8 · Developer API & Operations

Use public exports safely, understand the database, diagnose common issues and prepare the resource for production.

Client Exports

Admin Studio

```
exports['nord_crafting']:OpenAdminInterface()
```

Probe support used by integrations:

```
local supported = exports['nord_crafting']:OpenAdminInterface({ probe = true })
```

Bench UI

```
exports['nord_crafting']:OpenBench('fixed_1')
exports['nord_crafting']:CloseBench()

local open, current = exports['nord_crafting']:IsBenchOpen()
local bench = exports['nord_crafting']:GetBench('fixed_1')
local cache = exports['nord_crafting']:GetBenchCache()
```

Use these exports instead of manually sending internal NUI messages from another client resource.

Server Exports · Benches

```
exports['nord_crafting']:GetFixedBenches()
exports['nord_crafting']:GetPlacedBenches()
exports['nord_crafting']:GetAllBenches()
exports['nord_crafting']:GetBench('fixed_1')
exports['nord_crafting']:ReloadBenches()
```

Add a placed bench:

```
exports['nord_crafting']:AddPlacedBench(ownerCid, payload, def)
```

Access/opening helpers:

```
local payload, err = exports['nord_crafting']:GetBenchOpenPayload(source, 'fixed_1')
local allowed, reason = exports['nord_crafting']:CanPlayerAccessBench(source, 'fixed_1')
local ok, reason = exports['nord_crafting']:OpenBenchForPlayer(source, 'fixed_1')
```

`OpenBenchForPlayer` performs the normal payload/access path and then opens the bench on the target client.

Server Exports · Inventory

These exports expose Crafting's active inventory bridge to other server resources.

```
local invType = exports['nord_crafting']:GetInventoryType()
local count = exports['nord_crafting']:GetItemCount(source, 'iron')
local ok = exports['nord_crafting']:AddItem(source, 'iron', 5, { quality = 100 })
local ok = exports['nord_crafting']:RemoveItem(source, 'iron', 2)
local slots = exports['nord_crafting']:SearchInventory(source, 'slots', 'blueprint')
local inventory = exports['nord_crafting']:GetInventory(source)
local items = exports['nord_crafting']:GetItemList()
```

The bridge normalizes basic operations across supported inventory providers, but metadata/stash capabilities can vary by provider.

Server Exports · Blueprints & Organizations

Blueprint API

```
exports['nord_crafting']:GetPlayerBlueprintRecipes(source)
exports['nord_crafting']:GetBlueprintDefinition('pistol')
exports['nord_crafting']:BuildBlueprintMetadata(row, 85)
exports['nord_crafting']:CreateBlueprintItem(source, 'pistol', 85)
```

Organization bench API

```
exports['nord_crafting']:CreateNordCrimeOrgBench(orgId, data)
exports['nord_crafting']:CreateOrgBench(orgId, data)
exports['nord_crafting']:AddOrgCraftingBench(orgId, data)
```

Database readiness

```
local ready = exports['nord_crafting']:IsNordDBReady()
```

Server Exports · Admin Runtime

```
local ok, identifier = exports['nord_crafting']:AddAdmin(value)
local ok, identifier = exports['nord_crafting']:RemoveAdmin(value)
local isAdmin = exports['nord_crafting']:IsAdmin(value)
local runtimeAdmins = exports['nord_crafting']:GetRuntimeAdmins()
```

These are useful for a central staff resource that manages access dynamically.

`nord_staff` additionally integrates through the manifest-declared client Admin Studio export, so the two systems can share a single Crafting administration surface.

Database Tables

Nord Crafting creates/manages the following tables in this package:

Table	Purpose
`nord_crafting_settings`	Schema version and runtime options
`nord_crafting_meta`	Crafting metadata/version compatibility
`nord_crafting_blueprints`	Blueprint definitions
`nord_crafting_fixed`	Fixed benches
`nord_crafting_placed`	Portable/placed benches
`nord_crafting_logs`	Crafting log records
`nord_crafting_props`	Custom prop catalog
`nord_crafting_recipes`	Custom recipes
`nord_crafting_levels`	Level tiers
`nord_crafting_player_levels`	Player XP/level
`nord_craft_benches`	Compatibility/legacy bench schema
`nord_craft_recipes`	Compatibility/legacy recipe schema

The automatic initializer uses the configured Crafting version as the database schema version.

Discord Logging

```
Config.DiscordLogs = {
    Enabled = false,
    Webhooks = {
        Craft = '',
        Admin = '',
        Blueprint = '',
        Recipe = '',
        Stash = ''
    }
}
```

Enable only the destinations you actually use and keep webhook URLs private.

Typical logged actions include:

- Successful/failed blueprint operations.
- Admin actions.
- Recipe management.
- Crafting activity.
- Stash access.

Use separate webhooks when you want different Discord channels for player activity and administrative changes.

Troubleshooting

`blueprint` item not found

Nord Inventory: run/verify the Nord Inventory-only automatic installation flow, then confirm the item exists.

Other inventories: install the item definition/image manually from `install/itens/`.

Portable bench cannot be used

Verify the active inventory contains the item configured by:

```
Config.Portable.ItemName
```

Default: `craft_bench`.

No inventory detected

Check `Config.Inventory` and make sure the inventory resource starts before Crafting. For the generic bridge, valid configured modes include OX, QS, QB and ESX.

Bench appears but cannot be opened

Check:

- Player distance.
- Access type and minimum grade.
- Job/gang/org state.
- Bench cache synchronization.

- Assigned recipes.

Craft recipe missing from UI

Confirm the recipe ID is assigned to the selected bench and, when blueprint gating is enabled, that the player has the matching blueprint.

Database does not initialize

Verify `oxmysql` is started, database credentials are valid and the DB user can create/alter tables when automatic migration is enabled.

Admin Studio denied

Add the correct FiveM `license:` identifier to `Config.AdminLicenses` or use the runtime Admin Manager API.

Weapon preview is empty/wrong position

The resource does not create the shell. Correct `Config.WeaponPreviewShell.coords`, `heading`, mode and offsets.

Production Checklist

Before release:

- [ ] `oxmysql` and `ox_lib` start before Crafting.
- [ ] Framework auto-detection reports the expected framework.
- [ ] Inventory integration is tested.
- [ ] **Nord Inventory auto-installer** is verified when using `nord_inventory`.
- [ ] Third-party inventory item setup is completed manually when applicable.
- [ ] `blueprint` item exists.
- [ ] `craft_bench` (or renamed portable item) exists when portable benches are enabled.
- [ ] Admin licenses are correct.
- [ ] Database tables initialize without errors.
- [ ] Standard recipe craft tested.
- [ ] Failed craft behavior is accepted/balanced.
- [ ] Blueprint drafting and use consumption tested.
- [ ] Level progression tested.
- [ ] Job/gang/org bench permissions tested.
- [ ] Bench stash tested.
- [ ] Weapon bench tested if used.
- [ ] Discord webhook secrets are private.
- [ ] GitHub updater token is stored privately and not shipped publicly.
- [ ] `nord_staff` can open Crafting Admin Studio when the integration is enabled.
- [ ] Debug logging is disabled or intentionally configured for production.

Downloadable Reference Files

This page contains the key editable/reference files from the supplied **Nord Crafting v6.2.0** package, plus sanitized setup notes generated for this documentation.

Use the live resource files as the source of truth when you change configuration.

> The private GitHub updater configuration is intentionally **not attached** here because secret tokens should not be redistributed inside documentation exports.