Fulcrum Mission System (FuMS) Users Guide

Table of Contents

What is the Fulcrum Mission System (FuMS)

Fulcrum Mission System is a mission framework designed to run on a headless client. The framework supports easy development and modification of missions/encounters. The principle behind FuMS is to provide a framework under which a server administrator can easily manage and modify missions.

Requirements:

1. **You must be able to set up an Exile server.** I suggest you start with a vanilla mission and build up from there.
2. **You must be able to set up and run a headless client.** If you do not know what a headless client is, please Google it. If you are buying a server from a provider, you may not be able to run this system, or you may have to pay extra. I BELIEVE you can set up a headless client on a remote PC, say a second PC running at your house, but I have not tried that.
3. **You must know how to work on PBO files and have the tools installed –** This is necessary to maintain a server, so hopefully this is not news to you.

History

(https://github.com/horbin/FuMS-HC-Server/blob/master/README.md) This is where it all started.

I am the current caretaker of the system and have evolved it to its current form, version 0.5. I've included the history I could find but this system dates back to at least Arma 2. I found it broken and forgotten, but Horbin was gracious enough to hand it off to me and after a year I have completely rebuilt the broken parts, added a ton of features, and polished off a lot of the work of the guys before me had started. If I am still on the scene when Arma 4 finally launches (I am currently in lockdown due to the Covid-19 virus April 1, 2020) I will be rushing to make this work. If not, I hope someone does. This is a very elegant system that I was fortunate enough to be able to work on and add to. It's fun, it's smart, and it's HIGHLY configurable. I hope someday this thing really gets the glory it deserves.

- Steve Rodriguez (TheOneWhoKnocks)

Features

Mission Themes - A theme is a collection of missions that share a common mission loop, meaning only one of the missions in the theme are active at a time. Sort of like treasure hunt or simple Epoch /Exile style missions. FuMS supports an unlimited number of Themes, each able to have its own unique AI, loot tables, and missions.

Phased Mission System - A phased mission is a mission which branches depending on the logic provided in the mission's configuration. A mission can be set up to summon in reinforcements if a player is detected, or access a particular building. A mission can be set up to start in another region if a player completes, or fails a current mission. All of this logic is controlled by 'logic triggers', which are all customizable by mission.

Common Asset Management System (CAMS) – I have written the system used to control and tweak the mission content used here. Uses global variables to load your custom content and integrate into all missions. For example, the global variable *CAMS_Packs_ALL* is populated by the CART system with all of the content you run on your server (CUP, RHS, Unsung, etc) and then is used in the mission system to pick a backpack for AI. This allows you to add content without modifying the missions and will work with all of my other systems. (DAPE, DyCE)

MissionFX System – Another new module that can scan a mission that you create looking for certain key items. If found, it will replace that item with triggers for the various scripts. It is mainly used to launch ALIAS anomalies which are just proximity triggers for scripts. As a mission scripting tool, it lets you run any custom script as part of the mission system, so any future content can easily be added. See the MissionFX mission theme for a running example of the system.

Support

Unfortunately, the Exile forum has been shut down and the main thread for support is gone. (It actually can be found if you look for the WayBack machine. I may go harvest that thread someday…)

For now, I am trying to provide support via the Discord app. You can find me here: https://discord.gg/B2dEsgR. My work can also be seen on YouTube and the main code links are: https://github.com/ExiledHeisenberg/FuMS-HC-Exile

Installing the Fulcrum Mission System

Installation – Headless Client

NOTE: I recommend that you install this in a clean environment to try it out at first. Installing it into an existing server can be done, but you really need to know how your server works to be successful.

Configuring the Headless Client: A HEADLESS CLIENT IS REQUIRED FOR FuMS. *You may see some server-side only code, I am working on it. It doesn't work properly in this version.*

These instructions are for **installing a headless client on the same local PC as the server**. I assume if you are running a complex setup like a headless client on a second machine, you know what you are doing. You'll need to make some adjustments for your specific setup.

