels are represented in EN/PT locales;
- external buttons open the system browser, not an embedded untrusted webpage.

Link safety

Only `http://` and `https://` URLs are emitted. URLs are trimmed/deduplicated, capped in length, and each catalog entry exposes at most four buttons. NUI validates the URL again before opening it.

v2.0.0 developer link registry

Current curated developers inherit shared Store/Documentation links by developer name, so the same URLs do not need to be repeated in every recommendation.

Developer	Store	Documentation
Nord Lab	`https://nord-lab.tebex.io/`	`https://documentation.lunarocketeer.com/books/`
Renovax Scripts	`https://renovax-scripts.tebex.io/`	`https://renovax-scripts.gitbook.io/renovax-scripts-docs`

Per-entry `url`/`links` remain supported and are merged through the same sanitizer. Only `http/https` URLs are exposed, duplicates are removed and a recommendation exposes at most four link buttons.