Player Experience

- Overview · Player Experience
- Inventory UI & Dual Inventory
- Fast Slots, Hotbar & Special Slots
- Player Shortcuts
- Received Item Feedback
- Ground Items, Pickup & Throw
- Container Items & Backpacks
- Metadata, Durability & Expiration
- Metadata Privacy & Hidden Fields
- Durability Repair Items
- Rarity System & Slot Presentation
- Item Props & Use Animations
- Large Weapons on the Back
- Item Context Actions
- Personal Studio
- Player Search & Robbery
- Hands-Up Integration
- Equipment Slots 2.0

Overview · Player Experience

2 · Player Experience

Player inventory behavior, quick slots, lifecycle, item visuals, world drops and player-to-player interactions.

Inventory UI & Dual Inventory

Inventory UI & Dual Inventory

Player inventory behavior and the left/right storage model.

Applies to: `Nord Inventory v2.0.0`

Nord uses a dual-inventory layout: the player inventory remains on the left, while the active stash/container/vehicle/ground/player inventory is shown on the right. Server sessions determine what the right-hand inventory represents and when that session must close.

Key properties

- slot-based item instances;
- weight authority on the server;
- per-instance metadata;
- Fast Slots and dedicated special slots;
- drag/drop and context actions;
- live item feedback without requiring the full inventory to be open;
- storage session invalidation when access conditions change.

The default player capacity is **40 slots / 80,000 g**.

Fast Slots, Hotbar & Special Slots

Fast Slots, Hotbar & Special Slots

Default slots 1–7 and quick-use behavior.

Applies to: `Nord Inventory v2.0.0`

Special Slots & Quick Actions

Nord Inventory reserves two dedicated real player inventory slots after the Fast Slots.

Default slots

```
Config.UI.specialSlots = {
    enabled = true,
    slots = {
        { key = 'keys', slot = 6, category = 'key', label = 'Keys', icon = 'fa-key' },
        { key = 'technology', slot = 7, category = 'technology', label = 'Technology', icon = 'fa-microchip' }
    }
}
```

With the normal 5 Fast Slots:

- slots 1–5 = Fast Slots;
- slot 6 = Keys;

- slot 7 = Technology;
- normal inventory grid begins at slot 8.

The server validates the item category before every move. Client-side validation is only visual feedback.

Existing inventories

When an older player inventory already has an incompatible item in slot 6 or 7, Nord Inventory relocates it to the first free normal slot on load. If the inventory is completely full, the item is never deleted: it remains visible in the special slot with a warning until the player moves it out.

Amount control

The bottom `Amount` field controls drag quantity.

Example:

- stack contains 10;
- Amount = 3;
- drag the stack to another empty slot;
- server moves 3 and leaves 7 in the original slot.

This works for same-inventory splitting and cross-inventory transfers.

Use

Select a player item and press **Use**. Containers open normally; usable items call the standard server-authoritative use flow.

Give

Select a player item and press **Give**. The configured amount is given. If several players are within 3 meters, a compact player chooser is shown.

Unified bottom dock

Fast Slots, Special Slots, Amount, Use and Give share the same bottom row.

Personal Studio > Layout now controls the bottom player tools with:

- Fast Slots width;
- Player tools size;
- Fast / tools spacing.

The new tools also inherit the player's slot opacity, border opacity, corner radius, item image size, accent and icon color settings.

Personal Studio · Player Tools

The complete right-side bottom group is configurable from a dedicated **Player Tools** page in Personal Studio.

Per-player settings include:

- overall Player Tools size;
- Special Slot size;
- Amount control size;
- Use / Give button size;
- spacing between Fast Slots and Player Tools;
- Special Slot label visibility;
- helper-text visibility.

Appearance and Layout settings still provide the shared slot opacity, border visibility, corner radius, accent, item-image scale and icon color.

Drag directly to Use / Give

Use and Give are real drag targets.

- Drag a player-inventory item onto **Use** to execute the standard use flow.
- Drag a player-inventory item onto **Give** to open/execute the nearby-player flow.
- **Give** respects the Amount control.
- Items dragged from a secondary inventory must first be moved into the player inventory.
- Pinned Fast Slot items cannot be given until unpinned.

Quantity zero means all

The Player Tools Amount input accepts `0`.

- `0` = the complete current stack;
- `1+` = exactly that quantity, clamped to the available stack;
- dragging with `0` moves the full stack;
- dragging with a smaller value splits the stack;
- Give uses the same rule, so `0` gives the complete selected stack.

