DSMC

MISSION DESIGNER MANUAL - Version 1.3.1

BORING AND COMPLEX: A NECESSARY EVIL TO MISSION DESIGNER

This manual is here to provide additional information, instruction and "suggested design procedures" made with mission designers in mind.

This is done to address and answer some questions, for example:

- Why you need DSMC
- Install DSMC
- Single player/host and Server/Dedicated server mode
- What exactly is DSMC?
- How does it work?
- What are its limitations?
- How should I setup my host/server to use it?
- How does it relate to mission design?

We hope to answer all those questions in the next pages. Here you're in front of your trusted mechanic to learn how your car works and how to maintain her as it should. Let's start.

PS: read chapter 5 before anything else

TABLE OF CONTENTS

1   WHAT IS DSMC?

Dynamic **S**equential **M**ission **C**ampaign (DSMC) is a tool for mission designers, supporting versions of DCS World 2.8 and above[1]. This tool adds persistence to any DCS World mission, done by allowing the user to save the scenario at any moment and generate a new mission file based on the situation at the time of saving, that can be loaded, or edited, using the DCS mission editor.

DSMC is required to be installed on the host/server only, and has configuration settings available for both Dedicated servers, or Player as Host, via the special options GUI in DCS World.

DSMC creates a new *.miz* file that includes:

- All original triggers, scripts and embedded files as the original
- Updated ground units & ships[2] positions and states (alive, dead)
- Scenery object states, like bridges, houses and structures
- User static object changing coalition
- Updated airbases, Oil/Gas facilities & FARP ownership
- Updated warehouse contents of fuel, aircraft & ammo
- Updated mission start time
- Updated weather
- All the units & object that has been spawned during the mission (including CTLD FOBs![3])

Also, it will provide as optional features:

- Creation of "dead" static objects to reflect scenery wreckage after previous battles
- Automated creation of helicopters & planes flights (slots) in heliports & airbases
- *Many more things… check customization chapter!*

---

[1] Check version 1.1 or 1.0.8 for DCS 2.5.6 and before. Version 1.2.6 should work ok for DCS 2.7.
[2] Except for aircraft carriers due to DCS limitation
[3] CTLD fob specific features like being able to spawn crates or load troops won't be kept, cause it's something up to the ctld code. But check DSMC's CTLD version to provide further enhancement like this and more. See DSMC's discord, dsmc-automated-ctld channel.

DSMC also includes improved features and functions for server mode.

DSMC does not support capturing aircraft or missiles in flight and resume a mission later: it's done to create updated scenarios, not to freeze a mission and resume later.

2 WHY DSMC?

First of all, it's D.S.M.C.: The acronym means "Dynamic Sequential Mission Campaign"... and to be *really* honest, today it's much more "SMC" than "DSMC". "D" is left out for now: while the final objective is to create a dynamic campaign system, the first strong step is to achieve what DCS doesn't give us: **persistency**.

That is the milestone we're setting now with DSMC: to give the mission designer a tool to reduce the workload to update a mission scenario after a flight with 30 clients going here and there, destroying units, breaking down bridges, moving resources, etc. etc.... and also updating the position of every ground and sea units out there, with meter precision, tracking every single object spawned in, like units, structures, crates, etc etc.

Ok, so, why? Because some years ago I was a mission designer and I really waited a long time some sort of scripting messiah that creates this thing for me: that will free my soul of hours of works to move those f*****g Urals from one town to another, that tank battalion from those unpronounceable villages to that other, even less pronounceable, place (Caucasus was the main solution at that time).

Well, few years ago, almost 2014, I understood that no Speed, Grimes, Wunderwulf, XCom or Ciribob would do that for me... so I tried to create it on my own. Obviously annoying them a lot in the process, but at least I tried to solve my issues myself.

As I am not a programmer, it required me to learn three times over. My first attempt was named "DAWS", and even if badly coded, it hit some small targets, you know, even today there are people who call this mod DAWS instead of DSMC out of habit. And here we are.

I'm asking you only one thing, and I hope that you will stick to that before even thinking a question: RTFM. *Read The Fucking Manual.* It's a necessary pill to your sickness of trying to use this mod. PS: Don't get me wrong, we both know that if you're a DCS mission designer, you're sick as much as I am.

3 INSTALLATION

DSMC needs to be installed only in the machine that runs the mission (host or server) and it's the same thing for any use. It works with both DCS stable & openbeta versions. The following example references the DCS World Open Beta version.

Installation can be performed either in "manual" mode and in "mod manager" mode. **I strongly suggest the "mod manager" mode**, because it will be easier to check install errors and will allow you to remove the mode much easily when needed.

DSMC mod is provided as a .zip file that contains the mod folder named "DSMC_x.y.z", where x, y and z are the version numbers.

3.1 Manual install

To manually install DSMC you can copy "as is" <u>the content</u> of the mod folder (not the mod folder) right inside your Saved Games\DCS.openbeta\ folder.

The mod files are:

- DSMC folder
- Script\Hooks\DSMC_hooks.lua *file*
- Mods\tech\DSMC\ *folder*
- DSMC_Dedicated_Server_options.lua *file*

When you manually install the mod, to remove it you will necessarily "search & destroy" each of these folders/files.

3.2 Mod manager install

Two mod manager can be used with DSMC (about the others, I don't know): OvGME and JGSME.

Using OvGME, the .zip archive is already "ready-to-use": you only have to move it in your mods folder that aim to the Saved Games\DCS.openbeta\ folder.

Using JGSME, you have to unzip the archive content (not the archive itself!) inside you mods folder that aim to the Saved Games\DCS.openbeta\ folder.

3.3 Check a proper install

Check if an install procedure is correct is easy: you only need to check if the DSMC icon is available in main menù:

3.4 What is my version?

I recently added a simple file inside the package called DSMC_version.txt. To be sure you're using the last version, check there for both version numbers and the built number, comparing them with the last updates reported in the DSMC discord updates channel.

3.5 Additional check necessary to ensure DSMC to work

This mod doesn't require many operations, but to work properly it needs to be allowed to *create* the saved mission file obviously. But to prevent malicious software to do "bad" things, windows adds user control. That means that you have to ensure DCS has sufficient authorization to *write* the file. You need to do three things:

1. Run DCS as administrator.
2. Remove writing permission from the Saved Games\DCS.*yourversion*\ folder, by right-clicking on the DCS.*yourversion* folder and removing the check to the "Read-only" checkbox.
3. Remove writing permission from the DCS install folder (i.e. C:\Program Files\Eagle Dynamics\DCS.*yourversion,* same procedure as point 2.
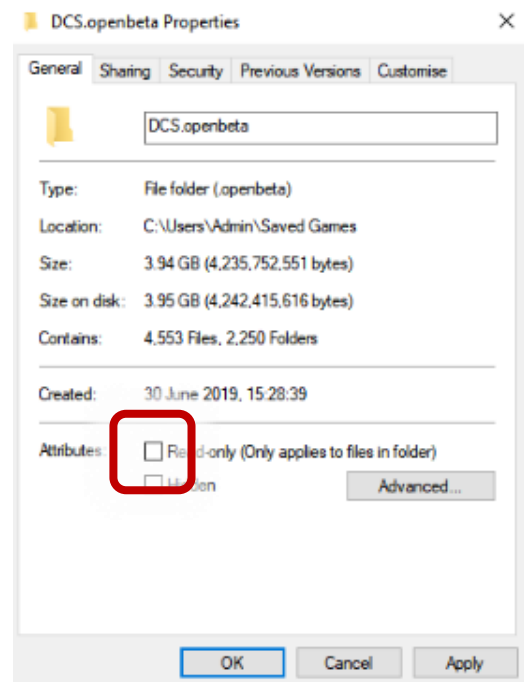
4   COMPATIBILITY & OTHER SCRIPS

If you are a proficient coder and have experience in mission design, DSMC is designed to be fully compatible with:

- Mist
- MOOSE
- LoATC
- CTLD
- CSAR script
- Gamemaster

Using these scripts combined with DSMC **require the user to be a proficient in coding** cause the code in most DCS scripts is not designed to work in a persistent environment: you will need to rework the way you implement them into your mission. Please read the following note and chapter 10 to be sure to make everything works well.

If you have doubt or you're a newbie in scripting, **please consider avoiding use any script in conjunction with DSMC**.

Instead, DSMC is **not** compatible with:

- SLmod
- Some unofficial aircraft modules (A4, Grimpen, etc)
- Any mods/scripts that add its own persistency system;
- Any other mods/scripts that change missionscripting.lua;

Many other mods have been reported to be compatible with DSMC and, as a general information, all other scripts that use scripting engine should work correctly once unless messing up with *missionscripting.lua*.

That said, there's a thing that every mission designer must understand about scripting compatibilities: DSMC use extensively the DCS scripting engine (SSE) to gather all the necessary information for its work. **Those function can be modified or broken by other script due to wrong or imprecise use** by the

mission designer or, else, because those scripts are not designed to be "save proof" for a second launch. **Those errors can affect DSMC preventing it work**.

For example, one of the most common errors by mission designer is using a script that point to a specific group name… the first mission it will work, but eventually this group will be killed. After that, once you launch the saved mission, the script won't find that group anymore because it has been killed… and, unless the script is written properly, it will cause an error capable of halting the mission or crash DCS.

Be sure to read "mission design guidelines" chapter about this.

5 BUG REPORT

DSMC is usually available to some testers from weeks to months before each release. This fact allows me to say two things:

If there are bugs, *complain them also*. Ok, you maybe never going to know their name, but at least now you know that they exist, and for sure they let *that* bug intentionally there only for annoying you today.

