How to make an MSO Mission

This manual is a walk through on how to make your own MSO. In it, I will describe the creation of a Guerrilla's vs Russian Army MSO on Utes.

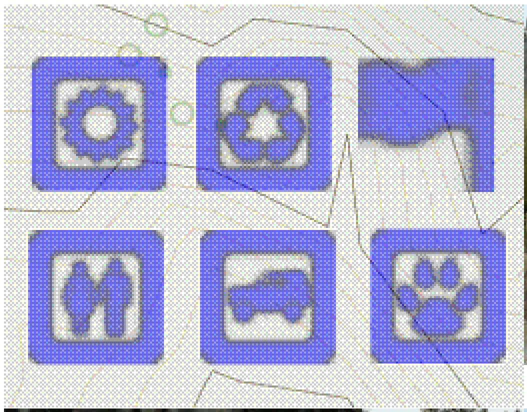
Table Of Contents

Mission Base - BIS Ambient Environment

This isn't necessary, it just adds ambiance to the mission. I usually put the following basics down in the Editor. In this example, I've used:

- BIS Functions - used in a lot of code, always good to put down first.
- CBA Required - used to ensure CBA is loaded up for the mission to load
- BIS Garbage Collector - will remove destroyed objects once you leave the area
- Ambient Civilians (Expansion) - the ALICE2 module (patched, see here)
- Ambient Vehicles - the SYLVIE module
- Ambient Animals (patched, see here)

This will populate the island and generate a little activity like people walking and cars driving.



Optional BIS Modules

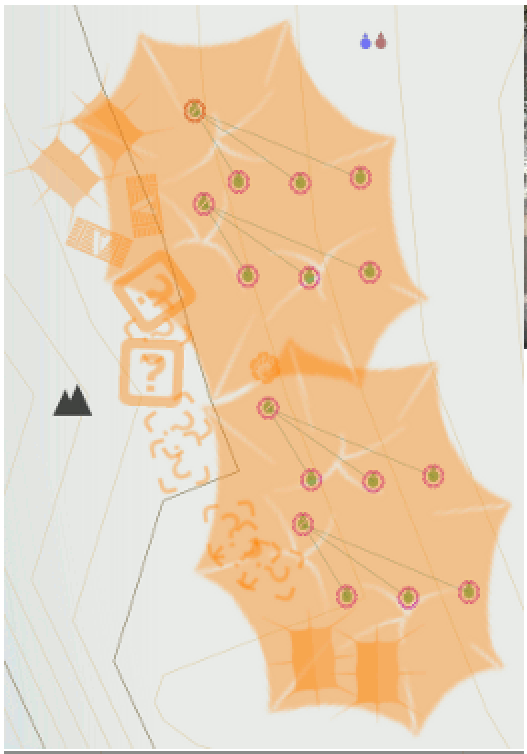
I usually add some other BIS modules that I like to include in my missions. They are:

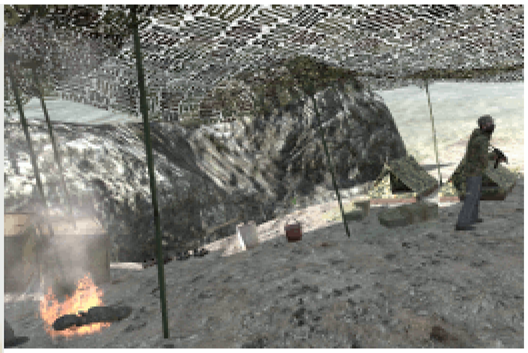
- Environment Colors - Module changes colors and brightness with ppEffectCreate depending on overcast and time of day. Colors will be desaturated during the night. (ARMA 2: Operation Arrowhead ONLY)
- Environment Effects - Module creates environmental effects such as dust, mist, snow and clouds. The effects depend on the time of the day and season (summer, winter etc). (ARMA 2: Operation Arrowhead ONLY)
- Weather - Creates clouds in the sky around the player.
- Military Symbols - Display units' military symbol in real time on map.
    - You need to add the following to the INIT of the Military Symbols module. I also like to put a 5 minute refresh.
      ```
      setGroupIconsVisible [true, false];
      BIS_marta_mainscope setvariable ["delay", 300]
      ```
- Surrendering - Makes AI soldiers surrendering when in confusion, or outnumbered.

