Intégration Serveur

? Guide d'Intégration Serveur - Admin Menu v2.1

Auteur: ProtoMehka **Version**: 2.1 **Date**: 10 Janvier 2025 **Framework**: Altis Life 5.0+

? Introduction

Ce guide détaille l'installation de la **partie serveur** du menu admin v2.1. Il couvre :

- Les fonctions serveur pour les requêtes SQL
- Le système de logging des actions admin
- La configuration du serveur
- Les permissions remoteExec

Prérequis :

- Serveur Altis Life fonctionnel avec extDB3
- Accès aux fichiers `life_server`
- Base de données MySQL/MariaDB configurée

? Structure des Fichiers Serveur

```
life_server/
└── Functions/
    └── Admin/
        ├── fn_getPlayerFinances.sqf    # Récupère cash/banque d'un joueur
        └── fn_logAdminAction.sqf        # Enregistre les actions admin en BDD
```

? Installation

1?? Créer le Dossier Admin

Si le dossier n'existe pas déjà :

```
life_server/Functions/Admin/
```

2?? Copier les Fichiers

Copiez les 2 fichiers `.sqf` dans `life_server/Functions/Admin/` :

```
 fn_getPlayerFinances.sqf
 fn_logAdminAction.sqf
```

3?? Modifier config.cpp

Chemin : `life_server/config.cpp`

Localiser la classe `TON_System` :

```
class TON_System {
    tag = "TON";

    // ... autres classes existantes ...
};
```

Ajouter cette classe **DANS** `TON_System` :

```
class Admin {
    file = "\life_server\Functions\Admin";
    class getPlayerFinances {};
    class logAdminAction {};
};
```

Résultat final :

```
class TON_System {
    tag = "TON";
```

```
    // Autres classes...
    class MySQL_Database {
        // ...
    };

    class Admin {
        file = "\life_server\Functions\Admin";
        class getPlayerFinances {};
        class logAdminAction {};
    };
};
```

4?? Modifier CfgRemoteExec.hpp (Client)

Chemin : `admin_menu.Altis/CfgRemoteExec.hpp`

Ajouter dans la section `/* Server only functions */` :

```
F(TON_fnc_getPlayerFinances,SERVER)
F(TON_fnc_logAdminAction,SERVER)
```

Exemple complet :

```
/* Server only functions */
F(DB_fnc_insertRequest,SERVER)
F(DB_fnc_queryRequest,SERVER)
F(DB_fnc_updatePartial,SERVER)
F(DB_fnc_updateRequest,SERVER)
F(TON_fnc_getPlayerFinances,SERVER)
F(TON_fnc_logAdminAction,SERVER)
F(life_fnc_jailSys,SERVER)
// ... autres fonctions serveur
```

? Détail des Fonctions

fn_getPlayerFinances.sqf

Description : Récupère le cash et la banque d'un joueur depuis la base de données.

Paramètres :

- `0: STRING` - PID du joueur cible
- `1: OBJECT` - Admin qui demande les infos

Fonctionnement :

1. Reçoit une demande du client (admin)
2. Exécute : `SELECT cash, bankacc FROM players WHERE pid='...'`
3. Envoie les résultats au client via `remoteExecCall`

Exemple de requête SQL :

```
SELECT cash, bankacc FROM players WHERE pid='76561198012345678'
```

---

fn_logAdminAction.sqf

Description : Enregistre une action admin dans la base de données.

Paramètres :

- `0: STRING` - PID de l'admin
- `1: STRING` - Nom de l'admin
- `2: STRING` - Commande exécutée
- `3: STRING` - Cible (optionnel, nom du joueur ciblé)
- `4: STRING` - Détails additionnels (optionnel)

Fonctionnement :

1. Reçoit les données d'une action admin
2. Échappe les guillemets pour la sécurité SQL
3. Insère dans la table `admin_logs`

Exemple de requête SQL :

```
INSERT INTO admin_logs (pid, player_name, command, target, details)
VALUES ('76561198012345678', 'Admin Name', 'TELEPORT', NULL, 'X:3612 Y:13138')
```

Gestion des valeurs NULL :

- Si `target` ou `details` sont vides, ils sont insérés comme `NULL` et non comme chaînes vides

? Sécurité

Protection SQL Injection

Les fonctions utilisent `regexReplace` pour échapper les guillemets :

```
_adminName = _adminName regexReplace ["'", "''"];
_target = _target regexReplace ["'", "''"];
_details = _details regexReplace ["'", "''"];
_command = _command regexReplace ["'", "''"];
```

Validation des Données

Vérifications avant exécution :

- PID et commande non vides
- Objet admin valide
- Limite de résultats respectée

? Checklist d'Installation

```
□ Dossier life_server/Functions/Admin créé
□ 2 fichiers .sqf copiés
□ config.cpp modifié (classe Admin ajoutée avec 2 fonctions)
□ CfgRemoteExec.hpp modifié (2 fonctions serveur ajoutées)
□ Table admin_logs créée en BDD (voir INTEGRATION_DB.md)
□ Serveur compilé sans erreur
□ Tests effectués avec succès
```

? Nouveautés v2.1

Fonctionnalités Utilisant le Serveur

Object Manager :

- Logging des suppressions d'items : `DELETE_ITEM | Type: weapon | Class: hgun_Rook40_F`
- Utilise `TON_fnc_logAdminAction` pour tracer les suppressions

Actions Admin avec Logging Renforcé :

- Toutes les commandes admin sont loguées via `TON_fnc_logAdminAction`
- Logs stockés dans la table `admin_logs` avec timestamp automatique

Spectate Mode :

- Compatible TFAR (détection automatique)
- Logging de l'action spectate avec nom de la cible

Pas de Modification Serveur Requise

Les fonctions serveur v2.0 sont **100% compatibles** avec v2.1 :

- ✅ `fn_getPlayerFinances.sqf` - Inchangé
- ✅ `fn_logAdminAction.sqf` - Inchangé

Aucune modification côté serveur n'est nécessaire pour migrer de v2.0 vers v2.1.

Version 2.1 - 10 Janvier 2025

Revision #4
Created 7 October 2025 13:24:48 by ProtoMehka
Updated 10 October 2025 14:24:18 by ProtoMehka