Thanks to them, and one in particular[4] which really takes fun in running DSMC in any (f*****g) kind of situation and beyond any possible scenario I could imagine, I was able to identify some very useful mission design guidelines.

But perfection is impossible, so bugs exist and I'm doing my best to remove them as fast as possible once they're recognized.

<div align="center">

→ ONCE ←

→ THEY ←

→ ARE ←

→ RECOGNIZED ←

</div>

Yeah, that's the point. Bugs are removed by checking in details the specific "wrong" behaviour or issue that happens, tracking what caused the issue and fixing it. The first part of the process is the most important, and it's called to **reproduce** the bug. As you can easily figure out, if I can't reproduce your issue, I can't find the problem and therefore I can't fix it. This article is interesting in that matter.

I don't know average statistics and as I'm not a programmer, I can't say nothing about the world outside this tiny mod. But I can tell about what usually happen here… **From the moment I can reproduce the bug, the fix time is between minutes to few days**. No more. Else, it can take months, literally. Cause maybe that stupid tiny bug

---

[4] If you don't know who I'm talking about, doesn't matter. But if you're a tester and think that I'm talking to you, well, it's you.

happens only if you have certain modules active, certain DSMC feature, in multiplayer environment and when the temperature in Miami is -5°C (spoiler alert: not so frequently).

Ok, so, what's the solution? Easy: you provide information to me about what was happening <u>exactly so that I can go straight to the bug</u>. How? Not by a phone call for sure… I'm asking only a very small effort, that anyway it's **very very important**.

This is the required infos:

- Send "logs". Logs file are **dcs.log**, **dcs.log.old**, **DSMC.log**, **DSMC_old.log**. You can find them in your Saved Games\DCS.*whateverversion*\Logs\ folder;
- Send the miz file you're using;
- Then, once you collected those 5 files, join DSMC discord channel and briefly tell me what triggered the bug from your point of view in the discussion-bugs channel:
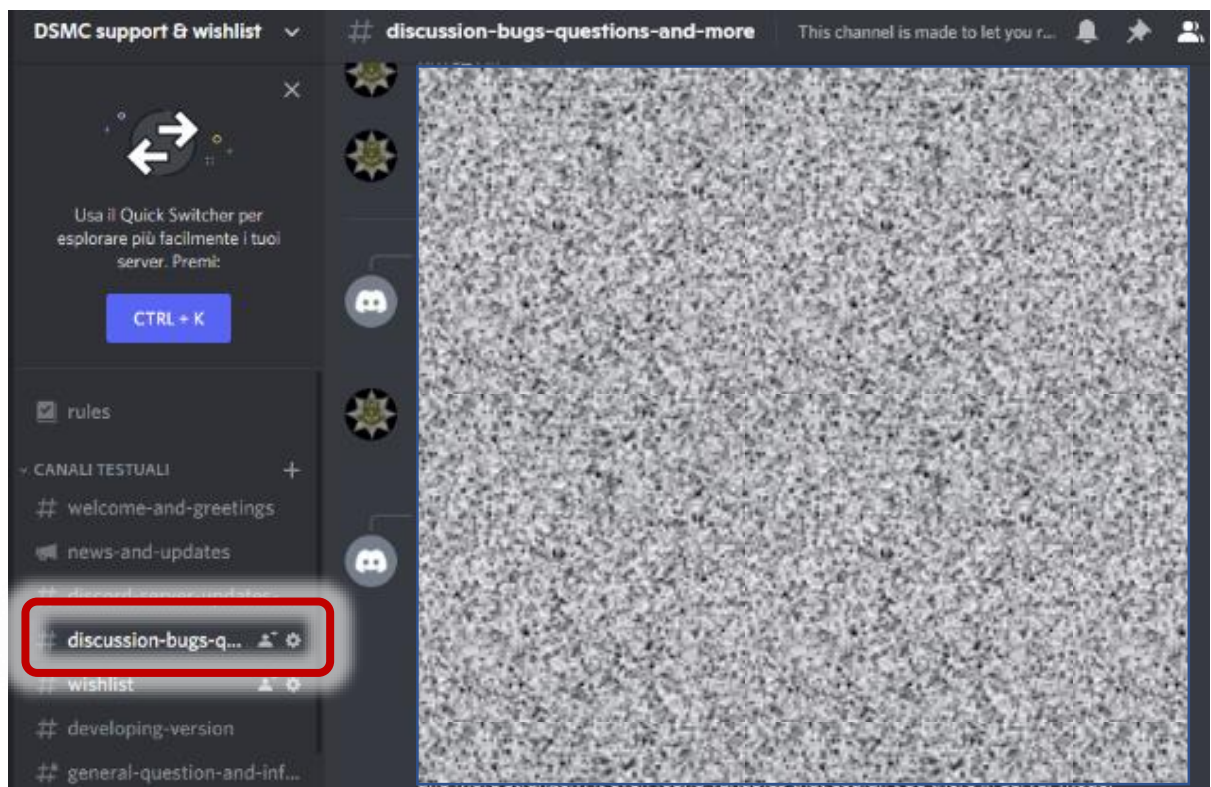
One last thing, I'm telling this to spare a lot of your time… and mine also. DSMC use extensively DCS native scripting functions, that

allow this mod to be very very resistant to the updates because ED try every time to do not mess up its scripting engine (SSE).

Sadly, many users that employ additional scripting frameworks such as mist or MOOSE are not trained to check if their implementation cause errors in the SSE (you can check reading dcs.log)... on the contrary, there's a frequent use of a "silencing" functions that prevent DCS to show the error if happens. That is not good.

Once an error happens, that SSE function is broken and won't work anymore in that mission run… and sometimes, nothing happens player-side, everything *seems* to work perfectly (it is not). If that functions are used by DSMC, it will break it. **But it's not a bug**. It's an error generated by other scripts, that needs to be addressed by the mission designer in its scripts. So please, before reporting a bug, check your scripts behaviour. The easiest things to do, when possible, is to try again with DSMC with those scripts disabled. If everything's works fine, There's something to be fixed your side. In that case I can provide help anyway in DSMC discord channel by checking the logs and telling you where the issue happens. This is happening at least 8 times out of 10 bug reports, it's perfectly ok.

6 DSMC USE: SP/HOST VS SERVER MODES

DCS can be used in single-player (SP) or in multy-player. Multy-player also can be done in different ways: you can simply host a mission, you can host with some "special" additions like "--norender", or you can use the *dedicated server* install provided by Eagle Dynamics which has to "graphics". Basically, in one version you have the GUI and the options, menù, in the other you don't.

The thing is that DSMC handle this two DCS way of work differently: if you have the GUI and the options menù, DSMC provide a "Special options" menù like any other modules and you can set your preference & customization there.

Else, if you run a version without graphics like dedicated server or "—norender" executables, then DSMC won't be able to read the options menù and it must be configured using a specific customization file that you will find right into your \Saved Games\DCS.whateverversion\ folder: *DSMC_Dedicated_Server_options.lua*.

6.1 Single player & host with graphics

This is the "basic" DSMC mode. It will require no additional modification and allows DSMC to be customized and configured using the special options menu, in the DCS World user interface.

This mode should be used for single players and multiplayer hosts who run the simulation without using a dedicated server. It relies on the 3D interface for configuration.

To do a quick test, follow those steps:

1. Go into the DCS mission editor
2. Open a mission
3. Save the mission renaming with "DSMC_*missionname*.miz". DSMC will only work if the first letters of the mission name are "**DSMC**", case sensitive
4. Load the mission, from the editor or from the main menu

5. Enter a client unit or, with Combined Arms, into a tactical commander or game master slot
6. You will see an additional option using F10 radio menu. If you choose "*DSMC*" submenu and the "Save scenery" it will start a sequence of trigger messages and it will print "*scenery saved!*" in some seconds (or, if you have a desanitized environment, it will simply print the "scenery saved!" message after a while).

You can now exit the mission, go to the default DCS mission folder (\Saved Games\DCS.*whateverversion*\Missions) and open the and you will see a new mission named "DSMC_*missionname*_**001.miz**".

Done!

Additionally, in multiplayer environment DSMC monitors the connected player counts. DSMC will perform an automatic save procedure by itself when the server is empty or only the host is online!

BEWARE: the save process is complex and heavy, **it might takes minutes for complex missions**. Do not run it continuously, or while clients are connected, in order to avoid momentary server disruption!

6.2   Servers: dedicated server & host with "--norender"

This mode is made with server administrators in mind, and allows some additional features & options such as:

- Extremely light auto-save process every "n" minutes (1 to 480)
- No screen information messages during the save process
- Customizable setup of the variables that can shape the behaviour of a server designed to work on a 24/7 environment

By design, the configuration of this mode is done using the "**DSMC_Dedicated_Server_options.lua**" file instead of the GUI special options menu. ".lua" are basically text files, but to be edited properly you'll need to use a very common app named "notepad++":

https://notepad-plus-plus.org/downloads/

In 'dedicated server mode' DSMC will perform an automatic full save every time the last client leaves the server.

DSMC also attempts to save when the mission is stopped by loading another mission or by closing DCS to windows. In case of "killing" the DCS process to perform a full restart of the server (not required), you must ensure that a 'graceful' process exit will be performed to ensure DSMC to save the required files.

To help server admin that want to run 24h servers, given that DCS and DSMC don't like that much the windows process killing procedure, DSMC provide:

1) An automatic "load saved mission" feature that will create an endless cycle like "flying the mission, saving the update, loading the updated mission";
2) If you really prefer to close DCS to desktop, then a "graceful" exit option is provided by an automatic close to desktop command that can be set every "n" hours, from 1 to 24: i.e., If set with 6 hours, after 6 hours of simulation DCS will try to close it to desktop. Try… why? Cause if there are clients connected, it will wait till the last one is out, checking every 5 minutes.

