Integrations and API

- Framework Bridge
- Keys, Fuel & Inventory
- Client Exports
- Server Exports — Garages & Vehicles
- Server Exports — Admin, Staff & Sync
- Server Exports — Vehicle OS
- Events & Live Sync
- Nord Phone Integration
- Nord Staff Integration

Framework Bridge

Framework Bridge

Framework-specific code is isolated under `client/bridge/` and `server/bridge/`.

Auto-detection order is:

1. Qbox / QBX
2. QBCore
3. ESX
4. Nord Core
5. Standalone

Framework ownership/economy is bridged while Nord Garages keeps its own authoritative runtime state.

Framework table mapping

The configuration contains default table mappings for ESX and QB/QBX. If your server uses a heavily customized ownership schema, adjust only the bridge/table mapping instead of rewriting the garage core.

Nord Core

The included bridge expects a `GetPlayer(source)` style export. If your local Nord Core API differs, adapt the Nord bridge file while preserving the core API contract.

Keys, Fuel & Inventory

Keys, Fuel & Inventory

Nord Vehicle Keys

When `nord_vehiclekeys` is used, Nord Garages integrates server-authoritatively.

Typical API usage includes registering the owned network entity, checking keys/physical key items and granting keys only when needed.

With `Config.Keys.keepOwnerKeysOnStore = false`, storing/impounding removes the physical vehicle key lifecycle where appropriate, and spawning recreates valid owner access without duplicating physical key items.

Fuel

The fuel bridge restores the saved fuel value during entity initialization, then stops writing so the active fuel resource controls normal consumption.

`lc_fuel` receives extra initialization safety to avoid persisting an uninitialized decorator as a false `0%` tank.

Inventory bridge API

Internally normalized operations include:

```
NordInventory.GetCount(source, item, metadata)
NordInventory.HasItem(source, item, amount, metadata)
NordInventory.GetItem(source, item, metadata)
NordInventory.AddItem(source, item, amount, metadata, slot, reason)
NordInventory.RemoveItem(source, item, amount, metadata, slot, reason)
```

```
NordInventory.RegisterStash(id, label, slots, weight, owner, groups, coords)
NordInventory.OpenStash(source, id, owner)
```

Client Exports

Client Exports

Garage UI

```
exports['nord_garages']:OpenGarage(identifier)
exports['nord_garages']:CloseGarage()
local garages = exports['nord_garages']:GetGarages()
```

Impounds

```
exports['nord_garages']:OpenImpound(identifier)
local impounds = exports['nord_garages']:GetImpounds()
exports['nord_garages']:OpenImpoundManager()
exports['nord_garages']:OpenJobImpoundManager()
```

Admin permission for management remains server-side.

Showroom helpers

```
exports['nord_garages']:IsInShowroom()
exports['nord_garages']:IsShowroomVehicle(vehicle)
exports['nord_garages']:GetShowroomVehicleData(vehicle)
```

Use these to ensure target/inventory resources ignore preview entities where appropriate.

Server Exports — Garages & Vehicles

Server Exports — Garages & Vehicles

Garage lookup/registration

```
exports['nord_garages']:GetGarage(identifier)
exports['nord_garages']:GetGarages(true)
exports['nord_garages']:GetImpounds(true)
exports['nord_garages']:GetGarageTemplates()
exports['nord_garages']:RegisterGarage(data)
```

Fleet

```
exports['nord_garages']:RegisterFleetVehicle(job, garageIdentifier, vehicleData)
```

Vehicle state

```
exports['nord_garages']:SetVehicleState(plate, state, garageIdentifier, details)
exports['nord_garages']:ImpoundVehicle(plate, garageIdentifier, reason, releaseAt, fine)
exports['nord_garages']:ReleaseVehicle(plate, garageIdentifier)
```

Staff-safe garage mutations

```
exports['nord_garages']:AddVehicleToGarage(plate, garageIdentifier, options)
exports['nord_garages']:ForceStoreVehicle(plate, garageIdentifier, options)
exports['nord_garages']:RemoveVehicleFromGarage(plate, options)
exports['nord_garages']:DetachVehicleFromGarages(plate, options)
exports['nord_garages']:StaffDeleteWorldVehicle(plate, garageIdentifier, options)
exports['nord_garages']:GetVehicleGarageState(plate)
```

All trusted mutation integrations should check the returned `ok`/`error` fields.

