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.

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

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)

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"",""xxxxxxxxxxx"",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 various 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, add 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:

- CAMS System
    - **CAMSConfig.sqf**
    - CARTS folder
        - Contains asset lists and custom content integration files
- 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

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"`. As each asset files is processed, it will add its own content to each matching global variable, resulting in a single array of all content.

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 overwrites 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. all are used by calling the same global variables as before.

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 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 for each entry 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 folder to a new folder and edit the ImmersionFX.sqf file

Base Server Configuration

File: FuMS\Themes\BaseServer.sqf

This file contains configuration information that impacts core FuMS functionality and details that apply across all themes and missions. This file also controls what themes should be loaded when FuMS initializes. The code is heavily commented as well. This file is basically one big array spaced apart to make it easy to configure, sp pay attention to commas and semi-colons.

Core Configuration

The first section determines FuMS and Admin tool behavior.

Server Config

Parameters

FuMS ON SERVER: BOOL: false (DO NOT CHANGE – This is not currently working)

- true = enables FuMS to run missions on the main server. Enabling this option can result in serious performance degradation on the server. This option is primarily provided to allow people who do not have an HC to run and test FuMS before going through learning how to set up the Headless Client
- false = FuMS will not spawn missions on the server. All missions and AI are controlled by the Headless Clients

DATABASE CONFIG: STRING: "testpath" (DO NOT CHANGE – This is not currently working)

- Place holder for future MySQL/REDIS data persistence and configuration support

ADMIN CONTROLS: BOOL: true

- true = enables FuMS administrative menu system. This menu system provides basic controls to allow designated admins to start/stop missions
- false = disables FuMS admin menu system
- See “Admin Controls” section for further details on how to use these tools

MIN FRAMERATE: SCALAR : 20

- FuMS will monitor the server FPS. When the FPS falls below the value specified, FuMS will suspend mission creation and script communication with the Headless Clients until the FPS returns above the set value.
- 90% of the server load from FuMS occurs during HC connection, due to the data transfer between server and HC. Because of this fact, the frame rate restrictions no longer apply to an 'operating' HC.

Notes

FuMS reads the map to gather information such as the center, size, and any exclusion areas where you don't want missions to spawn, modify this file: **\FuMS\HC\Util\GetWorldInfo.sqf**

- FuMS should work on any Arma3 map that has a valid map definitions file (ex. is a working map). See the above file for details.
- If you are adding a new map, OR need to add exclusion areas, such as spawn points, you may need to modify or add to data in this file.
- All stock Themes that come with FuMS are designed to operate on any map with no additional setup required except where noted in the name of the folder. Be sure to understand if a Theme is map specific
- Of course, FuMS is an Admin's mission tool, so it is HIGHLY recommended you tailor each theme to your server's loot/player experience level. By default FuMS loot is fairly generic and the AI are mediocre marksman.

THE NEXT TWO SECTIONS ARE UNKNOWN. AS I HAVE INHERITED THIS PROJECT, I'M NOT 100% SURE OF EVERY LITTLE PIECE OF CODE. I'LL DOCUMENT EVENTUALLY. IT LOOKS LIKE THIS:

```
[       // Exclusion Areas
        // See \FuMS\HC\Util\GetWorldInfo.sqf if you need to make changes
],
[       // Default Areas
        // See \FuMS\HC\Util\GetWorldInfo.sqf if you need to make changes
],
```

Active Themes

List all the themes that FuMS should load and control. When more than one theme is being run by FuMS, ensure each is separated by a comma. Also ensure the last theme in the list contains NO trailing comma. Commenting out an item will DISABLE the theme (ie Turn it off)

Parameters

["Test", -1],

- "Theme Name" : STRING : "Test"
    - Theme name is NOT case sensitive (it will be converted to caps)
    - Should be a single word or continuous string with no special characters
    - Must match a folder in \FuMS\Themes\
- HC Control: SCALAR : -1
    - Value controls where FuMS will control the theme
    - -1 = Theme will run on server (if server operation enabled), and ALL HC's that connect to the server
    - 0 = Theme will run on server (if server operation enabled)
    - 1,2,3 ... = Theme will only run on the 1st, 2nd, 3rd... HC to connect to the server

Notes

- When a theme starts, it instantiates its own mission control loop that is independent of all other themes.

- Themes that start on the server and/or HC's each have their own control loop and operate independently.
- A theme CAN be listed multiple times, each theme running independently.
- EX: ["SEM",-1], ["SEM",-1],["SEM",-1] will run 3 instances of the SEM theme, resulting in 3 active missions, 1 under each theme, running at all times.

Event and AI Radio message behavior section

This option is not currently functional but being worked on. This section allows you to turn Radio chatter support ON/OFF for all of FUMS, as well as some other global settings

Parameters

ENABLE RADIOCHATTER : BOOL : false

- true = turns on chatter system, allowing AI radio conversations to be generated
- false = turns off all radio chatter support options (to include reinforcements (see ***Triggers*** section)

ENABLE RADIOAUDIO: BOOL: false

- true = turns on local audio sound effects
- false = turns off sound effects from AI radio chatter

RADIO REQUIRED: BOOL: true

- true = player must have a radio on his belt to see/hear radio chatter messages
- false = players will hear/see messages at all times (when within range)

RADIOFOLLOW THEME: BOOL: true

- true = each theme's ThemeData.sqf will be referenced for the theme's radio behavior
- false = all FuMS themes will follow setting specified in this section

ENABLE AI CHATTER: BOOL: true

- true = turns on random radio chatter between AI and their base of operations
- false = turns off random chatter, death messages will still occur.

RADIO RANGE: SCALAR: 800

- Sets the range, in meters, that AI radio chatter can be heard by default
- This distance is measured from a player's location to the mission centers
- Chatter from each theme's BASE Operations always has unlimited range
- Radio Range in Theme file can override this setting

Notes:

This system is not supported yet. There is code there and I am working on it, but I am not supporting this at this time.

Soldier Defaults

This section provides basic load-out and logic behavior, as well as control over AI specific equipment.

Parameters

RIFLE MAGS: SCALAR: 6

- AI created will have 6 rifle magazines appropriate to their weapon upon AI creation by default

PISTOL MAGS: SCALAR: 4

- AI created will have 4 pistol magazines appropriate to their weapon upon AI creation by default

VCOM DRIVING: BOOL: true (THIS IS NOT CURRENTLY SUPPORTED OR TESTED)

- true = enables Genesis92x's VCOM_Driving V1.01 for AI in vehicles
- false = AI will utilize default Arma3 driving logic

SKILLS ARRAY: ARRAY:

- Values specified here **OVERRIDE** any value set in any other soldier configuration file
- Setting a value to zero, permits settings from **GlobalSoliderData.sqf** to take effect
- This entire array is currently set to 0 to allow the GlobalSoliderData.sqf to take effect but if you decide to use this section, it is heavily commented and matches Arma's skill layout

```
[
    0, // aimingAccuracy : target lead, bullet drop, recoil
    0,  // aimingShake : how steady AI can hold a weapon
    0,  // aimingSpeed : how quick AI can rotate and stabilize its aim and shoot.
```

```
        0,  // spotDistance : affects ability to spot visually and audibly and the accuracy of
the information
        0,  // spotTime : affects how quick AI reacts to death, damage or observing an enemy.
        0,  // courage : affects unit's subordinates morale
        0,  // reloadSpeed :affects delay between weapon switching and reloading
        0,  // commanding : how quickly recognized targets are shared with the AI's group.
        0   // general             : Overall multiplier for all other settings
    ],
```

SOLDIER ONLY GEAR: ARRAY

Items listed in these lists will be deleted upon AI death, preventing players from having access to the equipment.

- Default values:

```
    [
        [], //Uniforms
        [], // Vests
        [], // Backpacks.
        [], // Helmets
        ["launch_RPG32_F","launch_I_Titan_F"], // Weapons
        ["RPG32_HE_F","RPG32_F","Titan_AA"], // Magazines
        [] // Items
    ],
```

- The above defaults ensure the stock AT and AA weapons used by the AI are not available to players, but some missions do not use the stock gear. You'll need to fine tune this to your system.
- Other custom equipment can be placed here to permit admins to tailor the look and capability of the AI on his server
- This is a DELETION list. Equipment still needs to be placed on the AI in the appropriate SoldierData.sqf file
- This array only supports individual "String" names. Global array objects defined within FuMS and CAMS are not recognized by this array, so items must be listed individually

UAV AND UGV's: ARRAY

- List all UAV and UGV class names that FuMS should recognize as AI controlled.
- UV 'side' is determined by the prefix on the class name (I=Resistance, O=East, B=West).
- UV 'side' is NOT determined by the driver 'side' defined in the mission file.