BEWARE: DSMC does not support missions' lists.

DSMC will continuously increment the next mission to run in Dedicated Server mode. DSMC will create an additional duplicate of the saved file named DSMC_ServerReload_*nnn*.miz, where "nnn" is a progressive number beginning with 001. **If the mission list is made only by 1 mission, the DSMC one**, the saved file file will be automatically set as "first mission to run" when the dedicated server is launched or it will be directly loaded after the run time of the previous mission. Server Administrators can leverage this mode to have 'hands-off' continuous persistence.

6.3   Scenery save activation matrix

As a reminder, DSMC can perform two different save procedure:

- Full save procedure
- Autosave procedure

Full save procedure it's a heavy process that must be avoided when clients are *doing things*. Because of that, full save happens only in specific moments:

- When the last client disconnects from a server;
- When a client uses the F10 menu options (manual activation).

This matrix will try to sum up all the DSMC actions with different use:

		Autosave option disabled	Autosave option enabled
DCS execution mode	DCS exe (clean): single player	✓ F10 manual activation	✓ F10 manual activation ✓ *Autosave*
	DCS exe (clean) multiplayer host	✓ F10 manual activation ✓ Last client disconnects	✓ F10 manual activation ✓ Last client disconnects ✓ *Autosave*
	DCS exe with "--server" or "--norender"	✓ F10 manual activation ✓ Last client disconnects	✓ F10 manual activation ✓ Last client disconnects ✓ *Autosave*
	Dedicated server executable	✓ F10 manual activation ✓ Last client disconnects	✓ F10 manual activation ✓ Last client disconnects ✓ *Autosave*

Autosave is a very very light process that can be done in any DCS condition up to date without performance impact, while doing all the necessary maths only when the simulation is closing. Autosave does not create the new saved file. To do that, a save action is needed.

If you use auto-save feature, DSMC will automatically desanitized the file for you (while keeping all the previous modification you maybe did on it). Else, you will have to do that yourself.

BEWARE: By default, DCS forbids the interaction with files and folders outside itself when run from within the mission. This process lowers the security level of DCS which, by default, prevents malicious code from being executed onto the Local File System.

The save action, performed by F10 menù or automatically at client disconnect will activate a process that will eventually produce one ".miz" file, named "(DSMC_yourmissionname)_001.miz".

If the mission already has the 3 digits sequence at the end of the file name, it will add 1 to it. For example, if you start a mission called *DSMC_test_024.miz*, the saved file will be *DSMC_test_025.miz.*

Remember: **each save process will overwrite the save file you just created in the previous save events**. So, if you save five times the scenery during the save mission, each time you will simply overwrite the previous save. DSMC doesn't care about previous save events: it will always look for the loaded file name and try to update or append the 3-digit code.

7 HOW DOES IT WORK?

DSMC is a mod, loaded in DCS World using the inbuilt modding API. It's designed to be as light as possible during simulations, and if everything is ok you won't even notice that it's doing his job. How? Because almost all the functions that run within the simulation are event-related: they don't work continuously; they get in, only if called. And they use the same event recorder that DCS gives us and use for trigger systems.

Also, every function that runs inside the simulation, has as little calculations as possible and the least table traversing as possible. What does that mean? You shouldn't really care. What you should care about is that DSMC is trying its best to do not overload your system while running... at least when it's possible.

DSMC collects data to tables, and when it's the right moment, do all the necessary calculations and pack up a new .miz file. This part is very heavy to your system, so <u>it's never done during the simulation unless you call it by choice using F10 menu options</u>. DSMC does that automatically when the last clients disconnect... that is a crafty way to say that if you are in single player mode, you necessarily have to save the mission using F10 menu because you won't have clients connected.

There is also an additional trick, the autosave function. That is an interesting way to collect the saved data continuously every "n" minutes, and when the simulation closes down. This means both; if you end the mission or if you close DCS to desktop, it will create the saved files. When this option is enabled (or when you're running it in dedicated server mode or in a standard host with no 3D graphic), DSMC will automatically try to desanitize your *missionscripting.lua*! **This process may fail if you have your main DCS directory protected by windows from writing**.

In the end, when autosave option is enabled, you will also have a DCS crash recover save functionality! As soon as you restart DCS, this mod will check for saved data availability and if it finds what is

needed it will create a new autosaved file to be immediately set as current mission for servers & multiplayer.

In the next pages I'll try to add much more details about how the mod behaves and "think", if you like that word. Reading them will let you understand better the design and the workflow, if you like.

7.1   Structure

To add details about how it's done, DSMC is organized in different folders and group of files, which is the reason I suggest installing using a mod manager. Files, all placed in Saved Games\DCS.whatever\ folder, are separated in subdirectories:

- "\Scripts\Hooks\" folder. This contains a single mod file: DSMC_hooks.lua
- "\DSMC\" folder, with "\Docs\" and "\Files\" subfolder. Here you will find all the core files of DSMC
- "\Mods\tech\DSMC\" folder: here there are those files necessary for the in-game options menu

7.2   Why multiple files?

As said, DCS has an inbuilt API mod system, which is utilised by placing a file in the "\Scripts\Hooks\" folder, that is read automatically when you start DCS World.

This file leads the loading of all the other core files, say lua "modules", which are inside the "\DSMC\" folder. Which files are loaded? Those that are related to the options you choose in the special options menù… we could say that each file other than "UTIL.lua" and "SAVE.lua" are related to a particular option. For example, "SPWN.lua" is the file related about tracking spawned unit, or "MOBJ.lua" is the file related the persistence of map object state.

This is done to reduce the impact of flaws in a single file, which will break that features while keeping functional everything else… and also, this way, any unnecessary code is not even loaded to the

simulation. Is this better than loading everything? No, but it's cleaner… what is not there cannot be an issue when bug fixing.

Regarding the files included in DSMC, only three are mandatory to let it run: SAVE.lua, EMDB_inj.lua and UTIL.lua. Obviously if you delete the others, bad things may happen.

7.3 Workflow: loading the mod & data transfer

DSMC files are loaded as soon as you enter DCS, but are used only when you load a mission. DCS lua geeks like to say that this mod is run into the "Server environment". I won't dig too much into this, but to let you understand better what is going on, you should at least know that DCS World runs two different coding "environments".

One, the *server env*, is almost unlocked and powerful except for compiled .dll files. You can read, write, modify data as you like and you can use all the functions ED put there to make it work. This env is used in main menu, mission editor, etc etc.

The other, called *mission environment* is where it runs all the lua code used to run the simulation. This environment is heavily protected, so much that is completely separated from the server env. You can't even read or write (unless you "desanitize" it, as you know).

The relationship between those env are difficult, most like those between you and your cat, if you have one. Server env can send code, functions, info inside the mission env. Like your cat can do with you. But mission env can't even pass information to the server env, except for few on/off parameters (flags) or text strings (chat, radio messages). Like you when you try to teach something to your cat: it works only to the extend he likes… other things simply don't work.

Back to us, DSMC runs in the server env to take advantage of its power but we put some code inside the mission env to build a data collection and exchange interface. I call this "injecting the code".

So, step by step, this happens:

Each time a mission is loaded, DSMC_hooks.lua will check if the .miz file name start with the magic four letter "DSMC": if so, it loads everything you asked in the options menu. If not, it stops immediately letting you run DCS as if DSMC never existed. Also, it creates a temporary folder in the default mission folder used to archive autosave data and information.

Once loaded, the only file that perform actions inside the mission environment is "EMDB_inj.lua": it is the data collector for saving activity.

Our hooks file also set a series of callbacks that will pull the trigger of the code when needed, using DCS modding API system. If you run a "desanitized" server, those triggers are almost completely performed by reading & writing files outside DCS mission environment to the DCS external environment: some call it plugin env, other server env, but it's the same: it's the environment used for main menu code with free access to the main lua functions.

If you run a standard server, the only way to get data from a running mission without desanitizing the environment is by using trigger messages. Collected data is stored into tables that instead of being written out to the mission environment, is "serialized" into text and sent out via trigger messages. Those messages activate a callback, that will check message content to find a table and if so, it will load the message as a string that magically is retransformed into a table.

With autosave options, the EMBD_inj.lua code will collect updated information every "n" minutes and saves them into tables. These are directly written to lua files in the temp directory. That is why when using autosave you need to run the "desanitized" version; else, you won't be able to write those tables out.

As described before, those things are designed to be "light", with as low impact as possible on CPU workload on the sim. Data is collected mostly using DCS event handlers (hoggit wiki is your friend if you need a refresh), while the only recurring checks are about position & life points, and they're called in at every autosave.

7.4 Workflow: running the save procedure

In the first pages of this chapter you already had a sneak peek about when DSMC will save your scenario and when not, but here we will dig a little bit more.

The mod implements a very precise save procedure, which can be done "partially" or "completely". The procedure is also smart: it will check if you run a desanitized environment and adapt itself.

The partial procedure simply collects data from the mission env and stores them into tables. If you run a desanitized environment, DSMC will also proceed by saving those tables into physical files. This is a fast & light process.

The complete procedure does the same things of the partial one but also performs all the operations in the server environment, including creating the new miz file or setting up the server config file. That is an intensive process, that **can literally halt your server for minutes** if run during a game session.