I am not technical support for Bohemia so if you can't figure out how to make the headless client work, this system will not work. If you cannot complete this first part, stop. This a complex system.

1. Review this video. (Hopefully it stays up) – https://www.youtube.com/watch?v=p9h5fOR87x4&t=232s. This is really the best way to do this, he does a great job of walking you through it. The following steps are how I have configured my system, but I learned from that video.
2. Locate the arma3server.exe file in the main server folder and note the directory (ex. C:\Arma\Server\)
3. Place a copy of your @Exile directory (Client folder) into the main server folder. This is from step 2. The Exile client can be copied from your client or downloaded from the main site

4. Create a batch file to launch your server. You should have one already, but this will allow you to launch both the client and the server, and it will restart your server for you if you have it on a scheduled shutdown. NOTE: I suggest you restart the server every 4 hours as this is a memory hog of a system)

Example batch file to launch HC and server (Copy and paste the code below to see it correctly since there is one long command in there, see the REM statements). Mine is called "**StartServer.bat**"

```
@echo off
color 0a
title Server Monitor
start "HC_HAL" arma3server_x64 -client 127.0.0.1 -mod=@Exile -profiles="C:\DEV\Logs\HC_HAL"

:Serverstart
echo Launching Server

c:
cd "C:\DEV\ExileServer"
echo Server Monitor... Active !

REM The next line is one long command, not three different lines
start "Arma3" /min /wait arma3server.exe -mod=@Exile -servermod=@ExileServer;@FuMS -
config=C:\Arma\Server\@ExileServer\config.cfg -port=2302 -
cfg=C:\Arma\Server\@ExileServer\basic.cfg -filepatching -autoinit
REM This is the end of the command.

ping 127.0.0.1 -n 15 >NUL
echo Server Shutdown ... Restarting!
ping 127.0.0.1 -n 5 >NUL
cls
goto Serverstart
```

Modifying the mission.sqm file

See \Docs\Install Examples\mission.sqm to see a working file for Altis properly configured for a headless client.

5. Modify your mpmissions\Exile.Altis.pbo mission.sqm file
    a. Locate this section

```
class Groups
{
    items = 1;
    class Item 0
    {
```

6. Change `items = 1;` to `items = 2;`
7. Add the following AFTER the closing `};` for the 'class Item 0' definition

```
class Item1
{
    side="LOGIC";
    class Vehicles
    {
            items=1;
            class Item0
            {
                    position[]={10720.502,12.714643,11356.243};
                    id=100;
                    side="LOGIC";
```

```
                vehicle="HeadlessClient_F";
                player="PLAY CDG";
                init="this enableSimulation false; this allowDamage false";
                leader=1;
                skill=0.60000002;
                text="HC_HAL";
            };
    };
};
```

8. Save the mission.sqm file

Modifying your server's config.cfg by adding the following lines:

9. Add the following lines after the [motd] line (Around line 31):

```
localClient[] = {127.0.0.1};
headlessClients[] = {"127.0.0.1"};
battleyeLicense=1;
```

Adding client side content

I have added support for ALIAS anomalies that are pretty cool. In order to use the MissionFX system and accompanying mission, you must follow this step

10. Copy the contents of the directory ***exile.mission*** to your mission file (Example Exile.Altis).
11. Add the lines from the file **Examples\description.txt** to the end of your own **description.txt**.
    a. CHECK YOUR FILE BEFORE YOU DO THIS AND SEE IF YOU ALREADY HAVE A **class CfgSounds** SECTION. If you do, you should merge the two instead
    b. THE SAME IS TRUE OF THE **class RscTitles** section. If you have one, merge the content from the example files
12. Add the lines from the file **Examples\initServer.sqf** to the end of your own **initServer.sqf**.
13. Do not repack your mission PBO file just yet. You will be adding more in the next step, but that completes the headless client install.

Your server is now ready to accept an HC!

Installation - FuMS

1. Edit the 'initPlayerLocal.sqf' in your mpmissions/Exile.Altis.pbo and add this near the top.

   ```
   []execVM "HC\init.sqf";
   ```

   Note: ENSURE it is NOT within any conditional (if/then) statements.