Notes:

This may be adjusted in future versions to compensate for ill-named UAV/UGV

Loot Defaults

Provides global configuration of loot rewards and sets restrictions on player vehicles

Parameters

DELETE LOOT DELAY : SCALAR: 20 (minutes)

- Creates a timer that starts after mission cleanup occurs.
- When timer expires loot boxes created by the theme's last expired mission will be deleted.

SMOKE BOX OPTIONS: ARRAY

These settings allow for the configuration of 'smoke' signals to aid players in locating loot boxes

Default settings:

```
[
    true, // create smoke
    100, // start smoke when a player gets this distance from the box
    ["Red", "Blue”], // color of smoke
    1 // duration of smoke in MINUTES
]
```

- Setting the distance to 0 will make the distance unlimited
- Smoke can also be defined as a single color or a group array.
    - [“Red”]
    - ["Red","White","Blue"]
    - ["Green","White"]

VEHICLE PERSISTANCE: BOOL: false (THIS IS NOT CURRENTLY WORKING, LEAVE OFF)

- true = vehicles occupied by players are sellable and will persist through server reset
- false = vehicles NOT sellable and will vanish at server reset
- FuMS provides a message to all players entering a FuMS created vehicle of its potential temporary status. This message goes away for vehicles that remain through a server reset.

CRATE OPTIONS: ARRAY:

```
[B_supplyCrate_F","O_supplyCrate_F","I_supplyCrate_F","CargoNet_01_box_F"]
```

- Within LootData.sqf and GlobalLootData.sqf the type of box created for loot is defined by each loot type. However, you can specify “RANDOM” as loot and it will just pick the contents from one of the loot types.
- When the key word "RANDOM" is used in these files, the above array is referenced for the selection of loot container.

PROHIBITED VEHICLES: ARRAY:

```
["I_UGV_01_rcws_F","B_G_Offroad_01_armed_F","O_LSV_02_unarmed_F"]
```

- Vehicles listed in this array will not be usable by players. Allows you prevent players from taking mission support vehicles
- Players will receive a notification of this fact when attempting to enter a vehicle on this list
- Vehicles on this list will function normally for AI

VEHICLE AMMO: BOOL : true

- true = vehicle ammo set to zero when a player FIRST takes possession of a vehicle from AI
- false = vehicle ammo remains intact when players capture a vehicle from AI
- This flag does NOT impact ammo in player owned vehicles. It only applies to vehicles spawned by FuMS before they become player controlled

NEXT SECTION

File:

    FuMS\Themes\GlobalLootData.sqf

Example:

```
["Name", "Box Type", "RANDOM"],

  [// All weapons and quantity

     [FuMS_LightMGs, 2],

     [FuMS_AssaultRifles, 2]

  ],

  [// All magazines and quantity
```

```
    [FuMS_AmmoForEach, 4]    // 4 clips for each weapon Line defined above.

// This will provide 4 clips for the LightMG's and 4 for the rifles for a total of 8 clips (not 16 clips)

],

[// All items and quantity

    [FuMS_B_Navigation, 2],

    [FuMS_Crafting_Tools, 3],

    [FuMS_W_Opticslv1, 2],

    [FuMS_Food_Raw, 2],

    [FuMS_Drink, 2],

    ["G_Diving",2]

],

[// All backpacks and quantity

  [FuMS_Packs_ALL, 1]

]
```

Description:

This file contains configuration information that defines default FuMS loot data types. Each data type is defined by a name, container type, and list of loot to be placed in the container. Additional loot data types can be defined by Admins, simply append them to the end of the file.

Sections:

1) Configuration:

a) Name : STRING

i) Not case sensitive

ii) This string is how the loot types are referenced in Missions when looking up how to create and fill a loot container.

iii) If the name is "SCATTER" an algorithm will be used to scatter the loot around the loot's spawn location.

iv) "SCATTER" is a reserved loot name for this effect.

b) Box Type: STRING

i) Needs to be a valid container class name

ii) If the type is "RANDOM" one of the box types specified in BaseServer.sqf will be used.

iii) If the type is "SCATTER" then FuMS loot scatter routine will distribute the items on the ground randomly around the loot location.

c) Randomizer: STRING

i) If any value other than "RANDOM" is in this value, the loot will be ignored when a mission is configured to select loot randomly.

ii) Setting "RANDOM" will add the loot description to the pool of 'randomly' generated loot that certain missions are configured to use. (See Loot section of the MissionFile.htm).

2) Loot Sections

a) Loot description is broken into 4 sections

i) Weapons

ii) Magazines

iii) All other Items

iv) Backpacks

b) Each item defined in any of these 4 sections follows the following format:

i) ["<type>", quantity]

OR

ii) ["<type>",[quantity, add_min, add_max]]

(1) "<type">

(a) should be a valid item class name

(b) may also be a global variable containing a list of class names. FuMS will select 1 item from the list randomly.

(2) quantity - number of items that will be placed into the container

(3) add_min - ADDITIONAL minimum items to be placed in the container

(4) add_max- ADDITIONAL maximum items to be placed in the container

Ex: [Food_Cooked, [1,2,4]]

An item from the 'Food_Cooked' global variable will be selected.

Then 3 to 5 (1+2 to 1+4) of this selected item will be placed into the container.

3) Notes

a) Loot Options: "SCATTER" may be used in place of a Loot option name or "RANDOM" in the 'Loot Configuration' section of all missions. When "SCATTER" is specified, the loot option named "SCATTER" is selected. Loot contained in the 'SCATTER' option (defined in LootData.sqf or GlobalLootData.sqf) will be distributed on the ground in random directions around the loot's location. This options works WITH vehicles, so, if in the mission file, "SCATTER" is the loot option for the vehicle, the loot will be scattered around the vehicle when it is spawned. No loot box will be created when "SCATTER" is used.

b) For magazines the value of FuMS_AmmoForEach may be defined. When this is defined, ammo will be provided in the loot box for each weapon defined in the 'Weapons' section of the loot description.

NEXT SECTION

File:

FuMS\Themes\GlobalSoldierData.sqf

Description:

This file contains configuration information that defines default FuMS AI soldier types and their associated load-outs. This file is referenced by all themes (by default) when building AI for missions.

More soldier types can be added to this file, as long as the proper format is followed, as described below

Sections:

1) Name : STRING

a) Not case sensitive

b) This string is how the AI are referenced in Missions, when looking up how to configure the AI.

2) Skill Levels : ARRAY : 8 scalar values.

a) each value should range from 1.0 to 0.0

b) See BaseServer.sqf for override values

c) If the value in BaseServer is greater than zero the value in this array will be ignored.

d) *BaseServer values override skill values in this file!

3) Uniform : ARRAY: ["<type>", %chance]

a) "<type>"

i) should be a string class name for a valid uniform

ii) a Global variable name may be placed here instead

(1) Do not place the array name in quotes

(2) Array needs to be a valid list of uniform classes

(3) FuMS will select a random uniform from the global list

b) %chance

i) value between 1.0 to 0.0

ii) sets the chance that the AI will have the particular piece of equipment.

4) Vest, Helmet, Backpack, Primary Weapon, Secondary Weapon

a) Follows same rules and format as Uniform.

5) Primary Weapon Attachments: ARRAY : [%scope, %suppressor, %flashlight]

a) Each element of the array should be a value between 1.0 and 0.0

b) The value sets the percent chance the AI's weapon will have the specific piece of equipment.

c) FuMS will automatically match suppressor to the weapon type.

d) %scope may ALSO be a string or Array of strings. If it is desired to equip a specific scope the name of the scope can be placed here instead of a %chance.

Ex: [1, .3. 0]   change to ["optic_tws", .3. 0]

6) Belt Items: ARRAY: [%Map, %Compass, %GPS, %Watch, RadioType]

a) %'s follow same guidelines as Primary Weapon Attachments

b) Radio Type: 1-9, value of 0 = no radio

i) Under Exile, any non-zero value will place a radio on the AI.

7) Visual Items : ARRAY: [%Binoculars, %RangeFinders, %NVG's]

a) %'s follow same guidelines as Primary Weapon Attachments

8) Soldier Options:ARRAY: [DOW, UAmmo, AnitTank_Air, ISCAPTURED]

a) DiverOverWater (DOW)

i) true = AI that spawn over water will be automatically equipped with re-breathers and swim suites to permit them to swim to shore

ii) AI will KEEP their configured weapons.

iii) If AI with underwater weapons are desired, use the "DIVER" soldier definition in this file, or create one of your own.

b) Unlimited Ammo (UAmmo)

i) true = AI will not run out of ammo for their Primary weapon

ii) Warning: some weapons behave abnormally in AI control when provided with an unlimited ammo supply.

c) AntiTank_Air, one of true, "LAND", "AIR", or "RANDOM"

i) true = AI will be provided with an Anti-Tank weapon and ammo

ii) "LAND" = same as 'true'. AI will be provided with an Anti-Tank weapon and ammo

iii) "AIR" = AI will be provided with an Anti-Air weapon and ammo

iv) "RANDOM" = AI will be provided with either an AA or AT weapon and correct ammo. 50/50 chance of either.

d) ISCAPTURED

i) true = AI will not be fired upon by other AI, even of opposite faction

ii) Has no impact on "CAPTURE" AI logic. Capture AI logic will set/unset this flag on its own.