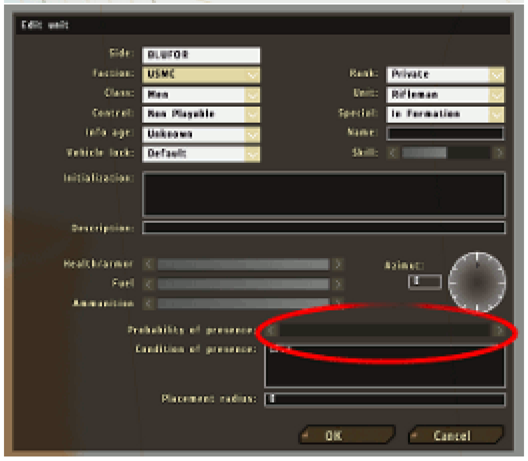
Mission Base - Friendly Forces

Ok, now the fun part - building your base. In my example, I just want a simple tent camp with 2 ammo caches and a fire. I've thrown in some aesthetics, but nothing else very functional.

At this stage another important thing to do, is initialise sides. You do this as some modules require the AI to be setup for them to work. I do this by dropping a unit for each of the sides I want to initialise (usually whatever side I've not yet put on the map) and set their Probability of Presence to 0%. This initialises SetFriend but doesn't place a unit on the map. You'll see the two units I've put down in the top right of the Camp setup graphic below.







Edit unit

Side:	BLUFOR		
Faction:	USMC	Rank:	Private
Class:	Men	Unit:	Rifleman
Control:	Non Playable	Special:	In Formation
Info age:	Unknown	Name:	
Vehicle lock:	Default	Skill:	

Initialization:

Description:

Health/armor:

Fuel:

Ammunition:

Azimut: 0

Probability of presence:

Condition of presence:

Placement radius: 0

OK Cancel

Don't forget to:

- Put down a respawn marker (respawn_guerrila in my example)
- An object that the players will use with the Log Book module later on

At this point, run your mission, ensure everything is working fine. We'll begin coding next.

MSO Foundation Code

Below is some common code used through the MSO modules. This forms the foundation of all the MSO modules you'll later insert.

Copy the following files and directories into your mission folder.

`<dialogs>`

```
<functions>
<modules>
<scripts>
description-mso.ext
init-mso.sqf
```

If you haven't already created a DESCRIPTION.EXT file, create one now and put the following in it.

```
#include <description-MSO.ext
```

If you haven't already created an INIT.SQF file, create one now and put the following in it.

```
[] call compile preprocessFileLineNumbers "Init-MSO.sqf";
```

In your INIT.SQF file, select the enemy factions you wish to spawn as part of the Convoys, EnemyPop, and ZORA modules by adding the following:

```
MSO_FACTIONS =
["RU", "INS", "GUE", "BIS_TK","BIS_TK_INS","BIS_TK_GUE","PMC_BAF"];
```

The factions can be any selection you desire.

Team Leaders have access to certain functions such as JIP Tasks. Team Leaders are chosen from the MP Rights module and **mso_uids.txt** file.

I'll explain each of the default modules and what they do.

Support Modules

RMM NOMAD - Persistent Player State

This should be the first module you initialise in your INIT.SQF after the defaut INIT-MSO.SQF.

All you need to create your own MSO mission is the NOMAD system. The NOMAD system is responsible for saving the state of your player upon disconnection, including such things as position, health, weapons and ammunition. NOMAD allows you to disconnect and reconnect at any time, resuming your player's status as it was when you disconnected. This is the heart of the Multi-Session Operation, allowing players to continue their missions for days instead of hours.

The components below are the mandatory requirements to implement an MSO mission.

- BIS Functions module
- Playable Unit
- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT

```
class Params {
   #include <modules\nomad\params.hpp>
};
```
- INIT.SQF
  ```
  execNow "modules\nomad\main.sqf";
  ```
- Copy the following files:
  ```
  modules\nomad\*.*
  ```

RMM Player Revive

This is the simplified Revive System. Revive automatically applies to all playable units.