The complete save procedure can be described like this:

- Gets the "save now" order
- EMBD_inj took all the events populated tables (units, logistic, deaths, etc) and saves them
- Hooks file code load these tables into the server env, and triggers SAVE module to do its job
- SAVE module opens the original miz file you loaded, modifies mission, warehouse, dictionary and mapResource file, then repacks everything;
- Hooks then check if DCS is in server mode: if so, it will open server config files (serversettings.lua) and sets the path of this new copy as the first file to be loaded once DCS run

That's it. Simple, right? Well, **NO**, for goodness sake, it's not simple. The complicated part is about matching data from the mission environment, transforming them with consistency in a format expendable for the server environment, and put every piece in the

correct place of the new .miz file that is created. That part is not covered in detail in this manual: really... with my ability to synthesize and my language skills I'll probably end up in something like Lord of the Ring capable to kill your grammar teacher in no more than 50 words [sic]. [*EDITOR: I have no idea what the mad person was writing here, so I left it as is for the enjoyment of future generations of scripters*]

7.5   Workflow: where are my saved scenery files?

This time the answer is truly easy. All saved missions are stored in your personal "Saved Games\DCS*.whatever*\Missions" folder. Would you like to change this? Don't. This time it's better if you adapt your habit on this.

Basically, you will have one file saved if you run single player, while you're going to find **two** files if you run a server with autosave enabled.

The file you will always find is **DSMC_*yourmizfilename*_###.miz**, where "*###*" is a progressive number starting from 001. That will represent the last complete save procedure picture of your mission. It will be always there after a successful save.

So, for example, you want to host a mission with your friend using your dedicated server. You set up everything, ready for launch.

Your mission file is "ExampleMission.miz" what will happen? Nothing, cause you didn't add the DSMC tag: DSC will run like DSMC is not existing.

Ok then, you rename it to "DSMC_ExampleMission.miz". This time DSMC will run: if you quickly connect to the server you might see the "introduction" message of DSMC that confirm correct mod loading. Once all the clients are in, you fly your mission.

At the end of the mission, everyone disconnect from the server. In less than few seconds, a new mission ".miz" file is created, named "DSMC_ExampleMission_001.miz". You can edit it, or not, and fly again.

If you fly another sortie with "DSMC_ExampleMission_001.miz", the new saved file will be named "DSMC_ExampleMission_002.miz", and so on.

The second file is **DSMC_ServerReload_*nnn*.miz**: as said before "nnn" is a progressive number used to enable the possibility for the server to load a mission after the other, without having to touch the Mission Editor.

Why creating other files instead of updating the first one? For two valid reasons:

1. Windows won't let you do that. While DCS is running, the open .miz file is considered "in use", therefore a VRS error will be prompted in dcs.log and nothing will be saved (that's a very good reason, you know);
2. Cause if something goes wrong…I bet you would prefer to have the original file to work on.

8   DSMC CUSTOMIZATION

In this paragraph I'll try to explain how each option is going to change DSMC behaviour & alter the save .miz file result.

You know, in the testing phase of DSMC (yes, we did one) some of the additional options DSMC provided weren't understood. Also, some of those options will require to be used correctly, else they won't work!

As described in chapter 6 the mod will "read" your customization preference in different ways depending on how you run DCS (single player, multiplayer with graphics, server mode…. Etc).

Basically, if you run DCS with graphic interface, being able to access to the options menu, you will see some customizable options directly in the "special options" menu, where DSMC is like any other modules.

Else, you must customize the *DSMC_Dedicated_Server_options.lua* file available in the mod root folder before enabling the mod.

DSMC needs also some stability in the developing process, therefore some behaviours can't be customized and will be present always, or at least depending on the design choices of the mission designer.

Since there are many things to discuss, here's a small list of where you can find each elements:

- Chapter 8.1: Always on features: things that you won't be able to customize, part of DSMC core;
- Chapter 8.2: Customizable options: things you can customize;

Each option is explained separately starting with those customizable by the options menu. You will see some "variable names" in Chapter 8.2: don't worry these are there as reference values for server customization which need .lua edit, cause main options work the same way and therefore it seemed stupid to me to write them twice.

8.1 Always-on features

DSMC is not a "mod collections" or a "Sandbox mod", it's a specific mod with some specific goals: some things that are mandatory to reach these goals and therefore won't be removable. They may me obvious, but it's not bad to make a simple list here. Let's start.

8.1.1 Units and user object persistency

- Ground units' position update: DSMC will save the last position of any alive units when you save the scenery;
- Ground units alive/dead state: Dead units will be removed. You can choose about the "wreckage" creation, but not about the killing;
- Spawned units will be added to the saved files unless being removed (killed or by script) when the file is saved;
- Static object can change coalition! If any static object is surrounded by enemy coalition units within 1000 m when the mission end, in the next mission it will change coalition like happens for airbases & helipads;
- Ships position update: **Except the carrier group**, ships will do the very same of ground units;
- **Air units are not… units. They are slots. Therefore, no update is provided**. That will preserve all your AI planning… check Automatic Slot Creation if you need dynamic behaviour; Also, if you need attrition on aircraft in the scenery, you have the resource tracking as per chapter 8.1.4 if the aircraft take off from a valid airbase or FARP (Else for air-birth slots or "on ground" slots, they won't be tracked in any way, and you'll always find them in the saved mission)
- **Routes of both ground and naval groups will be removed**. That is necessary to avoid weird situation where a ground group moved for 30 miles will try to go back to the start before moving into new position. As described elsewhere, you should provide some dynamic routing (In DSMC 2.0 routes will be automatically

generated in the saved mission, as it will have a tactical AI that will try to conquer the most important cities);

- Start time and date of the saved mission will be updated, you can choose the way DSMC can interact with it (check paragraph 8.2.2).

8.1.2 Map object state persistence

This feature keeps track of destroyed scenery objects. Scenery objects are bridges, buildings, factories, airports structures, **anything that is by default already on the map**. For example, a destroyed bridge will stay destroyed in the saved mission.

How it works: This is done by keeping track of every "death" event related to a scenery object. In the saved mission there will be a single trigger created (that will be "updated" in subsequent missions) that contains a very small "zone" for each object, in which anything is set to "dead". These zones are by default created as "hidden".

If you need to "restore" a destroyed object, you can simply get to the zone list, make them show up and remove the zone related to the object you want to restore. You will recognize those zones easily cause they all start with "*DSMC_ScenDest*" and followed by the DCS object ID. Here's what you can see in the saved file after a big explosion in Zugdidi (zones have been unhidden):

These zones are very small, but sufficient to set that object as dead.

Also, that works for airbase warehouses... if you destroy the fuel tanks, that airbase won't be able to refuel aircraft, even in subsequent missions and even if the warehouse in the mission editor show available fuel (in the next picture: Batumi airport fuel & weapon storage building as visibile in the mission editor)

An important side note: as said, if you destroy all ammo or petrol warehouses of an airport, it won't be able to give the corresponding item to anything spawning on that... even in a subsequent saved mission. But, as for a DCS limitation, anything that is spawned in the ME and activated at second "0" of the mission, will be able to load what is needed anyway. That is because the persistency trigger is set at mission start, but mission start is not before the creation of mission object such as airplanes or units. What does this mean? Two things. Think about this scenario: in mission "003" all fuel tanks of Gudauta are destroyed. Therefore:

- If you want Gudauta to be unable to give fuel to airplane, you must set those airplanes as "late activation" (even 1 second);

- If you need something to start immediately from Gudauta, simply add the group and don't touch anything else;

- If you want something to start anyway but not at mission start, you need to set as "uncontrolled" and then push the task to start it accordingly to your needs.

Performance impact: none.

Issues: not exactly a DSMC issue, but… destroying bridges could lead to bad things when you try to route convoys... and that's the purpose to break a bridge, to be honest. Therefore, keep an eye on what you task your ground forces after a few missions... you don't really want a Tank to do 3000 kilometers via the Roki tunnel because you wanted to cross a 200 m bridge nearby Sochi.

8.1.3 *Randomized climate stats-driven weather update*

This feature automatically updates the weather in the saved mission. The weather will be randomized using real world statistic data and will work differently in every map.

Randomization will modify, using month/day/hour dataset:

- Temperature;
- Precipitation;
- Wind strength and directions;
- Fog and/or Sandstorm;
- QNH;
- Clouds coverage/preset, minimum altitude and thickness.

So yes, in January over the shore nearby Sochi it's likely to find a bit of fog in early morning.

How it works: Very boring bunch of code here. I'll save you this one.

Performance impact: none.

Issues: currently the cloud randomization is limited as "try to identify preset over the available that might be closer to the randomized climate outcome". Therefore, it's expected to improve over time.

### 8.1.4 Resource tracking & scenery attrition

This feature is basically always on **but it can be skipped away easily if you don't need it**. Resource tracking, as obvious, track all items as aircraft, fuel & weapons **when the "warehouse" is not unlimited**. So, if you won't to use this feature you only need to leave every warehouse unlimited as DCS does by default.

When used properly this is a very powerful feature: you will be able to set-up from the easiest to the most complex attrition scenario, from limiting only some resources like aircraft or fuel to creating an articulated logistic net that can be enhanced by rotary wing operation or targeted by your enemies. Check scenery design guidelines for more details about the potential of this feature.

