Script BootStore

- Introduction
- Dépendances & Installation
- Activation de la licence (MasterKey)
- Ajout et configuration d'un véhicule
- Utilisation en jeu
- Commandes en chat
- Compatibilité Framework / Ciblage / Inventaire
- Options pour développeur & exports
- FAQ / Problèmes courants
- Config via le web

Introduction

BootStore transforme le coffre de vos véhicules en un véritable espace de rangement interactif. Les objets stockés sont visibles et manipulables directement en jeu : vos joueurs peuvent les prendre en main, les déposer au sol, les ranger dans un véhicule, ou les donner à un autre joueur.

Le mod inclut également :

- une **physique de suspension réaliste** (le véhicule s'écrase visuellement selon le poids transporté),
- un **effet de caméra dynamique** lors du portage d'objets lourds,
- une **mise en surbrillance** de l'objet visé,
- un **outil de configuration en jeu** pour positionner précisément chaque objet dans n'importe quel véhicule, sans toucher au code.

Dépendances & Installation

Dépendance	Statut	Rôle
ox_lib	Obligatoire	Notifications, TextUI, callbacks, cercles de progression
ox_target ou **qb-target**	Un des deux requis	Système de ciblage pour interagir avec les objets/coffres
Framework (ESX / QBCore / QBox)	Optionnel	Uniquement nécessaire si tu utilises le verrouillage par métier

> BootStore fonctionne aussi en **standalone**, sans framework, si tu n'as pas besoin de restrictions par métier.

Installation

Place le dossier de la ressource dans ton dossier resources. Dans ton server.cfg, assure-toi que les dépendances démarrent avant BootStore :

```
ensure ox_lib
ensure ox_target   # ou qb-target
ensure BootStore
```

- Renseigne ta MasterKey dans `config.lua` (voir chapitre suivant).
- Démarre ou redémarre la ressource.

Activation de la licence (MasterKey)

BootStore nécessite une **MasterKey** valide pour fonctionner. C'est ta clé personnelle qui identifie ton serveur auprès de MasterMods.

Comment l'obtenir

Rends-toi sur : **https://ressources.mastermods.dev/CreerMasterKey.php**

Génère ta clé, puis renseigne-la dans `config.lua` :

```
Config.MasterKey = "MM-XXXXXXXX"
```

Sans clé valide (ou avec la clé d'exemple laissée par défaut), le script affichera un avertissement en console et certaines fonctionnalités resteront désactivées.

Détail des options importantes

`Config.Framework` / **`Config.TargetSystem`** / **`Config.InventorySystem`** Laissés sur `'auto'`, BootStore détecte automatiquement ce qui tourne sur ton serveur. Si tu as plusieurs ressources compatibles installées en même temps et que la détection automatique se trompe, force la valeur correspondante.

`Config.InteractionMode` Détermine comment un joueur ouvre le coffre : viser chaque objet un par un (`aim`), ou ouvrir un menu unique listant tout le contenu (`menu`), ou laisser le choix au joueur (`both`).

Ajout et configuration d'un véhicule

Contrairement à une configuration manuelle dans un fichier, BootStore utilise un **outil de configuration en jeu**.

Étapes

1. Approche-toi d'un véhicule (moins de 5 mètres).
2. Tape `/config` en jeu.
3. Une interface s'ouvre : choisis le nom, le modèle 3D de l'objet, son type, son icône, et éventuellement une porte requise pour y accéder.
4. Valide : l'outil te fait ensuite positionner l'objet en deux temps (le fameux "double gizmo") :
   - **Phase 1** : position/rotation de l'objet **dans l'emplacement du véhicule voulu**.
   - **Phase 2** : position/rotation de l'objet **une fois tenu en main** par le joueur.
5. Une fois validé, la configuration est sauvegardée et s'applique immédiatement à tous les véhicules du même modèle.

Réutiliser une configuration existante (loadouts)

La commande `/loadout` permet de dupliquer la configuration d'objets déjà présents sur un véhicule proche, pour l'appliquer tel quel (position coffre + main incluses) à un autre modèle, sans repasser par le gizmo.

Utilisation en jeu

- **Ouvrir la porte si nécessaire** : viser le véhicule (mode `aim`) ou utiliser l'option de menu (mode `menu`), selon `Config.InteractionMode`.
- **Prendre un objet** : interagir avec l'objet ciblé (mis en surbrillance si `Config.EnableGlow` est actif).
- **Porter un objet** : l'objet apparaît en main, avec une éventuelle animation selon la configuration.
- **Poser au sol** : touche **E** par défaut (`mm_drop`), remappable dans les paramètres clavier de FiveM.
- **Ranger dans un véhicule** : interagir avec un coffre compatible en tenant l'objet.
- **Donner à un autre joueur** : interaction directe entre joueurs à proximité.
- **Effets visuels** : plus le poids porté/embarqué est élevé, plus la suspension du véhicule s'écrase et plus la caméra du joueur tangue (si ces options sont activées).