The NUI always resolves `0` to a positive server-side amount before sending the action, so the inventory core does not need unsafe zero-amount transfers.

```
-- Nord Inventory - player UI, hotbar and received-item feedback

Config = Config or {}

Config.UI = {
    theme = 'dark',
    columns = 7,
    slotSize = 'medium',
    showWeight = true,
    animations = true,

    -- Slots 1-5 are reserved for the normal Fast Slots.
    hotbarSlots = 5,
    hotbarKeybinds = true,

    -- Dedicated player slots shown in the bottom action dock.
    -- They are REAL inventory slots and are validated server-side by category.
    -- With the default 5 Fast Slots these reserve slots 6 and 7.
    specialSlots = {
        enabled = true,
        slots = {
            {
                key = 'keys',
                slot = 6,
                category = 'key',
                label = 'Keys',
                icon = 'fa-key'
```

```
            },
            {
                key = 'technology',
                slot = 7,
                category = 'technology',
                label = 'Technology',
                icon = 'fa-microchip'
            }
        }
    }
}

-- Standalone quick hotbar. This is independent from the full inventory NUI and
-- never takes mouse/keyboard focus, so the player can keep moving normally.
Config.Hotbar = {
    Enabled = true,
    Command = 'hotbar',
    OpenKey = 'Z',
    Mode = 'timed', -- timed | toggle
    Duration = 3200, -- milliseconds when Mode = timed
    ShowOnSlotUse = false -- automatic use feedback uses compact item cards, not the full hotbar; pressing 1-5 also reveals the hotbar
}

-- Compact OX-style item feedback. Multiple different item cards can be shown
-- at the same time, so reward/crafting bursts no longer wait in a long queue.
-- Repeated feedback for the same item is merged into the visible card.
Config.ItemNotifications = {
    Enabled = true,
    Receive = true,
    Duration = 1800, -- milliseconds each visible card remains on screen
    MaxVisible = 8   -- safety cap for simultaneous cards; oldest is replaced first
}
```

Player Shortcuts

Player Shortcuts

Default keyboard shortcuts and commands.

Applies to: `Nord Inventory v2.0.0`

Player shortcuts

Hover + E quick transfer

With two inventories open, place the mouse directly over an item and press `E`.

- Player item → open ground/trunk/glovebox/stash/container.
- Item in the open secondary inventory → player inventory.
- The complete stack is transferred.
- A matching stack is preferred; otherwise the first valid free slot is used.
- Fast/Special player slots are never chosen automatically.
- Pinned items and search/rob restrictions keep their normal protection.

The shortcut does nothing when the mouse is not over an item.

Inventory open / close

- `TAB`: opens the player inventory when closed and closes it when open.
- `ESC`: closes the player inventory immediately.

The TAB close is handled inside the NUI because, while the inventory has keyboard focus, the browser receives the key instead of FiveM's normal key mapping.

Received Item Feedback

Received Item Feedback

Compact item notifications for rewards, crafting and received items.

Applies to: `Nord Inventory v2.0.0`

Current defaults:

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

Different item notifications can be visible simultaneously. Repeated feedback for the same item is merged into the visible card. `MaxVisible` caps concurrent cards and the oldest is replaced first when the cap is reached.

Ground Items, Pickup & Throw

Ground Items, Pickup & Throw

Server-synchronized ground drops and safe throw limits.

Applies to: `Nord Inventory v2.0.0`

Ground behavior is configured in `config/world.lua` and is server-authoritative. The current build uses a default cardboard-box fallback prop and category/definition-aware visual models where available.

Core defaults include:

- open distance: **3.0 m**;
- draw distance: **15.0 m**;
- prop distance: **35.0 m**;
- lifetime: **1800 seconds**;
- ground inventory: **30 slots / 250,000 g**;
- maximum throw distance: **8.0 m**;
- maximum throw height difference: **4.0 m**.

The server validates actual inventory removal and resulting ground state. Client-side throw presentation is not treated as authoritative.

Container Items & Backpacks

Container Items & Backpacks

Items that own their own nested inventory.

Applies to: `Nord Inventory v2.0.0`

An item definition may expose a container inventory. Container state is tied to an item instance and opened through the server API.

```
local ok, result = exports.nord_inventory:OpenItemContainer(source, slot)
```

Use `GetContainerInventory(target, slot)` when a trusted server integration needs to inspect container storage.

`Config.MaxContainerDepth = 3` prevents unbounded nesting. Integrations should never attempt to bypass the configured depth by directly constructing database identifiers.

Metadata, Durability & Expiration

Metadata, Durability & Expiration

Per-instance data, lifecycle state and presentation.

Applies to: `Nord Inventory v2.0.0`

Every item instance can carry arbitrary metadata. Common Nord-managed fields may include durability/lifecycle data, weapon magazine state, container identity, rarity override, custom label/image presentation, and integration-specific values.