To enable the resource tracking you need to make the warehouse "limited" where you need. There are three type of resources in DCS:

- Aircrafts;
- Liquids (fuel);
- Munitions (& pods);

To access those, you'll need to enter the warehouse of the airbase, heliport, ship, oilrig, warehouse objects.
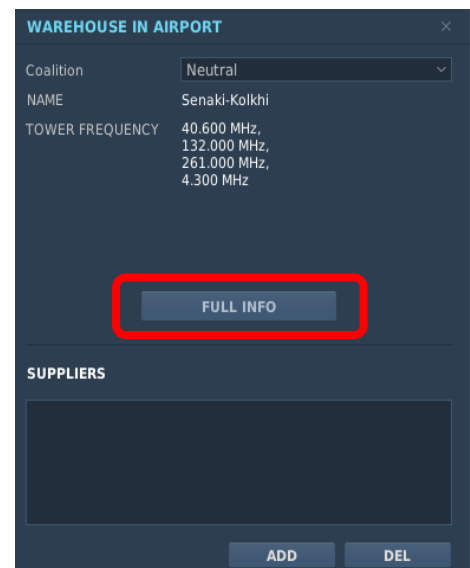
This is done by clicking the "full info" button while being in the airbase menù, that you get by clicking on the object in the mission editor map. Once clicked the resource manager menù will open on the left. There's a tab for each category, each one with a "Unlimited *(category)*" menù:
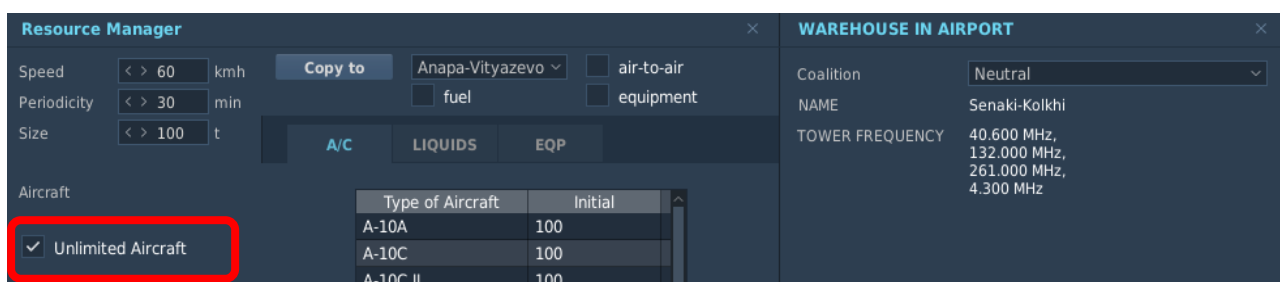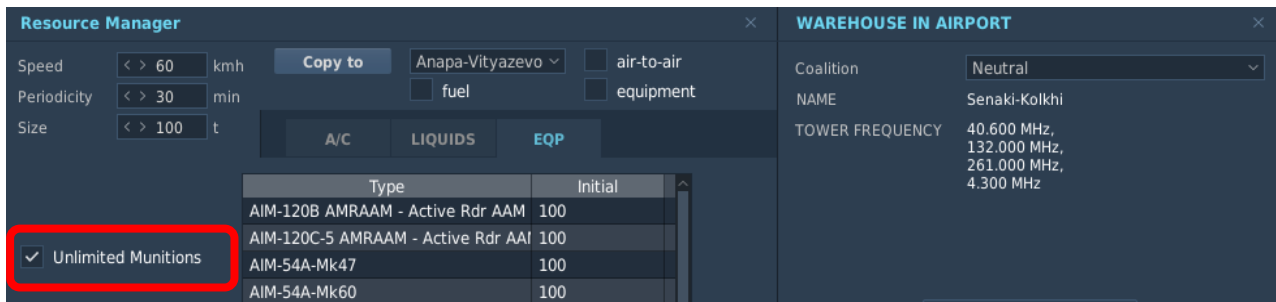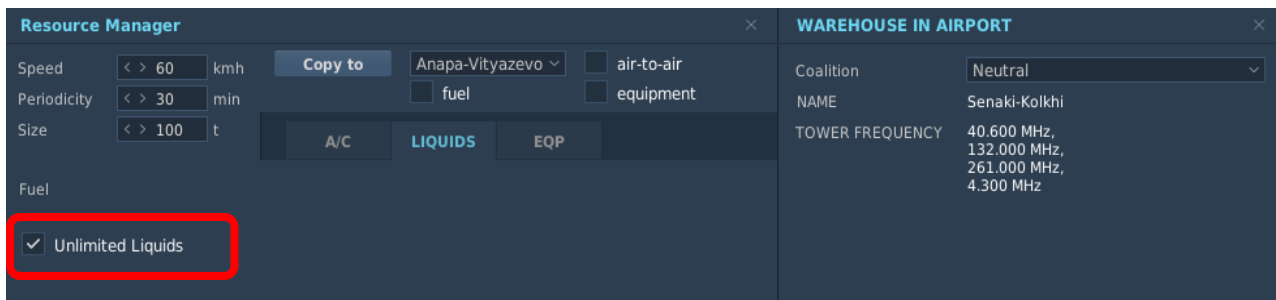
Resource will be tracked for every **unchecked** "unlimited" category, while will be ignored if the category is unlimited. What does this mean?

If you take-off from an airbase with a couple of mirage armed with matra magic, then you land in another airbase, all those things (including fuel aboard the airplanes) will be removed from the departure airbase and will be added in the arrival airbase. But those operations will be done only if the airbase is "limited": if the departure airbase is unlimited, you will have items added in the arrival base… but nothings happen in the departure. And vice-versa.

BEWARE: fuel tanks, pods & onboard guns won't be tracked. DSMC can't track "fuel tank" & "misc" items such pods, side guns. The POL stored inside the fuel tank is tracked, the object "fuel tank" is not. This is a limitation DSMC can't overcome due to DCS getAmmo() function limits.

How it works: That was a very complex task to achieve, and it sticks to a couple of rules you must adhere. Resource management in DCS (RMS) is not accessible by scripting, so **I was not allowed to track changes in real-time during the mission**. The only way to track that data is "running" is via parallel code outside. The main

differences you must consider are both about how items movement are tracked and how the resupply system is employed.

About items tracking, this is the most important things to know:

- The mechanism used by DCS to add and remove items from warehouse is by spawn & de-spawn. Instead, DSMC will use takeoff & landing events;
- The RMS works in real time during the mission… and it will still do its job while DSMC is running in the mission. And you will actually see the RMS adding & removing items in the warehouses. But those movements are not related to DSMC tracking: once the mission end, everything will be erased… and only movements tracked with DSMC will be used to update the warehouses.

About resupply net: While the real time RMS works will be ignored, DSMC provide a valuable supply system alternative. So:

- Real time supplies will not be tracked, so nothing that you could set using the RMS variables "speed", "periodicity", "size" and "operating level" will apply;
- But, still, DSMC will actually check the exact supply list that you can set in the mission editor. The suppliers can be used to built a full production-to-end of pipeline… but it's a complex task, therefore I added a more detailed explanation in the *mission design guidelines* chapter.
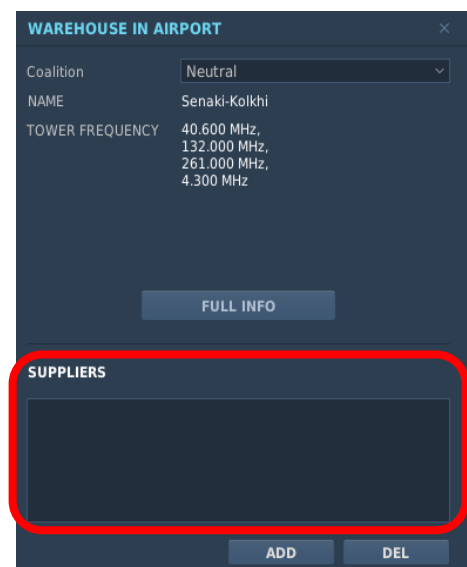
Again, **DCS maths is performed real time**, during the simulation, therefore it's all about available items & petrol in the running mission. Else, **DSMC maths is done after the mission**, so it's about the starting items & petrol available for the next mission.

Check *mission designer guidelines* chapter! it will help you to get the best this feature can give you.

8.1.5 Add spawned objects to the saved mission

DSMC will track spawned objects and add them in the saved mission file. Both units & statics.

BEWARE: as better explained in the mission design guidelines section, **you should be very aware that spawning should be controlled with care**: if you have a mission that spawns some groups of tanks each launch, you will have the scenario clogged with hundreds of tanks in very few save iteration… cause each time DSMC will save those tanks, but you keep spawning other at any saved mission start! DSMC can't know that those "spawned tanks" were the same of the previous mission.

The most common issue with this is about mission randomization, cause **if you use a defined set of assets that are randomized at each mission start, those assets will be duplicated each mission start**. First time you spawn 15 groups; then next time you spawn 15 groups again… but the first 15 are already there cause DSMC tracked them! Functions like "teleport" in mist or similar are nothing more than a refined "destroy & rebuilt", that use the spawning functions of DCS, and will have the same effect.

There's a solution: don't spawn, or do that only once in the very first mission: if you use AI tasking with mist, or MOOSE, or any other scripts to handle your enemy behaviour this will be a better solution over any randomization. It can be difficult in the very first approach, cause many mission designers (including myself) have been trained over time to "randomization is the way to ensure re-playability of the mission".

But that is the wrong approach here: re-playability is a factor when you can't alter the scenario over time, which is exactly what DSMC provide: here you don't need to re-play the mission! Here you need to set different objectives over time cause the previous ones might be gone very fast. Obviously coupling this with an AI enhanced behaviour to react to your actions, like a small repositioning script, is

much better but currently DSMC does not provide that and it's up to you as a mission designer to provide this at its best.

BEWARE: you can't spawn warehouse object category: that would break DCS mission structure while saving.