- BIS Functions module
- Playable Unit
- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\revive\functions.hpp>
  };
  ```
- INIT.SQF
  ```
  waitUntil{!isnil "revive_fnc_init"};
  if (!isDedicated) then {
     player call revive_fnc_init;
  };
  ```
- Copy the following files:
  ```
  modules\revive\*.*
  ```

RMM View Distance Settings

If you wish to add the RMM View Distance Settings to your mission, add the following folder/ scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- To add the View Distance settings to the Communications Menu, use the following:
  ```
  ["Settings","createDialog ""RMM_ui_settings"""] call _fnc_updateMenu;
  ```
- DESCRIPTION.EXT
  ```
  #include <modules\settings\dialog.hpp>
  ```
- Copy the following files:
  ```
  modules\settings\*.*
  ```

Usage

To change your View Distance, select the Communications Menu (0-8) and select Settings. Change your view distance and terrain detail, and then press ESC to escape.

RMM After Action Reports

Usage
After every incursion or event, you should add an After Action Report. Its best to submit them ASAP, so you don't forget and it helps keep every other player up to date on whats happening. All players are able to add AARs, which are displayed in the Map screen under Notes. To submit an AAR, select the Communications Menu (0-8) and select AAR. Fill in the details and click Submit to post.
Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- To add the View Distance settings to the Communications Menu, use the following:
  ```
  ["AAR","createDialog ""RMM_ui_aar"""] call _fnc_updateMenu;
  ```
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\aar\functions.hpp>
  };
  #include <modules\aar\dialog.hpp>
  ```
- INIT.SQF
  ```
  execNow "modules\aar\main.sqf";
  ```
- Copy the following files:
  ```
  modules\aar\*.*
  ```

RMM JIP Markers

Usage
Any enemy contacts should be marked with a JIP Marker ASAP and it helps keep every other player up to date on what's happening in the battlefield. All players are able to add JIP Markers, which are displayed in the Map screen. To create a JIP Marker, ALT+Click the map, where you can enter the marker type and text description. To delete a JIP Marker, ALT+Click near the marker, and select Delete Nearest from the dialog.
Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\jipmarkers\functions.hpp>
  };
  #include <modules\jipmarkers\dialog.hpp>
  ```
- INIT.SQF
  ```
  execNow "modules\jipmarkers\main.sqf";
  _crb_mapclick = _crb_mapclick + "if (!_shift && _alt) then
  {RMM_jipmarkers_position = _pos;
  createDialog ""RMM_ui_jipmarkers"";};";
  onMapSingleClick _crb_mapclick;
  ```
- Copy the following files:
  ```
  modules\jipmarkers\*.*
  ```

RMM Logbook

Usage

The Logbook facility can be added to any object you place on the map. This can be Notice boards, Notebooks, Laptops, or Satellite Phones. To use the Logbook, simply selection the action on the object, add your text and press submit.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\logbook\functions.hpp>
  };
  #include <modules\logbook\dialog.hpp>
  ```
- Copy the following files:
  ```
  modules\logbook\*.*
  ```
- To add the Logbook to an object in the Editor, use the following:
  ```
  0 = this execVM "modules\logbook\addAction.sqf";
  ```

R3F Logistics

R3F Logisitics provides the ability to carry, tow, air lift, store and view contents of vehicles and aircraft. It can be used to transport and position building materials to build Forward Operating Bases (FOBs) and transportation of damaged vehicles back to base.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  #include "modules\R3F_Logistics\desc_include.h"
  class RscTitles {
     #include "modules\R3F_Logistics\desc_rsct_include.h"
  };
  ```
- INIT.SQF
  ```
  execNow "modules\R3F_Logistics\init.sqf";
  ```
- Copy the following files:
  ```
  modules\R3F_Logistics\*.*
  ```

Vehicles

All vehicles are automatically added to the R3F Logistics system from the call in INIT.SQF.

Usage

Approach a vehicle and a selection of actions in yellow should activate, allowing you to carry, tow, lift, store and view contents per vehicle.

RMM Recruitment and Team Status

The Recruitment facility consists of two sections – AI recruitment and Dr Eyeball's Team Status dialog. The Team Status will allow you to dynamically create, join and leave groups. This is useful when recruiting AI to main static defences and later creating a new group to leave them autonomous.

The AI Recruitment dialog allows Team Leaders up to 6 AI units to join their group. This is handy when defending bases while players are disconnected.

Usage

The Recruitment facility can be added to any object you place on the map. This could be a HQ or Barracks object. The Recruitment module simply add the Recruitment and Team Status actions to the object.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  #include <modules\recruitment\dialog_baf.hpp>
  #include <modules\recruitment\dialog_us.hpp>
  #include <modules\recruitment\TeamStatusDialog\TeamStatusDialog.hpp>
  ```
- STRINGTABLE.CSV
  ```
  Language,English,German
  #include "modules\recruitment\TeamStatusDialog\stringtable.csv"
  ```
- Copy the following files:
  ```
  modules\recruitment\*.*
  ```
- To add the Recruitment options to an object in the Editor, use the following:
  ```
  0 = this execVM "modules\recruitment\addAction.sqf";
  ```