9) Soldier Gear : ARRAY: [["<type>", %chance], [min, max]]

a) Type, %chance portion follow same guidelines as in the Uniform section

b) [min, max] = IF the %chance is successful, the AI will then have a random number of the <type> item ranging between min to max.

10) Tabs and Respect: ARRAY: [["<type>", amount], ["<type">, amount]]

a) Establishes values to be awarded when this AI unit dies.

b) Available Options:

i) ["RESPECT", 10] = Provides 10 respect to the player killing the AI

ii) ["RESPECTGROUP",1] = Provides 1 respect to all players in the killing player's group when the AI dies.

iii) ["RESCUEGROUP",500] = Provides 500 respect to all group members when they rescue an AI.

iv) ["RESCUETABS",250] = Provides 250 tabs to all group members when they rescue an AI.

NEXT SECTION

File:

FuMS\Themes\<YourTheme>\ThemeData.sqf

Description:

This file controls overall behavior of the theme. This file controls what missions the theme should execute, and how and where the missions should start. All missions under this theme's control MUST be located in the theme's folder, along with this ThemeData.sqf file.

A Theme will create an independent theme control loop. This loop will run and monitor 1 mission at a time. The mission selected is determined by this configuration file. When one mission completes, the control loop will select another mission to run.

Sections:

1) Options

a. Establishes overall theme behavior.

b. Parameters

i. THEME NAME: STRING: <YourTheme>

1. Name must match folder name

2. Should contain no spaces or special characters

ii. MISSION SELECTION: SCALAR : 1

1. 1= Missions selected randomly from 'Mission List'.

2. 2= Missions run in the order of the 'Mission List'.

3. 3=Missions selected randomly from 'Mission List'. Once a mission is run, it will not be selected again until all mission on the list have run.

4. Creates an independent theme control loop for each mission in 'Mission List'. Then starts each of these control loops, launching the missions. Effect is to cause all missions in the list to be started at the same time.

a. Once started, each mission is running on its on control loop, independent of the other missions.

b. When a mission started with this option completes, its control loop will restart it after the configured delays expire.

5. Identical to option '4', except that the missions will only run once per server reset.

iii. RESPAWN DELAY: SCALAR: 60 (seconds)

1. Duration FuMS waits before selecting and launching another mission from this theme.

2. This timer is IN ADDITION TO the timer located in each individual mission under its Win and Lose status.

3. If the mission that was launched uses a fixed location there is the potential that the mission will be re-spawned on top of players after they complete the current mission. If the mission is using a random location (not fixed and not using a point from the 'Locations' section of this file) FuMS will attempt to avoid spawning a mission on top of players, their bases (if occupied), or other missions.

iv. GLOBAL LOOT: BOOL: true

1. true = Theme will use the GlobalLootData.sqf for creation of loot

2. false = Theme will use its own 'LootData.sqf'

a. This 'LootData.sqf' must be located in the folder with this theme.

b. The theme will not have access to loot options in the Global file.

v. GLOBAL SOLDIER:BOOL: true

1. true = Theme will use the GlobalSoldierData.sqf for AI creation

2. false = Theme will use its own 'SoldierData.sqf'

a. The 'SoldierData.sqf' must be located in the folder with this theme.

b. The theme will not have access to AI definitions in the Global file.

vi. AUTOSTART: BOOL : true

1. true = Theme control loop will start with FuMS initialization at server start.

2. false = Theme will remain off.

a. Use this option for a theme you wish to control with the admin tools

b. If you wish to make a theme unavailable on your server, it is recommended to comment it out of the BaseServer.sqf instead of setting this value to false.

vii. MINPLAYERS: SCALAR: 1

1. When number of players on line is greater than or equal to this number, the theme will run its missions.

2. Missions in play when player count drops below this number will remain. Once the mission is cleared the theme will suspend further missions until the player count is again reached.

viii. MAXPLAYERS: SCALAR: 100

1. When the number of players is greater than or equal to this number, the theme will stop running missions.

2. Missions in play when the player count exceeds this number will remain. Once the mission is cleared the theme will suspend further missions until the player count drops below the number.

c. Notes:

2) Mission List

a. List of missions that are available to be launched by the theme's control loop.

b. Missions that are launched from other missions (see phasing) and reinforcement missions should not be listed in this section unless it is your intent for the missions to also be created independently as a standalone mission.

c. For each item in the mission list there shall be an associated .sqf file with the same name located in this theme's folder.

d. Mission names should not contain spaces or special characters.

e. Parameters: Methods the missions can be defined:

i. ["BikeGang"]

1. The file 'BikeGang.sqf' will be used to define the mission

2. The location for the mission will be selected randomly from the 'Locations' section of this themedata.sqf file.

3. If the 'Locations' section is empty, FuMS will generate a random location based upon its SmartLoc processing.

ii. ["TestMission01",[10700,12200]]

1. The file 'TestMission01.sqf' will be used to define the mission

2. The mission will be created at the location [10700,12200]

3. Defining a mission in this way ties it to a specific map.

4. Offsets locations defined within the mission file will still function correctly.

5. Good method to place a mission with offsets at a fixed location.

iii. ["LandPatrol","Zaros"]

1. The file 'LandPatrol.sqf' will be used to define the mission

2. The mission will be centered in the town of "Zaros"

3. Defining a mission in this way ties it to a specific map.

4. Offset locations defined within the mission file will still function correcltly.

f. Notes:

i. FuMS SmartLoc processing:

1. Only utilized when creating a mission at a RANDOM location that is not utilizing the 'locations' section of the theme.

2. Attempts to find a point for the mission that does the following:

a. Does NOT overlap another FuMS mission

b. Does NOT spawn near players

c. Does NOT spawn near player bases, IF players are near the base.

d. If mission is defined as "LAND", mission area will all be land

e. If mission is defined as "WATER", mission area will all be water

3) Locations

a. This section provides a method to define a specific list of locations for your missions. This list is primarily designed to permit an Admin to control a mission set to a specific set of locations.

b. If a certain mission will always spawn at the same location, the static location can be specified along with the mission in the 'Mission List' above.

c. If 'totally' random locations are desired that are also checked with FuMS SmartLoc, leave this section blank.

d. Parameters:

i. Each of these can be combine within this list to generate a large array of possible locations

ii. [x,y]

1. Specific map location

2. Will make the theme Map specific

iii. ["Villages"]

1. Will identify all Villages on the map and add them to the list of Locations

2. Works on all maps

iv. ["Cities"]

1. Will identify all Cities on the map and add them to the list of Locations

2. Works on all maps

v. ["Capitals"]

1. Will identify all Capital cities on the map and add them to the list of Locations

2. Works on all maps

vi. ["Marine"]

1. Will identify all named water regions such as bays, and oceans

2. Works on all maps

3. These locations WILL be in water in most instances. There are some maps that define certain "Marine" areas that only have water at certain times or represent old/historical "Marine" areas.

vii. [<Name>]

1. <Name> may be replaced with the name of any village, city, or capital on the map

2. This location will be added to the list of Locations

3. This will make the theme map specific.

viii. [[x,y],"<Name>"]

1. [x,y] is a specific location on your map.

2. This will make the theme map specific

3. <Name> may be replaced with whatever text you desire.

4. Whatever mission spawns at this location will be given the name of <Name> on the map and in mission messages.

5. This permits a fix location on the map to be created, that has the potential to have random missions/setups at the location. All depending on how you customize the rest of the theme.

.

e. Notes:

i. FuMS SmartLoc is overridden by any locations defined here. Thus if the theme has locations defined within the 'Locations' section, those points will be the pool of points used to spawn missions, and NO overlap/player protection is provided.

ii. SmartLoc used by other themes WILL recognize these missions and work to avoid them.

4) Radio Chatter

a. This section defines how the radio chat in the upper left hand corner of the player's display will behave.

b. The size, position, color of this display can be modified by changing the settings in the description.ext file (see installation instructions)

c. Global control of Radio Chatter is set in BaseServer.sqf

d. Parameters

i. RADIOCHANNEL: STRING or SCALAR:

1. When radio chat enabled this controls what radio channel this theme's messages go too.

2. "ALL" = messages heard without players being required to have a radio

3. 1 = Value of 0-9, sets the radio required to hear chat for this theme.

4. 0=Quartz, 1=Garnet, 2=Citrine, 3=Amethyst, 4=Topaz, 5=Sapphire, 6=Onyx, 7=Emerald, 8=Ruby, 9=Jade

ii. SILENTCHECKIN: BOOL: false

1. true = AI squads will NOT check-in with their BaseOps when they spawn

2. false = AI squads will check in with their BaseOps when they spawn.

a. This option will often provide some context clues to the AI's location to the players.

iii. AI DEATH MESSAGE: BOOL: true

1. true = AI death's will be announced over radio chatter

2. false= no AI death messages.

iv. RADIO RANGE: SCALAR: 1500

1. value in meters.