2. Copy the @FuMS folder to your base server directory (C:\Arma\Server)

   Do this on your SERVER. No need to put this folder on your HC!

3. Modify your server start parameters to include -servermods=@FuMS;
   a. Obviously you add this to your existing launch statement.
   b. NOTE: This is already completed in the example script
4. Repack your mpmissions pbo!

You should now be able to launch the entire system by running that batch file that you created in step 4

How can I confirm that it's running?

The system logs information to two locations: 1. Your server.rpt file and 2. The RPT file for the headless client. You need to look in both locations to verify the system is up and running. Verify that you can open them both in a text editor (I use Notepad++) to proceed.

The server.rpt file is generally located in the ***C:\Users\(username)\AppData\Local\Arma 3*** folder

The client.rpt file is determined by the batch file. In the example, the location is determined in this part of the launch script: ***-profiles="C:\DEV\Logs\HC_HAL"***

Server side

You will see various reports showing the status of various parts. Here is the main sequence you will see as the server starts and what they mean: (There will be a lot of other information in these logs, this is just the FuMS stuff that tells you things are working.

```
9:59:28 "[CAMS:0.95] CAMSconfig.sqf: Launching..."

9:59:28 "[CAMS:0.95] init.sqf: Launching..."

9:59:30 "[CAMS] loadCart.sqf cartname : vanilla | cartdata:

9:59:30 "[CAMS] FrSB_fn_loadCart.sqf cartname : vanilla SUCCESS compiled in 0 seconds"

9:59:30 "[CAMS] System | Vanilla assets loaded"

9:59:30 "[CAMS] System | Vanilla ImmersionFX loaded"

… Lots of other info about the CART system will be displayed here ae the content is loaded into
the proper global variables …

9:59:37 "[CAMS] CAMS_isStable:true"

9:59:37 "<FuMS:vExile0.50> LoadCommonData: Preparing FuMS common data."
```

At this point the CAMS system in online and now FuMS is starting. If any errors have occurred so far, something is wrong with the asset system and you have to troubleshoot.

```
9:59:38 "<FuMS> BuildThemeMissionList:MissionFX: List from recursion: [""BanditCamp-
Flamer"",""BanditCamp-Sparky"",""BanditCamp-Farty"",""BanditCamp-
Screamer"",""SuicideBomberCamp"",""BanditCamp-Strigoi""]"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script MissionFXStart for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script MissionFXEnd for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnMinefield for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnFlamer for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnFarty for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnFartyPools for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnScreamer for theme MissionFX"
```

```
 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnStrigoi for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnSparky for theme MissionFX"

 9:59:38 "<FuMS> LoadCommonData: Preprocessing custom script SpawnCrazy for theme MissionFX"

 9:59:38 "<FuMS> BuildThemeMissionList:HeloPatrols: List from recursion:
[""HeloPatrolEast"",""HeloPatrolEast""]"

 9:59:38 "<FuMS> BuildThemeMissionList:SEM: List from recursion:
[""BanditCamp"",""Help_Helo"",""SpawnGuards"",""HeloCrash"",""EvacTownVeh"",""SpawnScavengers"","
"StrangeDevice"",""PlaneCrash"",""VehicleCrash""]"
```

There will be a lot more entries like this while the system checks its files. There is a very comprehensive file checking system so hopefully most errors will be caught here if you modify the system in any way. Again, if this is showing any errors, you should fix them now as it will affect the system if anything has been modified incorrectly.

```
9:59:42 "<FuMS:vExile0.50> Init.sqf:  Server side FuMS initialized and operational."
```