Use `SetMetadata`, `UpdateMetadata` and `GetMetadata` from trusted server code. Never trust a metadata payload sent directly by a browser/NUI or arbitrary client event.

`Config.EnableDurability = true` by default. Item history is also enabled in the current build.

Metadata Privacy & Hidden Fields

Metadata Privacy & Hidden Fields

How integrations can store internal identifiers without exposing them in item details.

Applies to: `Nord Inventory v2.0.0`

Metadata may be required for identity while still being inappropriate to show in the player item details UI. Item definitions can suppress detailed metadata presentation while retaining server-side metadata.

The included phone-style reference item demonstrates this pattern with `hideMetadataDetails = true` and persistent metadata defaults.

Use this for device IDs, internal ownership references, signed values, provider state or other technical fields which should remain available to scripts but not clutter player UI.

Durability Repair Items

Durability Repair Items

Internal repair consumables and the public durability API.

Applies to: `Nord Inventory v2.0.0`

Durability repair items

Nord Inventory can create internal usable items that restore durability on a specific target item without relying on another script.

Admin Studio

In **Behavior → Use behavior**, keep the handler on **Internal (Nord)** and set:

- **Item effect:** `Restore durability`
- **Target item:** choose an item that has durability enabled
- **Restore amount:** percentage points restored per successful use
- **Remove on use:** enable it if the repair item should be consumed

Nord only consumes the repair item when durability was actually restored.

Lua definition

```
['weapon_cleaning_kit'] = {
    label = 'Weapon Cleaning Kit',
    weight = 250,
    stack = true,
    close = true,
```

```
    use = {
        duration = 3500,
        actionText = 'Repairing with {item}',
        remove = 1,
        repair = {
            item = 'weapon_pistol',
            amount = 25
        }
    }
}
```

Server-authoritative behavior

When the repair item is used successfully, Nord Inventory:

1. validates that the configured target definition exists and uses durability;
2. searches the player's inventory for matching target instances;
3. selects the most damaged matching instance;
4. restores the configured amount without exceeding the target maximum;
5. consumes the repair item only after a real repair is applied;
6. updates inventory watchers immediately.

The same effect works with weapons and normal items because both use the standard `metadata.durability` state.

Safe failure cases

The use is rejected without consuming the repair item when:

- the player does not have the configured target item;
- the target definition does not use durability;
- every matching target is already at maximum durability;
- the repair configuration is invalid.

If several copies of the target exist, the most damaged copy is repaired first.

Runtime registration example

External resources can register repair items through the normal runtime API:

```
exports.nord_inventory:RegisterItem('pistol_cleaning_kit', {
    label = 'Pistol Cleaning Kit',
    weight = 200,
    stack = true,
    close = true,
    use = {
        duration = 3000,
        remove = 1,
        repair = {
            item = 'weapon_pistol',
            amount = 30
        }
    }
})
```

The repair itself remains server-authoritative inside Nord Inventory.

Rarity System & Slot Presentation

Rarity System & Slot Presentation

Opt-in rarity, metadata override and current levels.

Applies to: `Nord Inventory v2.0.0`

Item Rarities

Nord Inventory includes an opt-in rarity presentation system for item definitions and individual item instances.

Important: rarity is disabled by default

An item only has rarity when its definition explicitly contains a valid `rarity` value, or when an individual instance provides `metadata.rarity` and metadata overrides are enabled.

This item has no rarity:

```
['water'] = {
    label = 'Water',
    weight = 500
}
```

This item has rarity enabled:

```
['special_key'] = {
    label = 'Special Key',
    weight = 5,
    rarity = 'rare'
}
```

`Config.Rarities.Default` is only the level selected when rarity is enabled in Admin Studio. It is not automatically applied to all items.

Slot presentation

When rarity is enabled on an item:

- The slot border and subtle inner tint use the rarity color.
- A compact rarity icon appears in the slot corner.
- The rarity name is **not** shown on the slot.
- Hotbar and item feedback use the same icon/color identity.
- The Item Information modal can still show the readable rarity label.

Creating rarity levels, icons and colors

Edit `config/rarities.lua`:

```
Config.Rarities.Levels = {
    rare = {
        label = 'Rare',
        color = '#4f8cff',
        icon = 'fa-gem',
        order = 3
    },

    mythic = {
        label = 'Mythic',
        color = '#ff4fd8',
```

```
        icon = 'fa-star',
        order = 6
    }
}
```

`icon` uses a Font Awesome Free Solid icon class. Examples:

```
fa-circle
fa-leaf
fa-gem
fa-star
fa-crown
fa-diamond
fa-wand-magic-sparkles
```

New configured levels automatically appear in **Admin Studio → Item Management → Rarity**.