Server Exports — Admin, Staff & Sync

Server Exports — Admin, Staff & Sync

Admin access

```
AddAdmin(value)
RemoveAdmin(value)
IsAdmin(value)
GetRuntimeAdmins()
```

Compatibility aliases:

```
GiveGaragesAccess(value)
RemoveGaragesAccess(value)
HasGaragesAccess(value)
```

Staff snapshots

```
local rows = exports['nord_garages']:GetStaffVehicleSnapshot(limit)
```

The helper is designed for bulk authoritative reads instead of one export call per vehicle.

Framework reconciliation

```
exports['nord_garages']:ReconcileOwnedVehicleByPlate(plate, options)
exports['nord_garages']:SyncAllFrameworkVehicles(limit, force)
exports['nord_garages']:ReconcileImpounds(options)
```

Use these after trusted staff/ownership operations when another resource has changed the framework ownership table.

Ownership deletion

`DeleteOwnedVehicleRecord` is intended for explicit staff ownership deletion. Do not confuse this with `StaffDeleteWorldVehicle`, which is **world-only** and does not delete framework ownership.

Server Exports — Vehicle OS

Server Exports — Vehicle OS

Main exports include:

```
RegisterVIPProvider(name, handler)
RegisterProvider(kind, name, handler)
GetProviders()
IsPersistentParkingVIP(source)
GetVehicleIdentity(value)
ChangeVehiclePlate(value, newPlate, actor)
CompareVehicles(values)
SearchVehicles(source, query, limit)
GetVehicleHistory(value, limit)
GetVehicleSnapshots(value, limit)
RestoreVehicleSnapshot(value, snapshotId, actor)
GrantVehicleAccess(value, identifier, role, expiresAt, actor)
RevokeVehicleAccess(value, identifier, actor)
SetVehicleCollection(source, value, collection, favorite)
CreateGarageReservation(source, garage, vehicleId, minutes)
GetGarageReservations(garage)
GetGarageSlots(identifier)
UpsertGarageSlot(identifier, slot)
SetVehicleServiceData(value, patch, actor)
SetVehicleInsurance(value, patch, actor)
SetVehicleStolen(value, stolen, actor, details)
SetVehicleEVData(value, patch, actor)
GetMaintenanceStatus(value)
TowVehicleToImpound(value, garage, reason, actor, fine)
GetRecoveryCenter()
RepairVehicleState(value, action, garage, actor)
GetGarageAnalytics(days)
RestorePersistentParking()
RequestVehicleDelivery(source, plate, mode)
```

External resources should use these contracts rather than querying auxiliary Vehicle OS tables directly.

Events & Live Sync

Events & Live Sync

Nord Garages emits events so Staff, Phone and other integrations can refresh without polling the full database continuously.

Vehicle changed

```
AddEventHandler('nord_garages:server:vehicleChanged', function(payload)
    -- refresh local integration cache
end)
```

Vehicle OS event bus

```
AddEventHandler('nord_garages:server:event', function(eventName, payload)
end)

AddEventHandler('nord_garages:server:vehicleChangedV2', function(payload)
end)
```

Garage changes

Garage create/update/delete operations emit the garage change flow used by live admin/staff integrations.

Integration recommendation

Prefer event-driven cache invalidation plus occasional reconciliation. Avoid high-frequency SQL polling for every player/vehicle.

Nord Phone Integration

Nord Phone Integration

Server-side phone exports provide UI-safe normalized vehicle data.

```
local vehicles = exports['nord_garages']:GetPhoneVehicles(source)
local vehicle = exports['nord_garages']:GetPhoneVehicle(source, plate)
local garages = exports['nord_garages']:GetPhoneGarages()
```

The normalized vehicle snapshot can include plate, model/label, state, garage or impound identity, location, fuel and impound metadata.

A notification helper is also available:

```
exports['nord_garages']:NotifyPhoneVehicleChanged(ownerIdentifier, plate, action,
garageIdentifier)
```

Vehicle history/state changes invalidate the live phone cache so store, spawn, transfer, impound, release and reconciliation operations can refresh without a resource restart.

Nord Staff Integration

Nord Staff Integration

Nord Staff should use the trusted server exports instead of directly manipulating Nord Garages rows.

Give/store vehicle

After creating framework ownership, call `ReconcileOwnedVehicleByPlate` or `AddVehicleToGarage` so the new vehicle appears immediately without restart.

DV / remove from world

Use:

```
exports['nord_garages']:StaffDeleteWorldVehicle(plate, nil, {
    source = source,
    netId = netId,
})
```

This reconciles world tracking/garage state **without deleting ownership**.

Delete ownership

Use the explicit ownership deletion export only when the staff action genuinely removes the vehicle from the player's ownership database.

Global list

Use `GetStaffVehicleSnapshot(limit)` for bulk state and `SyncAllFrameworkVehicles(limit, force)` when a full framework sync is required.