Once your system shows this, the FuMS server side components are running and stable. If your headless client is running, it should auto connect any time now. When it does, you'll see this. This is the first test to make sure your headless client works.

```
9:59:43 "ExileServer - Player HC_HAL (UID HC3648) connected!"
9:59:45 "<FuMS> HeartMonitor: Server-HC Heart Monitor Slot #4 initialized for HC_HAL"
9:59:45 "##FuMsnInit: Global variables being handed off too HC HC_HAL id:4"
9:59:48 "<FuMS> HeartMonitor: Waiting for HC:HC_HAL initialization to finalize with signature
FuMS_HC_isAlive4"
10:00:02 "<FuMS> HeartMonitor: Waiting for HC:HC_HAL initialization to finalize with signature
FuMS_HC_isAlive4"
10:00:02 "<FuMS> InitHeadlessClient:  Starting transfer of XX Custom Scripts to HC <4:HC_HAL>"
10:00:07 "##FuMsnInit: Starting transfer of  104 Scripts to Headless Client <4:HC_HAL>."
```

There will be a lot of transferring of data that occurs, and the server will wait to hear back from the headless client that it is ready to go. You will start to see this:

```
10:02:23 "<FuMS> HeartMonitor: Waiting for HC:HC_HAL initialization to finalize with signature
FuMS_HC_isAlive4"
10:02:25 "<FuMS> HeartMonitor: Waiting for HC:HC_HAL initialization to finalize with signature
FuMS_HC_isAlive4"
10:02:27 "<FuMS> HeartMonitor: Waiting for HC:HC_HAL initialization to finalize with signature
FuMS_HC_isAlive4"
10:02:36 "##FuMS:Menus:init.sqf
AdminUpdateData:[1,""AdminThemeOn"",[false,true,true,true,true,true,true,true,true,true,true,true
,true,true,true,any,true]]"
10:02:41 "[DAPE] Waiting... HC's : [HC_HAL] | Players = []"
```

That last line is actually from my DAPE system and monitors what headless clients is sees. If you happen to be using this addon as well, that will also help you verify that your headless client is connected and working.

Your server is now ready and waiting for players to connect. You may see maintenance logs from the system, but unless you have some missions set to spawn without any players, the system will wait until the minimum number of players connects. (Default is 1)

NOTE: When the first player connects, it's like starting an engine. It takes a few minutes to settle down, so I suggest you connect yourself to the server as a player to kick off the mission engine. Otherwise your players will see a huge lag hit right about 5 minutes in for about 20 seconds.

Code is being updated to avoid this in future versions.

When a player connects

Once a player connects, the system first checks to see if it is an admin, but this will indicate that the client logic is loaded properly and each client is kicking off this process. As you can see, player "Tim" connected, was compared to the master admin list, and not found, so no admin for him!

```
11:22:59 "##InitMenu: Checking for Admin Access for R Alpha 1-2:1 (Tim) REMOTE"
11:22:59 "<FuMS> GetUserData   Player:R Alpha 1-2:1 (Tim) REMOTE with Guid:xxxxxxxxxxxxxx"
11:22:59 "<FuMS> GetUserData examining:[""TheOneWhoKnocks"",""xxxxxxxxxx"",1,[""<ALL>""]]"
11:22:59 "<FuMS:vExile0.50> HasAdminAceess: []"
11:22:59 "<FuMS:vExile0.50> HasAdminAccess: R Alpha 1-2:1 (Tim) REMOTE is not in AdminData.sqf.
Found R Alpha 1-2:1 (Tim) REMOTE"
11:22:59 "##InitMenu: Admin Access for R Alpha 1-2:1 (Tim) REMOTE | false"
11:22:59 "<FuMS> GetUserData   Player:R Alpha 1-2:1 (Tim) REMOTE with Guid:xxxxxxxxxxxxxx"
11:22:59 "<FuMS> GetUserData examining:[""TheOneWhoKnocks"",""xxxxxxxxx"",1,[""<ALL>""]]"
11:22:59 "<FuMS:vExile0.50> HasAdminAceess: []"
11:22:59 "<FuMS:vExile0.50> HasAdminAccess: R Alpha 1-2:1 (Tim) REMOTE is not in AdminData.sqf.
Found R Alpha 1-2:1 (Tim) REMOTE"
```