Admin Studio

Rarity now has its own page in the item editor.

The page contains:

- An explicit **Enable rarity** toggle.
- A live slot preview.
- Rarity level cards showing each configured icon and color.
- The current color and level information.

New items start with rarity disabled.

Runtime registration

```
exports.nord_inventory:RegisterItem('special_key', {
    label = 'Special Key',
    weight = 5,
    rarity = 'legendary'
})
```

To register an item without rarity, simply omit the field.

Instance rarity override

When `Config.Rarities.AllowMetadataOverride = true`, a specific item instance may provide its own rarity:

```
exports.nord_inventory:AddItem(source, 'special_key', 1, {
    rarity = 'legendary'
})
```

This lets one physical instance have a rarity identity without requiring another item definition.

Rarity remains presentation-first

Rarity does not automatically change:

- damage
- weight
- price
- durability
- drop chance
- crafting output

Other resources may read the rarity value and build gameplay rules on top of it if desired.

Slot background behavior

When rarity is enabled on an item, the **slot background itself** uses the configured rarity color.

Rarity does **not** control:

- durability bar color;
- validity/expiry bar color;
- interaction/selected border color;
- item label color;
- item image.

Those remain controlled by the player's normal inventory customization.

The implementation intentionally avoids CSS `color-mix()` for the slot background so the result remains reliable in FiveM CEF builds.

Slot opacity integration

When a rarity is enabled on an item, the slot uses the **same transparency/opacity system** as the player inventory editor.

This means:

- the rarity changes only the **slot color**;
- the slot still respects the player's configured `slotOpacity`;
- the border still uses the player's normal border/interation styling;
- the durability bar keeps using the player's lifecycle colors.

In practice, the normal slot background:

```
rgb(8 8 13 / var(--user-slot-opacity))
```

is replaced by the rarity color equivalent:

```
rgb(R G B / var(--user-slot-opacity))
```

So the opacity behavior remains the same and only the base color comes from the rarity definition.

Slot opacity behavior with rarity

The rarity slot background keeps its rarity color/tint, but all of its background layers now obey the same player `Slot opacity` slider used by normal slots.

This means:

- lowering **Slot opacity** also lowers the visibility of rarity backgrounds;
- raising **Slot opacity** makes rarity backgrounds stronger;
- the rarity color remains visible;
- the durability bar remains independent.

Current configured levels

- `common`

- `uncommon`
- `rare`
- `epic`
- `legendary`

Item Props & Use Animations

Item Props & Use Animations

Definition-driven held props and safe fallbacks.

Applies to: `Nord Inventory v2.0.0`

`config/items.lua` controls generic/category/keyword prop fallback plus model-specific hand offsets. Normal tools/objects can use the Pocket transition and weapons use a dedicated draw/holster transition.

Ammo and magazine items are intentionally excluded from generic held-prop fallback because their primary visual feedback is the weapon reload itself.

Integrations should define item-specific presentation where accuracy matters, and rely on fallback only for generic items.

Large Weapons on the Back

Large Weapons on the Back

Cosmetic long-weapon display for unequipped inventory weapons.

Applies to: `Nord Inventory v2.0.0`

Current defaults:

```
Config.BackWeapons = {
    Enabled = true,
    MaxVisible = 2,
    HideInVehicle = true,
    HideWhenDead = true,
    RefreshInterval = 750
}
```

This is presentation only. Weapon ownership, ammunition and durability remain server-authoritative inventory data. Two default back attachment slots use opposite diagonals to reduce model overlap.

Item Context Actions

Item Context Actions

Built-in and provider-registered actions available from an item.

Applies to: `Nord Inventory v2.0.0`

Right-click/context actions are assembled from built-in inventory behavior plus registered provider actions. External resources can add controlled actions with `RegisterItemAction` and remove them with `UnregisterItemAction`.

Action visibility should never be treated as permission. The server resource handling the action must still validate player/item state before mutation.

Personal Studio

Personal Studio

Player-facing preferences without changing server-wide authority.

Applies to: `Nord Inventory v2.0.0`

Personal Studio is for player presentation/preferences, not administrative item definitions. It can manage user-level presentation choices such as reload behavior or UI preferences while preserving server policy.

Preferences are persisted separately from inventory item definitions in the `nord_inventory_preferences` table.

Player Search & Robbery

Player Search & Robbery

Server-authoritative player-to-player viewing and one-way transfer.