8.2 Customizable DSMC options that alter mod behavior and saved files content

DSMC SETUP & CUSTOMIZATION FOR SINGLE PLAYER & SERVER WITH GRAPHICS

8.2.1

Saved scenery file (.miz) preferences

☑ Keeps destroyed vehicles & aircraft wreckage
○ Saved scenery start at end of current mission (continous time)

8.2.2
◉ Saved scenery start next day, random hour from 04:00 ▾ to 23:00 ▾
○ Saved scenery start on real world date, same time as original mission

8.2.3
☐ Automatically create clients slot on heliports
☐ Automatically create clients slot on airbases

8.2.4
Airbase slot creation require mission designer to follow specific rules, check manual!
Create slot for coalition: All ▾

8.2.5
☑ Automatic rebuilt of the supply net (see manual for details)

8.2.6
☐ Prevent atmosferic fog creation in saved files

CTLD scripts enhanced automation features

☑ Helicopter client pilot recognition on spawn
☑ Trucks (troops, crates), IFV, APC (troops) vehicles recognition   **8.2.11**

Beware: if you use DSMC's CTLD version, these option are forced on

Server with graphics options
These options takes effect only if you run a local multiplayer server from the main menù
If you run a dedicated server o 'norender' server, use 'DSMC_Dedicated_Server_options.lua'
Following option require, and automatically perform, desanitization of 'MissionScripting.lua'

8.2.7
☑ Autosave every (minutes): ⬤———————— 1

8.2.8
☐ Restart saved mission every (hours): ———⬤———— 6

8.2.9
☐ Remove F10 menù option to save the mission

8.2.10
☐ Enable detail debug mode. WARNING: activate only for reproduce a bug

Single player and host with graphic use is user friendly: you won't need to edit any lua file! There is a special options menu in DCS, like any other modules. The special menu page has the layout you can see in the figure above: there is only one very important thing you need to remember: **each time you change an option, you will need to close & restart DCS**. Let's see all the possible customization!

8.2.1 Create scenery wreckage

Variable name: *DSMC_StaticDeadUnits* – true/false

Enable/Disable the feature that creates a "dead" static object of the same shape that was the real unit where this unit died. Not applicable to map objects (that is another server-customization option).

Purpose? almost nothing, but it's useful for scenery immersion.

BEWARE: to avoid issues with DCS ground units pathfinding, if the deaths happens over a taxiway, a runway or a road, it won't create wreckage!

How it works: This is done by keeping track of every "death" event related to a unit or static object. DSMC will natively remove the object from the map. With this option enabled, it will also create a static object unit of the same type of the killed one (if available), set as "dead" and "hidden" in the mission editor to avoid confusion with other units.

This is pretty straightforward for ground units: where they die, the dead objects are created. It's less obvious for other categories:

- Infantry won't create dead objects. Cause this is not carmageddon… or if you're crueler, everyone has been eaten overnight by animals;
- Ships… well, they sunk. No static there;
- Planes & helos: they actually produce a static object where DCS decides that the unit is killed. Which might not be exactly where it crashes, but maybe near that point.

Here's a picture of what you should expect in the mission editor (note that I already checked the "show hidden units" box!):
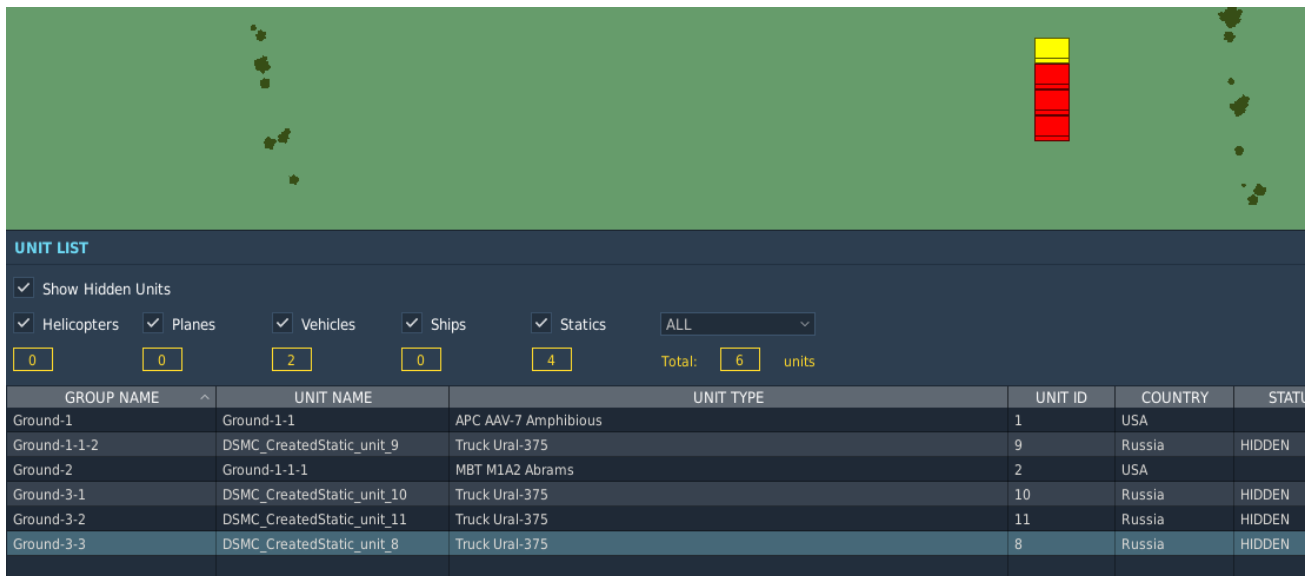
As you can notice if you can magnify the picture, you can see that:

- Dead units are hidden. I know, I wrote that before…
- Units name have been changed! They always start with "DSCM_CreateStatic_";
- Group name is not changed.

Looking in the objects tab there are the dead & hidden box checked:
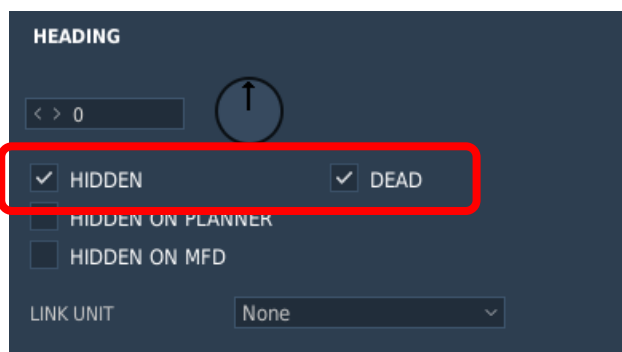
Performance impact: almost none, but with intense scenario might lead fast to hundreds of static dead objects.

Issues: none reported. Static creation might not happen if DCS does not have a proper "dead object" 3D model.

8.2.2 Mission date & start time update method

Variable name: *DSMC_UpdateStartTime_mode* – 1, 2 or 3

You can choose the way DSMC will modify the starting hour and date of the next mission. Here you have 4 choices:

1. Change it, using the start time as the exact simulation date & time of the save command *(option variable* to 1);
2. Change it, using a randomized hour in the next day relative to the mission flown and saved (*option variable* to 2);
3. Change it, using the very same hour of the original mission but the current real-world date (*option variable* to 3).

So, here you have to decide how this feature would change your saved file.

With **Option 1 "Saved scenery start at end of current mission (continuous time)"**, you should find the exact time (& date) of when the mission has been saved. For example, if you start a mission at 14:00 and the save happens at 16:50, then you will find 16:50 as start time in the saved mission file.

Option 2 "Saved scenery start next day, random hour from XX:XX to YY:YY" is different. First, it will change the mission date, adding 1 day: If your mission was set on 14th February 2018, the saved one will be on 15th February 2018. Then, it will change the mission start time with a random value starting from XX:XX and YY:YY (those values can be modified choosing the values, default 04:00 to 23:00). These two variables are defined in the lua config file for dedicated server as:

- Variable name: *DSMC_StarTimeHourMin –* 1 ÷ 14
- Variable name: *DSMC_StarTimeHourMax –* 15 ÷ 23

These options are used only if *DSMC_UpdateStartTime* is set to true and the additional *DSMC_UpdateStartTime_mode* option variable is set to "2". Basically when you use that start time update mode, you ask DSMC to choose a random "start hour" value that will be applied to your saved miz file. These option will limit the randomization span between a "minimum" and a "maximum" values.

For example, if you want mission starting only during daytime, you can set the min value to "9" and the max value to "15". As you can see, you can't have the opposite like a "only night" randomization…

but you can always set it with "1" and "23", basically telling DSCM to randomize the starting value all over the 24 hour of a day.

If you set a value outside the defined range, then your PC might explode as a nuclear bomb of about 3 megatons… or more likely it will automatically revert to a default value, that is "4" for minimums and "16" for maximum.

This enable a much more variable scenario: for example, if you run a 24/7 server that is restarted every 8 hours, you will have the scenario date moving forward 3 times a day that will eventually lead to almost 1 months in less than 10 days. This thing is quite useful coupled with weather update, because you can see your scenery changing a lot over time.

Option 3 "Saved scenery start on real world date, same time as original mission" is a very specific feature asked by one of my testers that I decided to integrate in the code: It will change the mission date to the real-world date, while keeping the mission start time always the same.

If the mission is saved on 24th May, real world, the saved mission will be set in 24th May. But the hour will be ignored, so if the original mission has been edited to start at 07:00, all the saved mission will start at 07:00. That can be an ideal solution for a 24/7 server that restart once a day at a specific time.