2. Affects how far away from the encounter AI radio chatter can be heard

3. to BaseOps responses can be heard from any range.

v. CALLSIGNS: STRING

1. For the AI and BaseOps, this sets the name of the AI groups and their BaseOps

2. These names are used in the radio chatter text to aid in telling the difference between radio chatter from different themes that may be on the same radio.

vi. CONVERSATION TEXT :ARRAY

1. Each array is formatted in an 'Action', "Text Response" pair.

2. The 'Action' sections must NOT be modified

3. The 'Text Response' may be changed to suit the needs of the theme

4. The '<' and '>' are reserved characters. Do not use these in the chat except to define the Keywords described below.

5. Keywords that may be placed inside the text

a. <DIST> = will place the distance the AI is from the encounter center into the text.

b. <DIR> = will place the direction the AI is from the encounter center.

c. <MSNNAME> = will place the mission name into the text

d. <POS> = will place the AI's position into the text

e. <#ALIVE> = number of AI remaining in the group making the radio call will be included in the chat text.

f. <#DEAD> = number of AI that have died in the caller's group.

g. <STATUS>= provides text that indicates if the AI are engaged with players or other AI.

e. Notes

i. None.

5) Custom Scripts

a. This section defines all custom scripts the theme will make reference too.

b. All scripts are assumed to have a .sqf extension!

c. Scripts listed in this theme must use unique names from scripts listed in other themes, unless it is intended for those themes to utilize the same script.

d. If themes do use the same scripts, the script MUST be present in both theme's folder.

e. Scripts used as Start and End scripts in missions will be passed certain mission information to be used by the custom scripts:

i. StartScript

1. Runs after are initial mission messages and objects are created, prior to 'mission state/logic' monitoring.

2. Inputs to all Start Scripts!

```
[ _encounterCenter, _missionOffset, _buildings, _groups, _vehicles, _lootboxes]
```

a. _encounterCenter - center of entire mission [x,y]

b. _missionOffset - offset from mission center. Building, AI, vehicles, lootboxes use this location as their center for spawning and AI logic. [x,y]

c. _buildings - list of all building OBJECTS created by the mission

d. _groups - list of all AI group OBJECTS created by the mission

e. _vehicles - list of all vehicle type OBJECTS created by the mission

f. _lootboxes - list of all container type OBJECTS created by the mission

3. Required Outputs for ALL Start Scripts:

[_buildings, _groups, _vehicles, _lootboxes, _yourCustomVariables]

a. _yourCustomVariables - placed what you want in this variable. It will be passed to the custom End Script. From the end script you can extract your needed custom data.

4. See \FuMS\Themes\Test\Scripts\StartExample.sqf for an Example!

ii. EndScript

1. Runs after mission completion parameters have been met and end mission messages and loot has been created.

2. Inputs to all End Scripts.

[_encounterCenter, _missionOffset, _buildings, _groups, _vehicles, _lootboxes, _yourCustomVariables]

a. See StartScript for variable descriptions

b. _yourCustomVariables - whatever data you chose to place in it via the mission's Start Script.

3. Required Outputs for ALL End Scripts. - NONE.

4. See \FuMS\Themes\Test\Scripts\EndExample.sqf for an example.

iii. Other Custom Scripts

1. Start and End Scripts can call additional custom scripts.There are two options to calling these scripts from within the Start/End scripts.

2. Storing the scripts Server Side:

a. Place the script in the \FuMS\Themes\<yourTheme>\Scripts folder

b. Place the script name in the list of scripts section of the ThemeData.sqf

1. (exclude the .sqf extenstion)

2. .sqf will be assumed to be the scripts extension

c. make calls or spawns to this script via the name used in the ThemeData.sqf

d. See \FuMS\Themes\Test\Scripts\StartScript for example usage.

3. Storing the scripts Client Side: (in the MPmission.pbo)

a. Add the scripts or addon to your MPmission.pbo

b. From within the Start and End scripts, make the appropriate execVM or calls with the appropriate path.

c. This path will be just like any other normal addon stored in your mpMission.pbo.

d. See \FuMS\Themes\Test\Scripts\StartScript for example usage.

.

NEXT SECTION

File:

FuMS\Themes\<YourTheme>\<YourMission>.sqf

Description:

This file defines all the specific characteristics of a mission.

Sections:

1) Title and Radius

a. Establishes overall theme behavior.

b. Parameters

```
["<YourMission>", 200,"NONE", [min, max, direction], "StartScript", "EndScript" ]
```

i. Mission Title

1. Should contain No spaces or special characters

2. Used internally by FuMS for tracking the mission.

3. Never made available for player view.

ii. Radius

1. Sets the encounter radius

2. Used in testing for mission overlap when settling on the missions location (if random)

3. Used in AI logic as the default 'max' range if AI logic ranges are not specified

4. DOES NOT impact circle size displayed on map, if this option is enabled.

iii. Random location Option

1. "LAND", "WATER", "NONE"

2. Setting this will force a scan of 'encounter Radius' meters around the center of the mission to ensure the same type of land/water is present

3. This evaluation is ONLY done if the mission's location is not being set to a location in the 'Locations' section of the mission's Theme.

iv. Positioning

1. After a good mission location is found, these three values position the mission's actual center a random distance from the original mission location (origin).

2. Parameters

[min, max, direction]

a. min - minimum distance from mission center that encounter origin is placed.

b. max - maximum distance from mission center that the encounter origin is placed.

c. direction - compass direction of 1-360 degrees that the mission center is moved from the origin.

1. 90 implies offset to the east, 180 offset to the south, etc.

2. A value of zero will imply a random direction will be selected.

d. Buildings, AI, Vehicles, and Loot boxes will use this 'position offset from center' when being created.

e. This location will also establish the 'centroid' for all FuMS AI logic and patrol patterns

f. The effect of using this sort of offset is that the 'bulk' of the mission can now be randomly placed within the mission's overall radius.

v. StartScript and EndScript

1. **NOTE: These options ONLY available if the positioning (item iv) is defined (default of [0,0,0])

2. Script identified should be specified TO include the file extension!

a. Example: "StartScript.sqf"

b. FuMS will look in the Theme's 'Scripts' folder for this script.

c. The script will be executed with a 'call' to the script. Based upon the admin's script, it could possibly suspend mission execution until the custom script completes.

3. See ThemeData.htm for further details

c. Notes: None.

2) Map Marker Configuration

a. Sets configuration options for map markers

b. FuMS uses two separate markers, grouped together for each mission

c. Parameters

```
["<Your Mission Name>","mil_objective","ELLIPSE","ColorRed","FDiagonal",200]
```

i. Mission Name

1. Name displayed to players in radio chatter and on the Map

2. If mission obtained its location from the 'locations' section of ThemeData.sqf and that location was from one of the keyword 'villages, towns, capitals, etc' this mission name will be overridden with the location's name from the ThemeData file.

ii. Type

1. May be any valid map maker object types

2. https://community.bistudio.com/wiki/cfgMarkers

iii. Shape

1. May be any valid map marker shape

2. "ICON", "RECTANGLE", or "ELLIPSE"

iv. Color

1. May be any valid map marker color

2. https://community.bistudio.com/wiki/setMarkerColor

v. Brush

1. https://community.bistudio.com/wiki/setMarkerBrush

vi. Size

1. Sets size of the map marker in meters

2. Does not impact encounter area, only size on the map.

d. Notes: None.

3) Notification and Messaging

a. Sets up how players are notified about this mission when it starts

b. Also sets up victory and failure messages if the option is enabled

c. All sections must be filled in, even if not used, or FuMS will reject the mission for format.

d. Parameters

```
[  true, "ALL",0, // radio config
```

```
  true,   // global messages

  true,  // map display

  30,   // win delay

  30,   // lose delay

],  [ <global message configuration>]
```

i. Radio Configuration

1. Turns On/OFF radio messaging for this mission

2. "ALL" - no radio required to hear messages

a. 0-9 = sets the channel the AI will use for chatter

b. Player must have that radio type on his belt to hear chatter

3. Range - distance from encounter center that players can hear AI chatter

a. chatter from the AI's 'base ops' has no range limitation

b. when players are outside range they will only hear one side of a conversation.

ii. Turns Global Messages On/OFF

1. Global messages consist of a spawn, victory, and failure message

2. Actual text for these messages is set up in the <global message configuration> portion of this section.

iii. Turns Map display On/OFF

iv. Win and Lose delays

1. Time in seconds after the condition is met before FuMS begins mission cleanup.

2. If AI are engaged with players when this timer expires, they will remain in play until a period of 3 minutes passes where the AI do not detect players.

v. <global message configuration>

1. This section sets up the text to be displayed in the global message for the mission

2. Format is:

a. Message Title

b. Message Header

c. Message Text

e. Notes: None

4) Loot Setup

a. Establishes loot options for the mission

b. 3 sections,

   i. Static - loot spawns with mission

   ii. Win - loot spawns when 'win' conditions are met

   iii. Lose - loot spawns when 'lose' conditions are met

c. Parameters

```
["Scatter" , [0,0] ]

   or

["Scatter" , [0,0], ["Random", [5,5] ]

// Note above example creates multiple loot boxes!
```

   i. Loot Type

1. Text value that is used to look up the loot in the LootData file assigned by the Theme.

2. Key word "RANDOM" will select a random loot type from the LootData file

3. Key word "SCATTER" will use that specific loot mechanic (see LootData documentation)

ii. Offset - where the loot should spawn relative to the encounter center

1. A hard coded location may also be used

2. hard coded position can be used to spawn loot at a safe zone, or spawn point, or unique geographic location.