Applies to: `Nord Inventory v2.0.0`

Player Search & Robbery

Nord Inventory can expose **Search player** and **Rob player** interactions when the target player is surrendering with their hands up.

Security model

The target option is only a client-side convenience. The server revalidates:

- target server ID;
- distance between both players;
- the target's replicated hands-up state;
- the mode (`search` or `rob`);
- every inventory transfer while the session is open.

The session closes automatically if the target lowers their hands, disconnects, or moves outside the configured distance.

Search

Search is read-only.

The searching player can inspect the target inventory, including items stored in Fast Slots and Special Slots, but cannot move items from/to the target inventory.

Rob

Robbery is one-way:

```
victim inventory  ->  robber inventory
```

The robber can never use the session to put/swap their own items into the victim inventory.

Pinned Fast Slot items are not protected from a valid robbery. If stolen, the victim's pin marker is removed before the item enters the robber inventory.

Configuration

Edit:

```
config/player_interactions.lua
```

Main options:

```
Config.PlayerRobbery = {
    Enabled = true,
    Distance = 2.5,
    SearchEnabled = true,
    RobEnabled = true,
    RequireHandsUp = true
}
```

Hands-up detection

Nord automatically detects common hands-up animation dictionaries and publishes:

```
nord_inventory:handsup
```

as a replicated player state.

If your hands-up resource uses a custom animation, add it to `Animations`.

If it already publishes a player state, add the key to `ExternalStateKeys`.

An external client resource can also explicitly tell Nord:

```
exports['nord_inventory']:SetHandsUpState(true)
```

Return to automatic detection:

```
exports['nord_inventory']:SetHandsUpState('auto')
```

Check the local state:

```
local handsUp = exports['nord_inventory']:IsHandsUp()
```

Target providers

When the inventory interaction mode uses Target, Nord registers global-player options for ox_target / qb-target / compatible providers.

When Target is not active, the TextUI fallback uses:

```
E -> Search
G -> Rob
```

The controls and labels are configurable.

Server exports

Trusted server resources can request the same validated flow:

```
exports['nord_inventory']:OpenPlayerSearch(source, target)
exports['nord_inventory']:OpenPlayerRobbery(source, target)
exports['nord_inventory']:IsPlayerSurrendered(target)
```

Reliable interaction mode

Player Search/Rob now has its own interaction selector and does not depend on `Config.Interaction.Mode`:

```
Config.PlayerRobbery.Interaction = 'auto'
```

- `auto`: use ox_target/qb-target when available, otherwise TextUI;
- `target`: target only;
- `textui`: TextUI only.

This prevents a world-interaction setting from silently disabling player robbery.

Built-in hands up

The system now works without a separate surrender resource:

```
X -> toggle hands up
/handsup -> toggle hands up
```

Configure or disable it in `Config.PlayerRobbery.BuiltInHandsUp`.

Remote Search/Rob options only appear after the target's replicated hands-up state is visible. This ensures the server and the robber agree on the surrender state before the interaction can be selected.

Server-synchronized surrender state

Hands-up detection is performed on the surrendering player's own client and sent to the server. The server then broadcasts a compact surrender cache to all clients.

Target/TextUI checks use that server-synchronized cache. This avoids differences between FiveM artifacts in how remote Player state bags and ped Entity state bags are exposed.

A joining/restarting client requests a full snapshot automatically.

Hands-Up Integration

Hands-Up Integration

Built-in surrender state plus compatibility with external hands-up resources.

Applies to: `Nord Inventory v2.0.0`

Current player-interaction defaults include built-in hands-up support:

```
Command: /handsup
Key: X
State key: nord_inventory:handsup
```

External systems can synchronize through:

```
exports.nord_inventory:SetHandsUpState(true)
local surrendered = exports.nord_inventory:IsHandsUp()
```

On the server:

```
local surrendered = exports.nord_inventory:IsPlayerSurrendered(target)
```

Search can be read-only while robbery is one-way from victim to robber. The server revalidates distance, target identity, surrender state and transfer constraints.

Equipment Slots 2.0

Equipment Slots 2.0

Applies to: `Nord Inventory v2.0.0`

Special/equipment slots remain real player inventory slots and are validated server-side. Existing category matching still works, and v2.0.0 adds explicit item allowlists.

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

A slot may use `category = 'technology'` or `items = { ... }` depending on how strict the equipment rule should be. Additional equipment slots are intentionally opt-in: v2.0.0 does not reserve more slots automatically on an upgraded server.

If you add slots 8+ to `Config.UI.specialSlots.slots`, account for them when choosing total player slot count and when migrating old inventories. Incompatible legacy items are relocated to a normal free slot when possible and are never intentionally deleted simply because a special-slot rule changed.