RMM JIP Tasks

Usage

If an MSO CO would like all players to focus on specific tasks, these should be created here. Only Team Leaders can add JIP Tasks, which are displayed in the Map screen. To create a JIP Task, SHIFT+ALT+Click the map, where you can enter the task title and text description. To delete a JIP Tasks, SHIFT+ALT+Click near the marker, and select Delete Nearest from the dialog.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\tasks\functions.hpp>
  };
  #include <modules\tasks\dialog.hpp>
  ```
- INIT.SQF
  ```
  execNow "modules\tasks\main.sqf";
  if (MSO_R_Leader) then {
  ```

```
   _crb_mapclick = _crb_mapclick + "if (_shift && _alt) then
{RMM_task_position = _pos; createDialog ""RMM_ui_tasks"";};";
   onMapSingleClick _crb_mapclick;
};
```
- Copy the following files:
  ```
  modules\tasks\*.*
  ```

CRB Flippable Vehicles

Usage

Any motorcycles or ATV will have a Flip Vehicle action added to them, to ensure they can be up-righted when rolled.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
     #include <modules\flippable\functions.hpp>
  };
  ```
- INIT.SQF
  ```
  execNow "modules\flippable\main.sqf";
  ```
- Copy the following files:
  ```
  modules\flippable\*.*
  ```

RMM Tyre Changing System

If you wish to add the RMM Tyre Changing System to your mission, add the following folder/ scripts:

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
      #include <modules\tyres\functions.hpp>
  };
  ```
- INIT.SQF
  ```
  execNow "modules\tyres\main.sqf";
  ```
- Copy the following files:
  ```
  modules\tyres\*.*
  ```

Vehicles

All wheeled vehicles are automatically added to the Change Tyres system from the call in INIT.SQF.

Spare Tyres

You will need spare tyres as resupplied objects if you want use this system. See Object Resupply below.

RMM Debug

Usage

To use the Debug console, you are first required to be an MSO admin (hardcoded in initplayer.sqf). You then select the Communications Menu (0-8) and select Debug. The console allows you to execute code on the server, clients or both.

Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  #include <modules\debug\dialog.hpp>
  ```
- INIT.SQF
  ```
  if (MSO_R_Admin) then {
     ["Debug","createDialog ""RMM_ui_debug"""] call _fnc_updateMenu;
  };
  ```
- Copy the following files:
  ```
  modules\debug\*.*
  ```

Ambience Modules

CRB Ambient Civilians

Usage

All towns are automatically populated with ambient civilians, civilian traffic, civilian vehicles and animals by using the appropriate BIS Editor modules in your mission. The Civilians module tweaks the settings as shown below.I find the default settings of the Ambient Civilians and Ambient Vehicles modules are too intense, and usually apply the following settings to minimise objects and increase performance.

```
// For ALICE2 module
// See http://community.bistudio.com/wiki/Ambient_Civilians
//
// Increase spawn distance for ALICE2 traffic
BIS_alice_mainscope setvariable ["trafficDistance",1000, true ];
// Reduce unit count formula to try to reduce number of civilian units
BIS_alice_mainscope setvariable ["civilianCount","round (2 * (sqrt %1))", true ];

// Dumb down civilian units to use less CPU (see http://creobellum.org/node/175)
[BIS_alice_mainscope,"ALICE_civilianinit",[{_this setSkill 0},{{_this disableAI _x} count
["AUTOTARGET","TARGET"]},{_this allowFleeing 1;},{removeAllWeapons _this;},{removeAllItems _this;}]] call
BIS_fnc_variableSpaceAdd;

// Artificial coeficient to set how much will be town's respect decreased once some civilian is hit or
killed.
// The higher the number is, the more is respect towards killer's faction decreased.
BIS_alice_mainscope setvariable ["respectModifyCoef", 0.7, true ];

// Value which is removed from town threat every 5 seconds (until threat reaches 0)
```

```
BIS_alice_mainscope setvariable ["threatDecay", 0.000005, true ];

// For SYLVIE Module
// http://community.bistudio.com/wiki/Ambient_Civilian_Vehicles
//
// Reduce vehicle count formula to try to reduce number of civilian vehicles
BIS_silvie_mainscope setvariable ["vehicleCount","round ((sqrt %1) * 1.0)", true ];

// Randomly lock and vary fuel
BIS_silvie_mainscope setvariable ["vehicleInit",{if (random 1>0.2) then {_this loc k true } els e {_this
setfuel (random 1)}}, true ];
```