Client side

In the client.rpt file, you will see the following items as the system comes online. NOTE: You will also see some log items show up in the players rpt file on their local machine. These can be useful for troubleshooting, if you get a real weird problem check those as well.

```
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "<FuMS> Headless Client HC_HAL connected with profile name HC_HAL"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "################################################"
13:56:48 "<FuMS> HC_Init: HC_HAL waiting on Server's Init Token"
```

The system will wait until it gets the signal from the headless client that it is ready to start the data transfer.

```
13:56:48 "<FuMS> HC_Init: HC_HAL waiting on Server's Init Token"
13:56:50 "<FuMS> HC_Init:   GetHCIndex:HC_HAL  FuMS_HC_SlotNumber:-1"
13:57:00 "<FuMS> HC_Init HC_HAL connected using owner id:4"
13:57:00 "<FuMS> HC_INIT: Receiving Common Data from Server."
```

```
13:58:06 "<FuMS> HC_INIT: Receiving Common Data from Server."
13:59:16 "<FuMS> HC_INIT: Script List size = 104"
13:59:16 "<FuMS> HC_Init: Compiling Custom Script MissionFXStart"
13:59:16 "<FuMS> HC_Init: Compiling Custom Script MissionFXEnd"
```

The data transfer is now starting and all local code is being moved to the headless client

```
13:59:32 "##Init_HC_HAL_Variables: Slot 4 initialized."
13:59:32 "<FuMS> FuMsnInit:  Heart Beat Started for HC_HAL. using slot# 4."
```

Data transfer is complete and the headless client is now operating. It will start the varius timers and you will start to see missions spawning.

CONGATULATIONS! You are now running the Fulcrum Mission System!

This is a complex system, but well worth the work. I have tried to make it fun and will continue to tweak the missions, ad functions, etc. but if you take the time to learn it, this is a highly configurable system that can run almost any scenario you can dream up.

Configuration

FuMS Server Configuration

Configuring FuMS involves understanding the organization and settings within 6 basic file types:

- Server Level
  - **BaseServer.sqf**
  - These are settings common to all of components FuMS on your server
- Loot Configuration
  - **GlobalLootData.sqf**
  - Establishes loot types common to all missions. These loot arrays are available to any mission in any theme
- AI Configuration
  - **GlobalSoldierData.sqf**
  - Establishes common soldier types for use by missions. Think of these as models of AI (ex. Sniper, Soldier, Pilot)
- Themes
  - **ThemeData.sqf**
  - All FuMS missions belong to a theme
  - A theme is a common grouping of missions that typically share some common objectives, regions, or styles
  - Each theme has its own folder in FuMS
  - Controls settings common to all missions under the same theme
  - *There is one ThemeData.sqf for each theme*
  - (OPTIONAL) You can have a **SoldierData.sqf** and **LootData.sqf** file in each theme to override the **GlobalSolderData.sqf** and **GlobalLootData.sqf** files and provide theme specific AI and loot per theme folder
  - (OPTIONAL) You can have custom scripts run before, during and after the mission based on mission spawn, mission completion (or failure), and in mission trigger
- Mission Files
  - One mission file per mission
  - Mission files are located in specific theme folders
  - Themes do not share missions
- CAMS System
  - **CAMSConfig.sqf**
  - CARTS folder
    - Contains asset lists and custom content integration files

CAMS System

The CAMS (Common Asset Management System) allows the customization of all of my addons by either modifying the included files or creating your own. The basic idea is this system creates a set of global variables that store information about Arma content packs. These variables are then used in the mission systems to defines things like the uniforms that your AI will wear, the weapons they use, even the model that is used to spawn AI (This determines the language they speak).

It does this by using a CART folder (Common Asset Resource Template). There is one folder per addon and each folder contains two files, Assets.sqf and ImmersionFX.SQF.