3. Hard coded positions makes the mission 'map' specific.

d. Notes:

i. An array of loot can be specified in any of the three sections to provide multiple loot containers at multiple locations.

5) Building Configuration

a. Determines placement and behavior of buildings and 'static' vehicles.

b. Manned static emplacements should be specified in the AI Vehicle section.

c. Accepts arrays of building data. Each element of the array must meet one of the following formats:

i. Static Vehicle

```
["I_UGV_01_rcws_F",[0,100],   0, [.5,   1,     .5,         .5,         .5]]
```

or

```
["I_UGV_01_rcws_F",[0,100],   0, [.5,   1,     .5,         .5,         .5], "FIRE_SMALL"]
```

1. Class Name - valid Arma3 vehicle class name

a. option: May be an array of class names, one of which will be selected randomly.

b. ["Land_UWreck_Heli_Attack_02_F","Land_Wreck_Heli_Attack_01_F"]

2. Offset - x,y distance from encounter center object will be placed

3. Rotation - 0-360 - direction object will face when created

4. Condition Array

a. Fuel - 0=empty 1=full

b. Ammo - 0=empty 1=full

c. Engine Damage - 1=full 0=no damage

d. FuelTank Damage - 1=full 0=no damage

e. Hull Damage - 1=full 0=no damage

5. Optional Fire/Smoke parameter

a. If used, must be one of the following:

b. "FIRE_SMALL", "FIRE_MEDIUM", "FIRE_BIG", "SMOKE_SMALL", "SMOKE_MEDIUM", "SMOKE_BIG"

c. Use of one of these options will result in the applicable effect being attached to the object.

ii. Building

```
["CamoNet_INDP_big_F",      [-20, 10],0,      0]
```

or

```
["CamoNet_INDP_big_F",      [-20, 10],0,      0, "SMOKE_BIG"]
```

1. Class Name - see Vehicle definition above

2. Offset - see Vehicle definition above

3. Rotation - see Vehicle definition above

4. Persist - controls building behavior after mission end.

a. 1 = building remains until server reset

b. 0 = building is removed with mission cleanup

5. Optional Fire/Smoke parameter - see Vehicle definition above

iii. In case 'i' and 'ii' above the "Class Object Name" may be replaced with an array of "Class Object Names"

```
["Land_UWreck_Heli_Attack_02_F","Land_Wreck_Heli_Attack_01_F"],[0,0], 0,0,"Fire_Big"]
```

1. In the above example one of the two helo wrecks will be selected at random by FuMS when the mission is created.

2. This random object selection is not available inside the M3 Editor Format described below.

iv. M3 Editor Format

```
["M3Editor", [0,0], 777, 0,

     // paste your array of building objects here

  [

     ["Land_Dam_Conc_20",[7442.48,7118.03,-0.0066452],122,[[0.848048,-0.529919,0],[0,-0,1]],false],

     ["Land_Dam_Conc_20",[7431.81,7101.22,0.04673],122,[[0.848048,-0.529919,0],[0,-0,1]],false],

     ["Land_Dam_Conc_20",[7421.2,7084.37,0.328979],122,[[0.848048,-0.529919,0],[0,-0,1]],false],

    ----additional M3Editor objects...........

  ]

]
```

1. M3Editor - Keyword that tells FuMS to follow M3Editor object format
2. Offset - similar to offset in Vehicle and Building definition

a. offset sets the location for the 1st building in the list

b. all other buildings will be placed in relation to the 1st building

c. FuMS automatically calculates offsets based upon the locations in the M3Editor objects

d. This offset mechanic allows files built on one map to be ported to other maps with no additional effort by the author.

e. if offset is [-1,-1] FuMS will use the actual locations contained within each M3Editor object.

3. 777 - place holder - provides Smoke/Fire options for all objects in the M3Editor array.

a. see part 5 Optional Fire/Smoke parameter.

b. replacing 777 with one of the optional Fire/Smoke strings will result in ALL buildings/objects within the M3 Editor definition to have the described effect!

4. Persist - see Building definition above.

5. Array of M3 Editor objects

a. Direct cut and paste of array of objects from M3 Editor into this array

d. Notes:

    i. Any combination of the above definitions may be used within a single mission

ii. If the M3Editor definition is used, it is recommended to place it 1st in the overall Building Configuration section.

iii. Setting the offset to anything other than [-1,-1] will enable the M3Editor objects to be used on any map, regardless of where the design was original created.

6) AI Group

a. Creates groups of AI for the mission and configures their behavior.

b. AI group section can be a list/array of objects as defined below:

i. Each line creates a new Arma3 'group' object

ii. AI will remain in their group and follow the logic assigned.

c. Parameters

```
[ ["RESISTANCE","COMBAT","RED","LINE"],[[1,"Sniper"],[3,"Rifleman"]],[  "TowerGuard",[0,0],
[0,0],[150,"ANY"] ]]
```

i. Side and Formation

1. Side:

a. RESISTANCE, WEST,EAST,CIV

b. Multiple sides can be defined within one mission

2. Behaviour:

a. SAFE, AWARE, COMBAT, STEALTH

3. Combat Mode:

a. BLUE, WHITE, GREEN, YELLOW, RED

4. Formation:

a. STAG COLUMN, WEDGE, ECH LEFT, ECH RIGHT, VEE, LINE, COLUMN

ii. Soldier Type Array

1. List of [quantity, "soldier type"]

2. "soldier type" is referenced in the SoldierData file set by the Theme

3. In the above example the group will contain 4 AI

4. 1st AI listed is defaulted to the group's leader

5. Leader will always have a radio of the type configured for the mission.

iii. AI Logic

1. Sets how the AI will behave

2. Logic Name - See AI_Logic documentation for specifics!

3. Spawn Loc - sets location AI will spawn

4. Dest Loc - sets location where AI will 'start' their logic

a. If spawn loc and Dest loc are different, the AI group will proceed from their spawn loc to the destination location before starting their assigned AI logic.

5. Other parameters - as defined by the AI logic selected.

d. Notes : None

7) AI Vehicle

a. Creates groups of vehicles and associated AI for missions

b. Section is grouped into Convoys

i. Each convoy has the following:

1. Group of Vehicles - each vehicle contains:

```
[  "B_Truck_01_transport_EPOCH",[-50,-610],[1,"Rifleman"],        "Truck01",[1.0]      ]

                        or

["O_Mortar_01_F",[0,60],[0],"None"]

// NOTE: above example is for 'static guns'.  Crew section is replaced with a 'facing'
```

a. Vehicle class name

1. can be any valid vehicle class name

2. Static Weapon, Vehicle, Helo, Boat, or aircraft

b. spawn location

1. offset from encounter center

2. or Static map specific location.

c. Crew members

1. assigned to the convoy's Driver group

2. will drive in place of the actual drivers

3. assigned to any hard points in the vehicle by default

4. will stay with the vehicle until it is disabled.

5. For STATIC WEAPONS the crew list is replaced by the 'facing' of the weapon!

d. Loot

1. Loot type taken from LootData as set by the Theme.

2. Loot is placed into the vehicles cargo.

3. Use "NONE" if no loot desired.

e. Damage

1. Optional setting

2. Value 0.0 - 1.0

3. Sets the amount of 'damage' that the vehicle has already sustained.

1. 1.0 = vehicle totally broken

2. 0.0 = vehicle fully healthy

4. Optional Array:

1. Can be any or all of the following:

2. ["engine",.5],["fuel",.5],["hull",.5],["vrotor",.5], ...etc

3. other examples: https://community.bistudio.com/wiki/getHitPointDamage#Notes

4. (just remove the 'hit')

5. See Captive.sqf mission file for example.

2. Drivers group

```
[["RESISTANCE","COMBAT","RED","COLUMN"],   [  [3, "Driver"]  ],   ["Gunner",[-25,0],[0,0],[0]   ]]
```

a. Follows identical format as the Group AI section

b. Any AI_Logic may be used. (some may not make sense though.....)

c. Spawn location does not matter because members will be placed into individual vehicles.

d. Drivers used for Static Weapons should use the specific 'Gunner' logic so that they are assigned to the correct hard point.

e. Drivers are assigned 1 vehicle at a time, excess drivers are ignored.

f. If there are insufficient drivers, the vehicle will be without a driver!

g. NOTE: For UAV and UAG the 'side' is determined by the class name of the vehicle, not the setting in this section!