Commandes en chat

Commande	Usage
`/config`	Ouvre l'outil de configuration sur le véhicule le plus proche
`/loadout`	Duplique la configuration d'un véhicule proche vers un autre modèle
`/refreshcache`	Force le rechargement du cache des objets configurés autour du joueur
`mm_drop` (touche E)	Pose l'objet actuellement tenu en main

Compatibilité Framework / Ciblage / Inventaire

BootStore s'adapte automatiquement à :

- **Frameworks** : ESX, QBCore, QBox, ou standalone (aucun framework).
- **Ciblage** : ox_target ou qb-target.
- **Inventaire** : pont optionnel vers ox_inventory ou qb-inventory (selon `Config.InventorySystem`).

Si tu utilises plusieurs de ces ressources en parallèle sur ton serveur, force la valeur correspondante dans `config.lua` plutôt que de laisser `'auto'`, pour éviter toute ambiguïté.

Verrouillage par métier (jobs)

BootStore peut restreindre l'accès à certains objets selon le métier (job) du joueur, si un framework compatible (ESX, QBCore, QBox) est détecté ou forcé.

La vérification du job se fait **côté serveur**, directement auprès du framework, ce qui garantit qu'elle ne peut pas être contournée depuis le client du joueur.

> En mode standalone (aucun framework), aucune restriction de ce type n'est appliquée.

Options pour développeur & exports

Si tu développes une autre ressource (script d'anti-triche, mini-jeu, script métier personnalisé...), BootStore expose des événements et des exports pour s'y greffer sans modifier son code interne.

Événements diffusés (`AddEventHandler`)

```
BootStore:onPropTaken(source, propData, vehicleNetId)
BootStore:onPropDropped(source, propData, coords)
BootStore:onPropStored(source, propData, vehicleNetId)
BootStore:onPropGiven(sourceFrom, sourceTo, propData)
BootStore:onVehicleSpawnedWithProps(vehicleNetId, propsList)
BootStore:onConfigSaved(vehicleModel, propData)
```

Exports disponibles

```
exports['BootStore-INT']:IsPlayerHoldingProp(source)        -- bool
exports['BootStore-INT']:GetHeldPropData(source)            -- table | nil
exports['BootStore-INT']:GetVehiclePropsConfig(netId)        -- table (liste, vide si
rien/refusé)
exports['BootStore-INT']:ForceDropProp(source)               -- bool (true si une dépose a été
déclenchée)
```

FAQ / Problèmes courants

Le script ne démarre pas / rien ne fonctionne. Vérifie que ta MasterKey est bien renseignée dans `config.lua` et qu'elle n'est pas la clé d'exemple. Génère la tienne sur https://ressources.mastermods.dev/CreerMasterKey.php.

Les objets configurés n'apparaissent pas dans le coffre. Utilise `/refreshcache` pour forcer le rechargement du cache, ou vérifie que le véhicule a bien été configuré avec `/config`.

Le job ne restreint pas correctement l'accès à un objet. Assure-toi que `Config.Framework` correspond bien à ton framework (ou laisse `'auto'` si un seul framework tourne sur ton serveur).

Deux systèmes de ciblage sont installés et ça ne fonctionne pas comme prévu. Force `Config.TargetSystem` sur la valeur souhaitée plutôt que de laisser `'auto'`.

J'ai activé `Config.Debug` mais je ne vois rien. Les logs de debug s'affichent dans la console F8. Les commandes `/mm_testrotorder` et `/mm_liveadjust` ne sont disponibles qu'avec `Config.Debug = true`.

---

13. Support

Pour toute question, bug, ou demande d'assistance, contacte l'équipe MasterMods via Discord : **discord.mastermods.dev**

Config via le web

BootStore inclut un **tableau de bord en ligne**, accessible depuis le site MasterMods, lié directement à ta MasterKey.

Connexion

Rends-toi sur le tableau de bord BootStore et connecte-toi avec ta MasterKey (la même que celle renseignée dans `config.lua`). Chaque clé n'a accès qu'à ses propres données — impossible de voir ou modifier la configuration d'un autre serveur.

Ce que tu peux y faire

- **Visualiser et gérer tes véhicules configurés** : consulter la liste des modèles configurés via `/config`, sans avoir besoin de te reconnecter en jeu.
- **Bibliothèque de presets** : parcourir, valider ou gérer les presets d'objets créés en jeu (voir chapitre 6).
- **Bibliothèque de loadouts** : gérer les loadouts créés via `/loadout` pour les réutiliser plus facilement d'un véhicule à l'autre.
- **Suivi en direct** *(si activé)* : une carte affichant la position et le statut des objets actuellement actifs sur ton serveur.

Accès

Le tableau de bord est disponible à l'adresse fournie sur le site MasterMods, dans la section BootStore. Connexion requise via ta MasterKey.