To automatically add these settings, include the Civilians module, add the following folder/ scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- INIT.SQF
  `execNow "modules\civilians\main.sqf";`
- Copy the following files:
  `modules\civilians\*.*`

RMM Call To Prayer

Usage

Enables all minarets on the map, to emit a call to prayer at various times of the day.
Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class CfgSounds {
     #include <modules\ctp\cfgsounds.cpp>
  };
  ```
- INIT.SQF
  `execNow "modules\ctp\main.sqf";`
- Copy the following files:
  `modules\ctp\*.*`

CRB Wild Dogs

This module was originally written by Big Daddy as Blitzy the Guard Dog. This module randomly places a trigger of wild dogs who will attack any WEST units within 100m. If you wish to change the SIDE that the dog will attack, use "WEST","EAST" or "GUER" as the first parameter.
Add the following folder/scripts:

- Default MSO initialisation Init-MSQ.sqf and Description-MSO.ext
- DESCRIPTION.EXT
  ```
  class cfgFunctions {
      #include <modules\dogs\functions.hpp>
  };
  ```

- INIT.SQF
  `["GUER"] execNow "modules\dogs\main.sqf";`
- Copy the following files:
  `modules\dogs\*.*`

Usage

If you wish to add a patrol dog to your Friendly units, put the following in the leader unit's INIT field:

```
0 = [this] spawn dogs_fnc_blitzy;
```

Re-Supply

There are 2 types of re-supply:

- static re-supply (building materials, ammo crates, etc) using **resupply.sqf** whereby items are respawned once they have been move 10m from their original location, and
- vehicle re-supply (vehicles, artillery pieces) using **vehicle.sqf** whereby, once the item is destroyed will it be respawned.

Static Re-Supply

Backpacks Resupply

I use the following to replace Backpacks are they are being taken. The backpacks are replaced in 60sec once they have been moved away from the area. Add the following to their INIT field.

```
0 = [this, 60] execVM "scripts\resupply.sqf";
```

Ammunition Resupply

Ammunition crates are replaced every 4hrs. Place the following in their INIT field.

```
0 = [this, 14400] execVM "scripts\resupply.sqf";
```

Building Materials and Static Weapons Resupply

Most building materials are replaced every 8hrs. Place the following in their INIT field.

```
0 = [this, 28800] execVM "scripts\resupply.sqf";
```

Vehicle Re-Supply

Support and Attack Vehicles Resupply

Below are settings that will respawn a support or attack vehicle only if its been destroyed. The vehicle will be resupplied in 24hrs. Place the following in the vehicles INIT field.

```
0 = [this, 86400, 9999999] execVM "scripts\vehicle.sqf";
```

Transport Vehicles Resupply

Below are settings that will respawn a transport vehicle only if its been destroyed. The vehicle will be resupplied in 12hrs. Place the following in the vehicles INIT field.

```
0 = [this, 43200, 9999999] execVM "scripts\vehicle.sqf";
```

Miscellaneous

Watchtower Soldiers

When setting up a base, placing soldiers in watchtowers is quite effective. To place a soldier, position them close to the tower, and add the following to their INIT field.

```
0 = [this] execVM "crB_scripts\crB_HousePos.sqf";
```

Notice Board (Logbook)

I usually place a notice board at the front of the base and add the following to the INIT field.

```
0 = this execVM "modules\logbook\addAction.sqf";
```

Recruitment and Team Status

I usually place a HQ module down and add the following to the INIT field.

```
0 = this execVM "modules\recruitment\addAction.sqf";
```

Street Lamps

Drop a game logic object down and place the following in the INIT field to create a lamp.

```
("Land_Lampa_Ind_EP1" createVehicleLocal getpos this) setdir ((getdir this)
+90);
```

Facilities

Its advisable to place a Field Hospital and Vehicle Service Point at your base. See the mission for what building supplies, weaponry and other respawnable objects have been used.