1. (I=Resistance, O=East, B=West)

3. Troops Section

```
[["RESISTANCE","COMBAT","RED","COLUMN"],[[1,"Sniper"],[1,"Rifleman"]],["BoxPatrol",[-70,-
600],[0,0],[0]]]
```

a. Follows identical format as the Group AI section

b. Multiple groups may be listed

c. Groups listed in this section will be placed in the 'cargo' area of the vehicle

d. Spawn location for these groups does not matter because the groups will be 'auto inserted' into the vehicle upon creation

e. If vehicle becomes disabled, or Driver logic drops off troops, troops will then proceed to their 'destination' loc and execute their AI Logic.

f. Troops in excess of the vehicles capacity will be ignored (not created), and a log written to the HC's .rpt file.

ii. Multiple convoys may be defined

1. Just as in the AI group section, AI may be created on different 'sides' within one mission.

8) Mission Logic

a. This section sets up all 'triggers' that will be monitored by the mission, as well as how those triggers will produce certain actions such as win, lose, and mission end results.

b. Mission Logic is broken into two sections:

i. Trigger List

1. This section is an array/list of Trigger Name/ Trigger definition pairs.

```
["<TriggerName>",[ <Trigger Definition> ]
```

2. Trigger Name

a. A unique name that is NOT case sensitive.

b. Name should not use spaces or special characters

c. This name is what is used in section 2 (below) to identify what sort of logic is attached to the trigger.

d. Reserved Names:

1. "FuMS_KillMe" - used internally by FuMS to close missions from outside normal control logic.

2. "OK" - trigger defined in all missions. This trigger will always evaluate to true.

3. Trigger Definition

a. One of any of the Trigger Definitions found in the Trigger Definition section of this document.

b. Ex: ["Trig1", ["ProxPlayer",[0,0],150,1]]

1. Trig1 is a player proximity trigger that will evaluate to true when '1' player gets within 150m of the encounter's center [0,0].

ii. Action Logic Section

1. This section defines 'Action' / 'TriggerName' statement

2. When the 'TriggerName' portion evaluates to true, the defined Action will occur.

3. Action

a. Below are the currently defined actions in FuMS

b. ["WIN"]

1. Causes any mission defined victory messages and loot to be produced.

2. Prevents "Lose" action once a Win occurs.

c. ["LOSE"]

1. Causes any mission defined loss messages and loot to be produced.

2. Prevents "WIN" action once a Win occurs.

d. ["END"]

1. Causes all defined triggers for the mission to terminate

2. deletes all objects related to the mission.

3. terminates the mission and all the mission's children.

e. ["CHILD",["<MsnName>",[location],reps, frequency (secs)]]

1. Launches the indicated mission based upon the provided parameters.

2. Children missions will 'pass on' their resources to the parent when the mission ends.

3. "<MsnName>"

1. Name of the mission in the theme's folder to be launched.

2. does NOT include the .sqf extension

4. Location mission should start

1. See XPos definition for variations

2. standard [x,y] is an offset from the 'parent' mission's center

5. Reps

1. Maximum number of times the 'parent' can create this Child.

6. Frequency

1. Minimum time in seconds between Children spawnings.

2. Once this minimum time has been met, the 'Triggers' that result in this action must still evaluate to TRUE in order for the next child to spawn.

7. See RaidTown.sqf in the TownRaid theme for a working example!

f. ["STEPCHILD",["< MsnName>",[location],reps, frequency (secs)]]

1. Launches the indicated mission as a STEPCHILD

2. StepChildren do NOT pass on resources to the parent when the mission ends.

3. Other parameters identical to "CHILD".

4. TriggerName list

a. This is a list of trigger names, as defined in section two of 'Trigger List'

b. The list is a comma separated array.

c. When the statement evaluates to true, the Action associated with the statement will be executed.

d. TriggerNames seperated by commas (,) are AND'd together.

e. The keyworkd "OR" can be used within the statement.

```
["END",["Trig1","Trig3","OR","Trig2"      ]]
```

f. In the above example the END action will be executed with Trig1 AND Trig2 evaluate to TRUE, OR when Trig2 is TRUE.

g. See TestMission01.sqf for an example mission.

9) Misc Instructions

a. Incorporating M3 Editor Files into your Mission

i. Take building array from your M3Editor file starting at:

```
_objs = [ ......... ];
```

ii. Copy the outer most brackets and everything in between

```
    [ .........]
```

iii. Paste what you copied into the array below.

```
["M3Editor", [-1,-1], 777, 0,

   // paste your array of building objects here

]
```

iv. Place a copy of the array in step 3 into the 'Buildings' section of your mission. This array you are pasting is treated just like a previous FuMS building definition. Just ensure you place this data as the 1st element in the buildings section. The mission can still contain other vehicle and building definitions.

v. If you desire placement to use an 'offset' specify it in the parameter after "M3Editor".

```
   (EX: ["M3Editor", [0,0], ......
```

vi. If you desire placement based upon the locations in your M3Editor file, set the offset to [-1,-1]

(EX: ["M3Editor", [-1,-1], ...

vii. Notes:

1. The 1st component listed in your building objects array will be used as the 'anchor point' when offset is used.

2. Setting '0' to '1' in the last parameter will cause the structures to remain after mission completion.

(Ex: ["M3Editor, [0,0], 777, 1, ...

3. If you are pulling a building complex/file that was built for another map, make sure you use the [0,0] offset mentioned in step 5. And if you need to place this building complex/file in a specific location, specify this location by statically defining the mission in the ThemeData.sqf.

4. Ex: ["testMission01",[12300,45600]] will spawn this mission at [12300,45600], with the 1st building in your list being located at [12300,45600].

b. XPos and Location assignments

i. All location definitions in FuMS follow the below behavior.

ii. [x,y] - 2D

1. Standard is [x,y] which will generate an x,y OFFSET from the encounter center.

iii. "<TownName>" - Map Name

1. When used, the town name will be located by FuMS and the center of that town's location used.

iv. [x,y,z] - 3D

1. FuMS will assume this is an actual map location and use the values without modification.

c. AI Logic Definitions

Last updated 5/21/15.

i. This section contains details on the AI patrol behaviors available in FuMS. Parameters and syntax listed are what is proper for use within mission files, when building or modifying a mission. The options shown in this file do not describe the actual script's parameters. Refer to the specific .sqf script for those details. This document is available as a quick reference for mission editors and designers.

ii. Valid Logic Objects

1. ["BUILDINGS",   [spawnloc], [actionloc], [range]]

2. ["EXPLORE   ", [spawnloc], [actionloc], [radius]]

3. ["BOXPATROL",   [spawnloc], [actionloc], [radius]]

4. ["CONVOY",   [spawnloc], [actionloc], [speed, FlagRTB, FlagRoads, FlagDespawn, convoyType]]

5. ["SENTRY",   [spawnloc], [actionloc], [radius]]

6. ["PARADROP",   [spawnloc], [actionloc], [speed, altitude, FlagRTB, FlagSmokes]]

7. ["PATROLROUTE", [spawnloc], [actionloc], [behaviour, speed, [locations], FlagRTB, FlagRoads, FlagDespawn, flyHeight]]

8. ["ZOMBIE", [spawnloc], [actionloc],[detection_Range, FlagCanThrow, Wonder_range]]

9. ["TRACKROUTE", [spawnloc], [actionloc], [behaviour, speed, [locations], FlagRTB, FlagRoads, FlagDespawn, flyHeight]]

10. ["TowerGuard", [spawnloc], [actionloc], [radius,"Optional_Building_Name"]]

11. ["Gunner", [spawnloc], [actionloc], [0]]

**All loc's can be defined via 3D map specific coordinates, 2D offsets, or a town name (ie "Starvos")

iii. "EXPLORE"

1. syntax: ["AREAPATROL",[spawnloc], [actionloc], [radius]]

2. -spawnloc: offset or specific map location group will spawn.

3. -actionloc: offset or specific map location at which group will begin their patrol.

4. -radius: size of the area, in meters, for the group to patrol.

5. Behavior: 12 evenly spaced waypoints are generated at a 'radius' to 'actionloc'. These waypoints are then randomized and the group set to start patrolling these waypoints. Group will recycle to the 1st waypoint when the 12th waypoint is reached.

iv. "BOXPATROL"

1. syntax: ["BOXPATROL", [spawnloc], [actionloc], [radius]]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: offset or specific map location at which unit will begin their building search

4. -radius: size of the area, in meters, for the group to patrol.