How it works: It takes the value from the "mission" file inside the ".miz" archive and then update accordingly to the option variable. Those values are the same you set in the mission editor, time & weather menu. No specific additional action required.
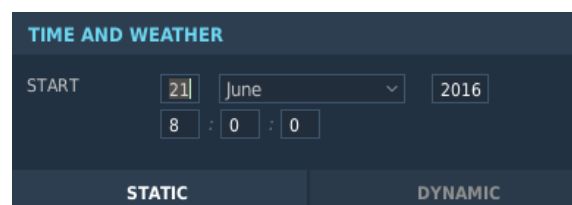
These data are stored in the ".miz" file in 4 variables, that refers to the year, month, day and hour. The hour value is in seconds, while the others are integer numbers. For example, 14th May 2019 starting at 06:00 would be:

- Year: 2019

- Month: 5
- Day: 14
- Start time: 21.000 (seconds)

DSMC will gather those values directly from the mission file and then alter them accordingly to the option you choose.

Performance impact: none.

Issues: when thinking about sceneries that you want to be mostly in day-light, pay attention at option 2 during winter. To force day-light condition, randomization boundaries can be altered accordingly with advanced server options. Pay attention with mission dated 31st December 1969… please avoid that specific date: windows won't like it sometimes (trust me).

8.2.3 Automatic client's flights (slots) creation

Variable name: *DSMC_CreateSlotHeliports* – true/false

Variable name: *DSMC_CreateSlotAirbases* – true/false

These options allow DCS to clear the all client's flights in an airbase or heliport and add new flights according to the DCS warehouse content, **up to 8 single-ship flight for flyable aircraft**, if the warehouse is not set as "Unlimited Aircraft" and if it's belonging to any coalition but "Neutral", depending on parkings availability.

This feature is designed with 24/7 hours' server in mind, to allow a plug & play updated client's flights layout each time the mission restarts. That said, these features can also be useful for squadrons or mission designer that do mission editor update of each saved files.

When employed correctly, this option will enable a very powerful gameplay feature, with a couple of drawbacks to be considered. The gain is easy to understand:

- No more slot block needed: The system will dynamically remove & add slots each time;
- No action required by the mission designer to add or remove slots from the mission file in case of coalition change of airbase

or FARP: The system add slots of the coalition which own the airbase (the warehouse, to be precise): you will actually be able to "stole" items & therefore slots when you get the airbase/FARP ownership;

- AI flights won't be affected, it's only a client flight thing.

The drawbacks:

- Read carefully "how it works" and chapter 11.8. Can be difficult to avoid errors with airbases, and you will need to use the warehouse system, or it won't work as expected;
- Every created flight won't have waypoints: the pilots will need to add them (or the mission designer starting from the saved mission;
- All the existing client flights (in the *limited* airbase/FARP) will be removed in these airbases/FARP, so pay attention to any script that use naming convention. *This is anyway one of the key mission design guidelines of DSMC…*

How it works: from now all the client's group added in the mission editor won't be aircrafts or helicopters groups anymore. For us, those are **slots**. Why? There's a couple of differences between all the other unit's types and aircrafts:

- the firsts do not have store in the DCS warehouses (RMS – Resource Management System), while the latter does.
- The firsts can be placed on map w/o limits, while the latter can be added (on airbases) as much as available parkings… and no more.

If you want to use this feature, the only two important rules to make this happen, are:

- the airbase/FARP must be not neutral;

- the warehouse must not have the option "unlimited aircraft" checked;
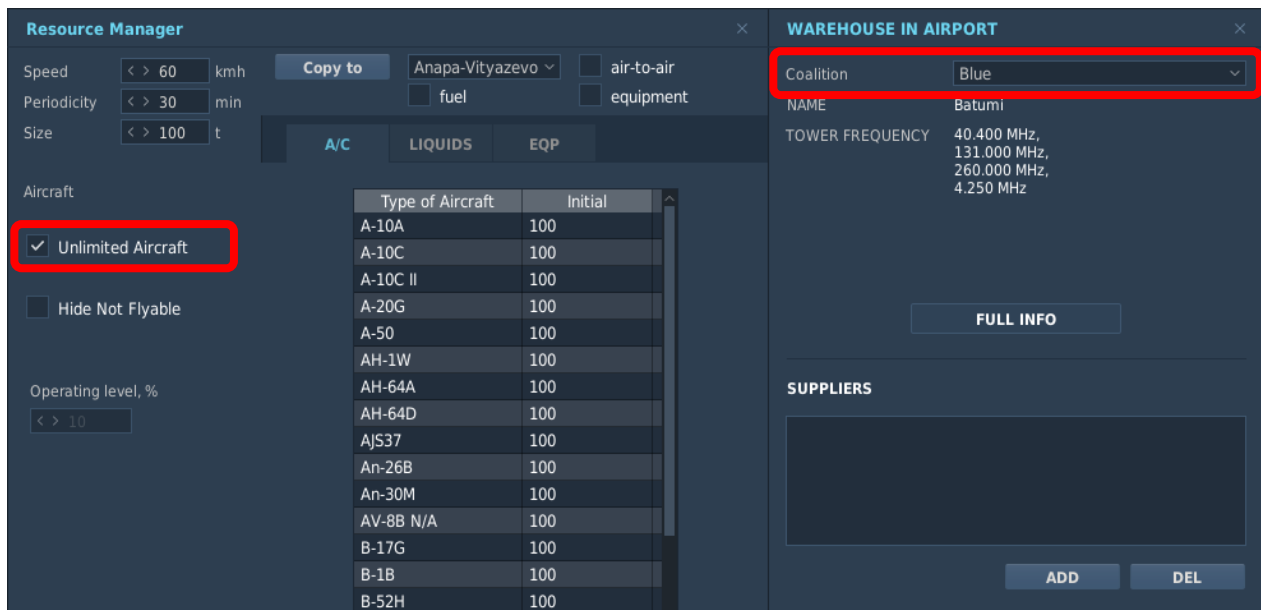
If these rules are not met, you PC will melt dow… ok, no, it simply won't create slots **in that specific airbase/FARP**. Since usually no-one touches neutral airbase warehouse and that DCS sets them by default unlimited, if you don't deliberately change it, it won't generate slots. There are also a couple of fact that as a mission designer you might want to consider:

- Airbase that does not meet the slot creation requirement also won't remove existing client's slots! So, if you won't to change clients group in a specific airbase… leave it "unlimited aircraft";
- The airbase could still be limited in liquids and munitions! Only the aircraft tab will define slot creation;
- Ships won't create slot in any conditions, even if set properly: so they will preserve their original client's slots.

If you want to employ this powerful feature, it's highly recommended that you read chapter 11.8 to fully understand how it works and what you can and can't do with it.

8.2.4 Create slot for coalition

Variable name: *DSMC_CreateSlotCoalition* – "all" / "blue" / "red"

It's set "all" by default, so that helicopter' slots are created for both coalitions. You can either set this "blue" or "red" to limit the slots creation to only one coalition.

If slot creation is disabled for one coalition, it will also mean that clients slots won't be deleted even in heliports/airbases that are set to "limited".

8.2.5 Automatic rebuilt of warehouse supply net

Variable name: *DSMC_WarehouseAutoSetup* – true/false

To better understand why this option can be relevant in your scenery, you might want to check *mission design guidelines* chapter first. But, if you won't, you might simply ignore and leave it true: if you don't set a supply net, or the system does not find the necessary objects in the simulation, it won't have effect.

This option will make the whole supply net to be rebuilt each time you save a mission, before logistic and resupply calculation takes effect.

That is a VERY useful tool if you as mission designer decide to agree to the standard supply net, explained in the *mission design guidelines*, paragraph "Standard supply net" n° X.X. Why? Easy explained: DSMC provides the possibility to "conquer" static objects, like warehouse and FOB. If your coalition conquer a fuel depot, or a warehouse, this warehouse should be linked to your resource supply net.. but it won't by default unless the mission designer change it in the mission file. With this option active, every mission the supply net will be rebuilt from scratch and therefore any newly acquired warehouse/FOB/etc will be included in the net and will be used immediately while calculating the resupplies.

8.2.6 Prevent atmospheric fog creation

Variable name: DSMC_DisableFog – true/false

This option, if true, will disable fog creation even if weather condition allows it.

This might help with specific scenarios for clients or players less experienced with fog conditions.

8.2.7 Autosave mission to prevent data loss in case of crash

Main variable name: *DSMC_AutosaveProcess* – true/false

Option variable name: *DSMC_AutosaveProcess_min* – 1 ÷ 480

Enable/Disable the autosave feature by the main variable. This feature is designed to prevent data loss in case of DCS crash or issues during very long & complex saves. This will also enable a recover feature that will try to create an updated .miz file at the next DCS start.

The option variable will define which frequency DSMC will perform a data recording used then for the save procedure. This data gathering procedure is very very light and it's unlikely to have performance impact.

How it works: explained in "How does it work" chapter.

Performance impact: almost none but not totally zero, still I wasn't able to measure it in any tested scenario.

Issues: none directly related, but if you use mods or a script that overwrite *missionscripting.lua* at launch for any reason (i.e. SLmod) this feature might not work and can halt/prevent DSMC to work properly. If you don't know what I'm talking about, you should be safe.

8.2.8 Automatic mission restart after "n" hours

This option explain itself: you can set a value in hours from 1 to 24. After that numbers of hours, DCS will save the mission and load the saved file immediately (safe mode enabled, check below).

Since some server admin prefer to close completely DCS each time a new mission has to be started, DSMC will retain the options layout from 1.2.6 and previous versions.