Assets.sqf - These files are located in the **addons\DEMS\CAMS\carts\(content name)** directories and pre-load the global variables used throughout my add-ons. You can use the existing files or the included template files to add your own CART file. These generally group all of the content of a particular type into a single global variable, for example ***CAMS_U_Civ*** is a global variable that contains all of the civilian uniforms

ImmersionFX.sqf - These files are located in the **addons\DEMS\CAMS\carts\(content name)** directories and these a more functional in that you can choose specific classnames for a purpose as opposed to all of the possible content available. For example, ***ImFX_Heli_Troops*** is a combination of the global variables defined in Assets.sqf ***CAMS_Heli_Unarmed_E, CAMS_Heli_Unarmed_I, CAMS_Heli_Unarmed_W***. You can use this to make custom groupings of classnames and use them the same as the other global variables, so in a mission you can use ***ImFX_Heli_Troops*** as the vehicle type and it will pull from this array.

CAMSConfig.sqf – This file determines what content is loaded by the system. There are three variables:

```
CAMS_useVanilla     = true;             <=- Load Vanilla content?
CAMS_useExile       = true;             <=- Load Exile content?
CAMS_cartList =     [                   <=- List of folder names to scan
                        "helicopters",
                        "marksmen",
                        "jets",
                        "apex",
                        //"unsung"    <=- Commented out by default

                        //"cupw",     <=- Commented out by default
                        //"cupv",     <=- Commented out by default
                        //"cupu"      <=- Commented out by default
                    ];
```

Layering content on top of content

The entire system works by layering the data on top of previous layers. By default, the system loads the vanilla content, Apex, Helicopters, Jets, Marksmen, and Exile. It starts by creating the global variables using the Vanilla CART, then each cart after that adds its content to the same global variable. This allows you to use the same variable throughout your system without changing it, but as you add content to your server you can easily integrate it to existing missions. See here:

CAMS_Heli_Unarmed_E =
(Vanilla) + (Jets) + (Helicopters) + (Apex) + (Marksmen) + (Exile)

Overriding Vanilla Content (Total Conversion)

You can alternatively override the default vanilla content using the CART system and changing one value. In each Assets.sqf file including the one in the template folder, the global variable is defined like this: (This is just one small part of the file)

```
[
    "CAMS_U_Soldier_W",1,false,
    [
        "U_B_CombatUniform_mcam, "U_B_CombatUniform_mcam_tshirt"
    ]
],
```

This is telling the system that you are updating the `"CAMS_U_Soldier_W"` global variable, it is a specific list (`1`) as opposed to a global grouping (see next section), you are NOT overwriting the existing value (`false`). The classnames you are adding are `"U_B_CombatUniform_mcam, "U_B_CombatUniform_mcam_tshirt"`

To override the variable instead of adding to it, change the option from false to true

```
[
    "CAMS_U_Soldier_W",1,true,
    [
        "New Uniform Classname 1", "New Uniform Classname 2"
    ]
],
```

This allows you to override any other content from being used and do a total conversion. In the Unsung CART folder for example, it removes all of the content loaded before it and makes it so only the Unsung content is used throughout the system. Unsung weapons, uniforms, gear, etc.

Map Immersion System

A new system being developed is the Map Immersion System. This allows you to specify a building to swap on any map with content from your custom content. The current work is being done with the Unsung content so please review that CART folder and look at the ImmersionFX.sqf file. You will see a section titled "Map Immersion System" near the end that looks like this:

```
_swapArray =
[
        ["Land_wpp_Turbine_V1_F","Land_WX_Windmill",180],
        ["Land_Chapel_Small_V2_F","LAND_CSJ_pagoda2",90]
        ….
];
```

The format is

```
["Original classname","Replacement classname", rotation needed to match original building]
```

You can use this system to swap out any item I suppose, I am still testing, but so far it works really well. As content is released that includes new buildings, you can easily build a CART folder by copying the TEMPLATE to a new folder and edit the ImmersionFX.sqf file

Missions

Default FuMS distribution has some theme sets starting with no players. All others are defaulted to start when at least one player is logged in.

Go read the \Docs\ Folder to learn how to customize the themes and missions to meet your server's needs!