5. Behaviour: 4 waypoints are built at the corners of a square with a diagnol of 2*'radius'. The group will proceed to the eastern most waypiont then continue in a counterclockwise pattern.

v. "BUILDINGS":

1. syntax: ["BUILDINGS", [spawnloc], [actionloc], [range]]

2. -spawnloc: offset or specific map location group will spawn.

3. -actionloc: offset or specific map location at which group will begin their patrol.

4. -range: range from 'actionloc' that the group will look for buildings to search.

5. Behaviour: Units within the group will wait until they are on foot. At this piont they will proceed to 'actionloc'. Upon arrival, all buildings inside of 'range' radius will be identified. A random search of these buildings will begin. When a unit gets to a building, it will inspect all locations within the building before continuing onto another random building on its list.

6. Note: There is a tendancy for units to get 'stuck' on certain structures. A routine has been added to try and identify when a unit is stuck and teleport it to a nearby road segment.

7. If no buildings are found within 'range' the unit will continue to its current active waypoint (if any).

vi. "CONVOY"

1. syntax: ["CONVOY", [spawnloc], [actionloc], [speed, FlagRTB, FlagRoads, FlagDespawn, convoyType]]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: offset or specific map location at which unit will perform its drop-off or evacuation.

4. -speed: "NORMAL", "FULL", or "LIMITED" - limited and normal will provide better 'convoy' effects

5. -FlagRTB: true=group will return to 'spawnloc' upon completing its action at 'actionloc'.

6. false=group will drop off its cargo, then begin a boxpatrol pattern around their drop off location.

7. -FlagRoads: true=group's vehicles will spawn on a road, and drivers will attempt to use roads to get too 'actionloc'

8. -FlagDespawn: true=group, its vehicles, and its occupants will despawn upon returning to 'spawnloc'

9. -convoyType: "XFILL"= see below

10. Behavior: group will move from spawnloc to actionloc. At 'actionloc', if convoyType is "XFILL" an evacuation call will be made to all units within 200 meters of 'actionloc'. All units, up to the capacity of the group's vehicles will proceed to and board the vehicles. When loading is complete, vehicles will return to 'spawnloc'.

11. If convoyType is left blank, then at 'actionloc', vehicles will disembark all units in their cargo.

vii. "SENTRY"

1. syntax: ["SENTRY", [spawnloc], [actionloc], [radius]]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: offset or specific map location at which unit will take up its guard position.

4. -radius: distance from 'actionloc' that the group searchs for available buildings.

5. Behavior: Group will proceed to 'actionloc'. Upon arrival, group will search 'radius' for available buildings. Individual units in the group will then enter the buildings and climb to the highest points in the buildings and take up a "COMBAT" ready position.

viii. "PARADROP"

1. syntax: ["PARADROP", [spawnloc], [actionloc], [speed, altitude, FlagRTB, FlagSmokes]]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: offset or specific map location at which group (of pilots) will drop off their cargo (paratroopers)

4. -speed: "NORMAL", "FULL" or "LIMITED"

5. -altitude: in meters pilots will fly their vehicles

6. -FlagRTB: true= group of pilots will return to spawn location after paradrop. false=group of pilots will loiter in drop area

7. -FlagSmokes: true=paratroopers will drop smoke when approaching the ground.

8. Behavior: Group assigned will fly to actionloc and release all AI in their vehicle as 'cargo'. AI released will execute a halo type paradrop script. Driver's vehicle will spawn airborne.

ix. "PATROLROUTE"

1. syntax:["PATROLROUTE", [spawnloc], [actionloc], [behaviour, speed, [locations], FlagRTB, FlagRoads, FlagDespawn, flyheight]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: offset or specific map location at which behaviour will start.

4. -behavior: "CARELESS", "SAFE", "AWARE", "COMBAT", "STEALTH" - impacts AI's tendency to follow roads, and use lights.

5. -speed: "NORMAL", "FULL" or "LIMITED"

6. -[locations] : array of 2d locations, 3d locations, OR town names.

Keywords "Villages","Cities","Capitals" may also be used, in which case a random location of that type will be selected.

(ie ["Villages","Villages"] would select two random villages to patrol between)

7. -FlagRTB: true= group of pilots will cycle through all locations

8. -FlagRoads: true= group will attempt to use roads to get to the locations

9. -FlagDespawn: true= group will despawn upon reaching final location.

false= group will repeat route.

10. -flyheight: if 'air' vehicle, altitude to fly the route.

11. Behavior: group will patrol the locations specified. This logic is designed for longer distance, fixed location type patrols. If vehicle is of type 'air' it will spawn airborne.

x. "ZOMBIE"

1. syntax: ["ZOMBIE", [spawnloc], [actionloc],[Detection_range, FlagCanThrow, WonderRange]]

2. -spawnloc: offset or specific map location group will spawn

3. -actionloc: --no functionality-- leave as [0,0]

4. -options

a. Detection_range = distance in meters beyond which zombies will not identify/engage AI and players.

1. Zombies have an accute sense of vehicles and will detect players in vehicles if they come within this range regardless of Line of sight.

b. FlagCanThrow: true = zombies have a chance to move vehicles out of their way when pursing players.

c. Wonder_range = distance a target must get away from the zombie to lose it.

1. Players can lose the zombie in a range shorter than this, but it depends on their ability to block LOS/detection of the zombie to cause it to forget about the player(zombie types have different memories!)

5. Behavior: Full implementation of Ryan's Zombie and Demons AI.

a. Zombies will wonder around until they locate a player or non-side "WEST" AI.

b. Zombies will pursue until dead or distracted by a closer target.

c. Zombies will attack cars, boats, helicopters, and static weapons when they are occupied by players/AI.

d. Zombies will not attack unoccupied vehicles, but will throw them!

6. Type Zombies:

a. Zombie Jump 0%/10m ThrowCar 0% Memory 2sec

b. ZombieSoldier Jump 5%/20m ThrowCar 60% Memory 4sec

c. ZombieSpider Jump 50%/40m ThrowCar 5% Memory 8sec

d. ZombieBoss Jump 80%/35m ThrowCar 80% Memory 12sec

xi. "TRACKROUTE"

1. syntax:["TRACKROUTE", [spawnloc], [actionloc], [behaviour, speed, [locations], FlagRTB, FlagRoads, FlagDespawn, flyHeight]

2. - identical behavior to "PATROLROUTE", with addition of a symbol, appropriate to the AI's vehicle being continuously displayed at the AI's location.

xii. "TowerGuard"

1. Syntax:["TowerGuard", [spawnloc], [actionloc], [radius,"Building_Name"]]

2. -spawnloc: offset or specific map location <-- ignored for this logic, be retained for continuity of Logic series

3. -actionloc: offset or specific map location at which AI will be placed.

4. -radius: distance from 'actionloc' to search for rooftops

5. -"Building_Name": a building name may be provided. In this case, AI will only be placed on top of these buildings. Default is "ANY"

6. Behavior: Script will search 'AREA' for all enter-able buildings. If "Specific" building is provided, the list will be filtered to only buildings matching this name and the closest to the 'Origin' will be selected.

AI will be spawned at the top of these buildings and remain there.

xiii. "Gunner"

1. Syntax: ["Gunner", [spawnloc], [actionloc], [0]]

2. -spawnloc: offset or specific map location <-- ignored for this logic, be retained for continuity of Logic series

3. -actionloc: offset or specific map location at which AI will be placed.

4. -[0] - dummy value for now.

5. Behavior:

6. Used as a defined logic for 'drivers' in the Convoy section of missions. AI under this logic will man, and remain in static weapons.

xiv. "Captured"

1. Syntax: ["Captured", [spawnloc], [actionloc],[behaviorOption, [[rescue locations]]]

2. -spawnloc: offset or specific map locaiton

3. -actionloc: offset or specific map locaiton AI will move towards before starting default 'behaviour'

4. -behaviorOption

a. -1 = stay at action loc and do nothing

b. 0-99 = follow the group leader for mission group number X

5. [rescue locations]: array of offsets, location names, and map locations that when the AI get within 50m they are considered rescued.

6. Behavior:

a. Flee: If within 200m of a rescue spot captive will run to that location. Otherwise it selects a random rescue spot from the list provided. Unit sets CARELESS behavior and sprints to its destination.

b. Stay: Unit will stop at its location and assume the stance of the player issuing the order.

c. Follow: Unit will follow the player, mimicking his stance. If the player gets more than 250m away from the unit, the unit will resort to a 'Stay' behavior using the stance of the player at the time the unit loses him.

d. Board: Captive will move to the closest non-AI controlled, operational (non-zero dmg) vehicle within 200m and board it.

e. Evac Points: Temporarily shows the Captive's evac waypoints on the map. (30sec time out)

f. Everyone Out!: Directs all captives aboard a vehicle to exit the vehicle and follow the player.

g. Note: The captives are a little 'shell shocked' and will sometimes take a few seconds to respond to commands, but they will always acknowledge they heard the command via 'system chat'.

h. Note: Captives that start a mission as 'cargo' in an AI vehicle will remain in the vehicle until it is disabled.

i. Note: Captives assigned to follow an AI group will stop following that group when the group leader dies.

j. Note: Captives remain 'neutral' to AI until directed to Flee.

xv. Custom Variables attached to each AI

1. FuMS_AILOGIC = [patrolType, msnCenter, spawnLoc, patrolLoc, patrolpatternoptions]

2. FuMS_XFILL = [themeIndex, side, "xfillstatus"]

3. FuMS_MSNTAG = [theme, mission]

xvi. Logic inherrient to all AI created by FuMS.

1. *AI_Killed: RadioChat message on death. AI killed by vehicles or other AI have all their gear destroyed.

2. *AI_Evac: If unit is within the call radius (~200m's) of a unit conducting an XFILL operation, the unit will proceed to the XFILL'ing vehicle and board it (stopping all other 'patrol behavior's). At this point the unit will remain in the vehicle until the vehicle reaches its destination. At this point the unit will despawn. If the vehicle is disabled, or its driver killed, the unit will continue on foot to the vehicle's destination and then despawn.

3. *DriverLogic: If a unit in a driver seat detects its vehicle's position does not change by more than 2m for more than 150 seconds, the vehicle will be teleported to a nearby road segment, under the assumption it is stuck. If the vehicle conducting an "XFILL" this behavior is suspended while other units approach and board.

d. Trigger Definitions

Last updated 6/27/15

i. This section contains details on the various FuMS 'Triggers' that can be defined within each mission. FuMS triggers should not be confused with ARMA3 triggers. Though FuMS does use some ARMA3 triggers, many of the triggers within FuMS behave differently than a standard ARMA3 trigger.

ii. FuMS triggers, once established are checked every 5 seconds. By default all triggers are 'false' and will return true when the defined parameters of the trigger are met.

iii. ["Reinforce",chance,"<fileName>"] - only define in the WIN state!!

1. If this is defined in the WIN state, then when a unit in this mission calls for help, there is a 'chance' percent that the Theme's base operations will respond with reinforcements. If base ops does send in reinforcements it will occur through the execution of the mission file named <fileName>. <fileName> is a .sqf file and needs to be in the Theme's folder.

2. Note: The reinforcement file does not require the use of the ["NO TRIGGERS"] option, but if it is desired to not suspend the parent triggers, ensure that NO TRIGGERS is commented out of the actual reinforcement mission.

3. Reinforcement mission can be called multiple times!

4. Note: Reinforce trigger is not considered in the WIN conditions of the WIN state.

iv. Valid Triggers

1. ["LowUnitCount", side, numAI, radius, offset]

2. ["HighUnitCount", side, numAI, radius, ofset]

3. ["ProxPlayer",offset, range, numPlayers]

4. ["Detected", group#, vehicle#]

5. ["BodyCount", numAI]

6. ["Timer", seconds]

7. ["AllDeadorGone"]

8. ["Reinforce", chance, "mission"] <-- only use in Win State trigger area!

9. ["ZupaCapture", [[[offset], radius, time], [[offset], radius, time]]

10. ["DmgBuildings","INDEXER", amount]

11. ["DmgVehicles","INDEXER", amount]

12. ["OK"]

13. ["Captive", numAI]

14. ["Progressive", side, start, stop, increment]

v. ["LowUnitCount", side, numAI, radius, offset]

1. -side: "EAST","WEST","GUER","CIV","LOGIC","ANY","CAPTIVE"

2. -numAI: number of AI of type 'side' below which make the trigger TRUE.

3. -radius: distance the trigger searches when counting AI

a. '0' = trigger will look map wide.

4. -offset: centroid of the trigger. Supports 2D offset, or 3D map absolute positions.

5. Behavior: When the number of units on a 'side' drop below or equal to 'numAI' within 'radius' of 'offset' the trigger registers TRUE.

vi. ["HighUnitCount", side, numAI, radius, offset]

1. Behavior: When the number of units on a 'side' exceeds or equals 'numAI' within 'radius' of 'offset' the trigger registers TRUE.

vii. ["ProxPlayer",offset, range, numPlayers]

1. -offset: centroid of the trigger. Supports 2D offset, or 3D map absolute positions.

2. -range: range from 'offset' the count is conducted.

3. -numPlayers: the number of 'players' the number of players to count.

4. Behavior: When at least 'numPlayers' are detected within 'range' of 'offset' the trigger registers TRUE.

viii. ["Detected", groupIndexer, vehicleIndexer]

1. -groupIndexer : group that is performing the 'detection' check. See INDEXER below for specifics.

2. -vehicleIndexer: vehicle that is performing the 'detection' check. See INDEXER below for specifics.

3. Behavior: If the indicated group or vehicle is actively detecting a player the trigger will register TRUE.

4. A value of "NONE" will turn off the check for that type.

5. A value of "ALL" will run the check for all of that type.

6. Ex: ["Detected", "NONE","ALL"] will run the check for all vehicles, and no groups.

7. Ex: ["Detected", "2", "3-5"] will run the check for the 3rd group in the mission file, and the 4th through 6th vehicle created.

8. Note: Unoccupied vehicles count in this check if you are trying to determine a vehicle number in a 2nd convoy group.

9. Note: UAV, UAG's are listed as a vehicle, not a group.

ix. ["BodyCount", numAI]

1. -numAI: number of AI, regardless of side that are killed by players during mission execution.

2. Behaviour: If the number of AI killed during mission execution exceeds 'numAI' the trigger will register TRUE.

3. Note: AI killed by player vehicles will not count.

4. Note: AI killed by other AI WILL count towards the total.

x. ["Timer", seconds]

1. -seconds: time in seconds

2. Behavior: After 'seconds' of time elapse from mission start the trigger will register TRUE.

xi. ["ZupaCapture", [[[offset], radius, time,"name"], [[offset], radius, time,"name"]]

1. Behavior: establishes capture points of 'radius' size, at each of the 'offset's specified.

2. When a player remains within the radius for 'time' the point becomes 'captured'. Once all points are captured the trigger will return a value of true.

xii. ["DmgBuildings","INDEXER", amount]

1. "INDEXER" - type STRING " " (not an array [ ])

a. This is a zero based index on which buildings on which damage will be watched.

b. Valid formats may be a single building, range of buildings, or a combination of single buildings and ranges.

c. Examples:

1. "0" = will watch the 1st building created by the mission.

2. "1-5" = will watch the 2nd-6th buildings created by the mission.

3. "1,2,4-8" = will watch the 2nd, 3rd, and 5th-9th buildings created by the mission

4. "ALL" = entire list will be used.

2. amount

a. A value from 1.0 to 0.0

b. When damage value of the building reaches this amount it is 'counted' towards satisfying the trigger

3. Trigger will return true when ALL buildings in the "INDEXER" list are damaged to the 'amount' set in the trigger.

xiii. ["DmgVehicles","INDEXER", amount]

1. Identical behavior to "DmgBuildings" above.

2. Vehicles created in BOTH the building section and vehicle section are counted when trying to determine the proper 'indexer' values.

3. Example:

a. If you have two static vehicles in the building section it will be index ref 0 and 1

b. Any vehicles that are then created in the 'Vehicles' section would be index referenced starting with 2

c. IE you have a mission with 3 static vehicles in the building section and a patrol of 4 other vehicles.

1. Indexer = "1,5"

2. This would watch the 2nd static vehicle in the building section and the 3rd vehicle (6th total) in the "Vehicles" section.

xiv. ["OK"]

1. This trigger is always TRUE

2. Use of this trigger will result in the associated action occurring immediately upon mission start.

xv. ["CAPTIVE", numAI]

1. -numAI: number of AI, 'rescued' by players

2. Behaviour: If the number of AI rescued during mission execution equals (or greater) 'numAI' the trigger will register TRUE.

3. AI using 'captured' logic that is spawned in children missions will count toward the parent's Captive trigger goal.

****NOT IMPLEMENTED AT THIS TIME****

xvi. ["Progressive", offset, range, Num_start, Num_stop, incrementl]

1. -offset: centroid of the trigger. Value can be 2D position, 2D offset, or 3D map absolute position

2. -range: distance from offset the trigger is conducted.

3. -Num_start: number of players that will start to activate this trigger.

4. -Num_stop: number beyond which additional players will not activate this trigger.

5. -increment: increase in players past 'Num_start' that cause this trigger to activate again.

6. Behaviour: Activates similiar to "ProxPlayer". Once this tirgger activates off meeting the 'Num_start' requirement, it will fire again each time 'increment' is made. The trigger will continue to activate until 'Num_stop' is reached.

****NOT IMPLEMENTED AT THIS TIME****

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.