AI

Custom configurations

Below are some examples and optional modules that are being developed. This is just a place to dump good ideas as I structure this document

Example of batch file to launch server and HC with 8 processors and the CUP content

```
@echo off
color 0a
title Server Monitor
start /affinity 3C "HC_HAL" arma3server_x64 -client 127.0.0.1 -
mod=@CBA_A3;@Exile;@CUP_Units;@CUP_Vehicles;@CUP_Weapons -profiles="C:\DEV\Logs\HC_HAL"
REM start /affinity 3C "HC_HAL" arma3server_x64 -client 127.0.0.1 -
mod=@CBA_A3;@Exile;@CUP_Units;@CUP_Vehicles;@CUP_Weapons -profiles="C:\DEV\Logs\HC_HAL"
rem start /affinity 3C "HC_HAL" arma3server_x64 -client 127.0.0.1 -mod=@Exile -
profiles="C:\DEV\Logs\HC_HAL"

:Serverstart
echo Launching Server

c:
cd "C:\DEV\ExileServer"
echo Server Monitor... Active !
start /affinity C0 "Arma3" /min /wait arma3server.exe -
mod=@CBA_A3;@Exile;@CUP_Units;@CUP_Vehicles;@CUP_Weapons -
servermod=@CBA_A3;@ExileServer;@infiSTAR_Exile;@FuMSDEV;@CUP_Units;@CUP_Vehicles;@CUP_Weapons;@sl
z -config=C:\DEV\ExileServer\@ExileServer\config.cfg -port=2302 -
cfg=C:\DEV\ExileServer\@ExileServer\basic.cfg -filepatching -autoinit
REM start /affinity C0 "Arma3" /min /wait arma3server.exe -mod=@CBA_A3;@Exile -
servermod=@CBA_A3;@ExileServer;@infiSTAR_Exile;@FuMSDEV;@CUP_Units;@CUP_Vehicles;@CUP_Weapons;@sl
z -config=C:\DEV\ExileServer\@ExileServer\config.cfg -port=2302 -
cfg=C:\DEV\ExileServer\@ExileServer\basic.cfg -filepatching -autoinit
rem start /affinity C0 "Arma3" /min /wait arma3server.exe -mod=@Exile -
servermod=@ExileServer;@infiSTAR_Exile;@FuMSDEV;@slz -
config=C:\DEV\ExileServer\@ExileServer\config.cfg -port=2302 -
cfg=C:\DEV\ExileServer\@ExileServer\basic.cfg -filepatching -autoinit

ping 127.0.0.1 -n 15 >NUL
echo Server Shutdown ... Restarting!
ping 127.0.0.1 -n 5 >NUL
cls
goto Serverstart
```

Jurassic Raptor Addon Support

This is leftover from the original system. I have not worked on this section, but I am keeping it here in case I decide to ever get it working again. If you want to figure out how to make this work and send the info to me, I'll credit you of course. I am still re-writing this, again it is left over from the original version.

http://makearmanotwar.com/entry/ec2EDrOCkM#.VT0zFfnF9EK

1. Download the addon.

2. Place the '@Jurassic Arma - Raptor Pack' folder in the base folder of your server. At the same folder level as the @Exile folder.
3. Place this folder in the same location on your HC, IF your HC does not share the same source folder as your server.
4. Add @Jurrasic Arma - Raptor Pack to the -mod option of your server command line
5. Add this to the -mod option for your HC.
6. Add this to the -mod option for your client.
7. Ensure your players download the mod, and add the proper @Jurassic.... to their start parameters.
8. Enable the "Jurassic" theme in BaseServer.sqf.
9. Learn to generate a bi-key for the mod and add it.
10. OR
11. edit your config.cfg file and set 'verifySignatures = 0;'
12. repack and play!
13. Note: Raptors use the same AI patrol logics as regular FuMS soldiers, so BoxPatrol, building searching, PatrolRoute, etc logics will
14. all work with the "RaptorM", and "RaptorF" AI types.
15. Note: Feel free to add Raptors to your own encounters!, see the Jurassic theme for examples.