24/7 Server setup mode

Variable name: *DSMC_24_7_serverStandardSetup* – multiple values

This option is no more than an "options setup pack" to simplify or standardize the setup for a 24/7 server, and resembles fully the automatic restart option described for the single player/host with graphic mode.

It can be set with:

- false; if you set it as boolean "false" it will ignore the variable, and all the subsequent variable set will work as described below;
- 0 ÷ 24; numeric values from 0 to 24, and they are meant to be "hours". If set to 0, it will ignore the standard setup. If you set it like that, DSMC will save the scenery & restart the newly saved mission every "n" hours if no clients are connected (without closing the server). If clients are connected, it will retry every 10 min until all the clients are gone (a message will be prompted to anyone online every 10 mins calling for RTB and disconnect);
- "04:30"; text values in the "hh:mm" format. It's intended like a specific hour&minutes in the day where DCS will restart with the newly saved mission. With this set, DSMC will save and close & restart the newly saved mission once a day, right at the specified hour, and will kick out any client online (*and resources of non-landed aircraft will be lost!).* No warning is provided.

If you want to set up an autorestarting server, 24/7 online, and if you don't exactly have any specific reasons to change one of the behaviour described with the numeric or hour values, I strongly suggest to use this variable in one of the provided settings. By default in the DSMC newly downloaded package you will find it to "6", that means a restart every 6 hours.

Single variable options

If you need a more specific setup for your server, you can set *DSMC_24_7_serverStandardSetup* to "0" and then configure the following options one by one:

Update server mission list

Variable name: *DSMC_updateMissionList* – true/false

One of the useful features for 24/7 servers provided with DSMC is the update of the mission list when it closes: this allow a server to restart having the saved mission as the one that will be loaded.

But this could compromise any other situation, therefore if you set this option as false (default) the mission list won't be updated anymore.

Autosave additional features & setup

Variable name: *DSMC_AutosaveExit_hours* - 1 ÷ 24

I added this for servers that works 24/7 to automatically close DCS after a defined amount of time, in hours.

It can be set with numeric values from 1 to 25, and they are meant to be "hours". Any number above 24 (but I suggest to keep 25) means that this option won't have any effects and virtually the mission won't stop. Instead, any values between 1 and 24 will command a DCS shutdown after "n" hours.

Variable name: *DSMC_AutosaveExit_time* - 1 ÷ 23

This option works only if *DSMC_AutosaveExit_hours* is set to "false". It accept value from 1 to 23, and it will cause the server to save and close right at that time in 00:00 – 23:59 time span, real world. If you want your server to reset every morning at 5:00 am, you set this "5".

Variable name: *DSMC_AutosaveExit_safe* – true/false

This option is set true by default: it will make DSMC to check if any client is connected to the server. If so, it will print an advice and then

wait 10' to retry. Else, if false, it will save the server and close regardless of any connected clients.

So… why use this feature instead of closing the simulation with the same task manager script then?

Because this option does a very useful thing: DCS won't go down exactly when the timer goes to zero… *it will try* to go down. As said, if any client is still connected, then it will print a message to everyone and then it will retry in 10 minutes.

So, if one of your clients is going to be back late from a mission, the server will not close "killing" the pilot and the related resources. **It will wait until the player disconnect and then, when the timer arrives to the next 10 minutes' step, it will shut down.**

The main positive things about this is to have a controlled and gracefully shutdown: with very complex mission (>2000 units) the save time might be so long (>2 mins!) that windows might recognize DCS as "halted" and kill the process itself. Doing this, there's a probability that the save won't work as expected.

Restart option

Variable name: *DSMC_AutoRestart_active* – true/false

This is the key options you need if your intent is take advantage of the "graceful" and "guaranteed" mission end provided by DSMC, while still needing you DCS to fully close to desktop.

If you set this true, DSMC will load the new saved mission directly. If you set this false, DSMC will close DCS to desktop.

BEWARE: DSMC does not provide the restart function if you close the DCS to desktop

8.2.9 *Remove F10 menù option to save the mission*

Variable name: *DSMC_DisableF10save* – true/false

This option allows you to remove the F10 menu "save scenery" action. This cause in MP DSMC automatically saves in multiple ways, so it might be safer to prevent a player doing a save action while playing that might result in a violent lag or worse, server crash.

8.2.10   Debug mode

Variable name: *DSMC_DebugMode* – true/false

Enable/Disable the debug verbose log calls. That should be always left off unless you're willing to provide me as much information as possible for debug. Leaving this option as checked not only will produce and enormous amount of data into the dcs.log and DSMC.log files (Saved Games\DCS.*whatever*\Logs folder), but will print a lot of annoying triggered text message during the simulation. Really, leave it off unless needed to reproduce and report a bug.

Performance impact: depends on scenario complexity.

Issues: none, but if you're activating this…

8.2.11   CTLD support script: automatic helicopters recognition

First variable name: *DSMC_ctld_recognizeHelos* – true/false

Second variable name: DSMC_ctld_recognizeVehicles – true/false

DSMC automatically detect if CTLD is working inside the mission. The main requirements to make the DSMC's support to work properly are:

- CTLD must be loaded in the first 4 seconds of the mission;
- CTLD tables' names & structure must be the same than the vanilla code that you can find here: CTLD by Ciribob.

DSMC will provide these additional features:

- Automatic recognition of previously builded FOBs or FOBs added using mission editor: they will work as if you actually spawned them;
- Any group that is infantry-only will be recognized as an extractable group;

- If the first variable is true, you won't need naming conventions for any helicopter: they will be recognized as soon as they spawn in;
- If the second variable is true, every Truck, APC and IFV vehicles will also added as pilot, to let CA users to act as a transport.

9 DSMC WITH MULTIPLE INSTANCES OF DEDICATED SERVERS

This chapter has been prepared by Gh0st to let anyone that needs to launch multiple DCS servers to use DSMC in one or more instances at the same time.

9.1 Terms and definitions

In the following section, several terms will be used repeatedly. These terms will be underlined in the document.

There will also be code blocks. These are not pictures. You CAN copy/paste them.

The underlined terms will be defined as follows:

Installation Directory

The directory where DCS itself is installed. The location where the DCS game install files and executable are located.

Saved Games

The "Saved Games" folder in Windows, classically located at:

e.g. "C:\Users*YOUR USER HERE*\Saved Games"

It is **REQUIRED** to use the default C: drive saved games folder location.

Server

A running DCS executable that hosts a dedicated multiplayer mission

(Server) Instance

A reference to a *specific* multiplayer mission

(Server) Process

A reference to a *specific* DCS <u>server</u> hosting a *specific* <u>instance</u>

Install PATH

The Windows file path where the Installation Directory is located:

e.g. "C:\Program Files\Eagle Dynamics\DCS World OpenBeta"

Instance PATH

The Windows file path where the <u>Instance</u> data is located in the <u>Saved Games</u> folder:

e.g. For instance name "ServerTemplate"

"C:\Users*YOUR USER HERE*\Saved Games\ServerTemplate"

StartBat

The batch file (.bat) used to start your <u>Process</u>

Argument(s)

Parameters passed to a command line function

Variable

A word used to store a value

Port

The port used for players connecting to the Server

WebGUI_Port

The port used for the hoster to connect to the WebGUI management console for a Server

autoexec.cfg

A config file located in the Instance PATH "Config" folder, used for configuring various Server parameters. e.g.

"C:\Users*YOUR USER HERE*\Saved Games\Server1\Config\autoexec.cfg"

serverSettings.lua

A config file located in the Instance PATH "Config" folder, used for configuring various Server parameters. e.g.

"C:\Users*YOUR USER HERE*\Saved Games\Server1\Config\serverSettings.lua"

9.2 Setting up Multiple DCS instances

This section does not pertain directly to DSMC, but it is required to be followed in order to use DCS for multiple instances effectively. **If your server instances are not configured correctly, you will not be able to use DSMC!**

These methods have been tested with many 24/7 servers over long periods of automatic restarts with DSMC. (6+ Months server run time, 300+ automatic restart cycles per server)

Windows Server 2019 was used as the OS, though any Windows OS *should* work

1. **Install DCS to a location of your choosing**
   My preference is to *use a location other than the C: drive*.
   For this tutorial, I will install DCS Open Beta Dedicated Server to my "H:" drive in a folder named "DCSBeta". The installation directory is named "DCS World OpenBeta Server"

   1.1.   Install PATH: "H:\DCSBeta\DCS World OpenBeta Server"
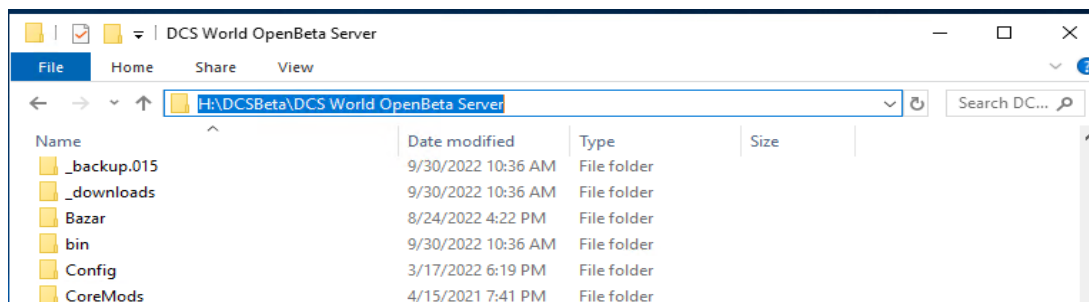
   1.2.   As always, change the Installation Directory so it is no longer "Read Only"

   This will reset and need to be changed after every game update. It is not strictly required, but better safe than sorry.
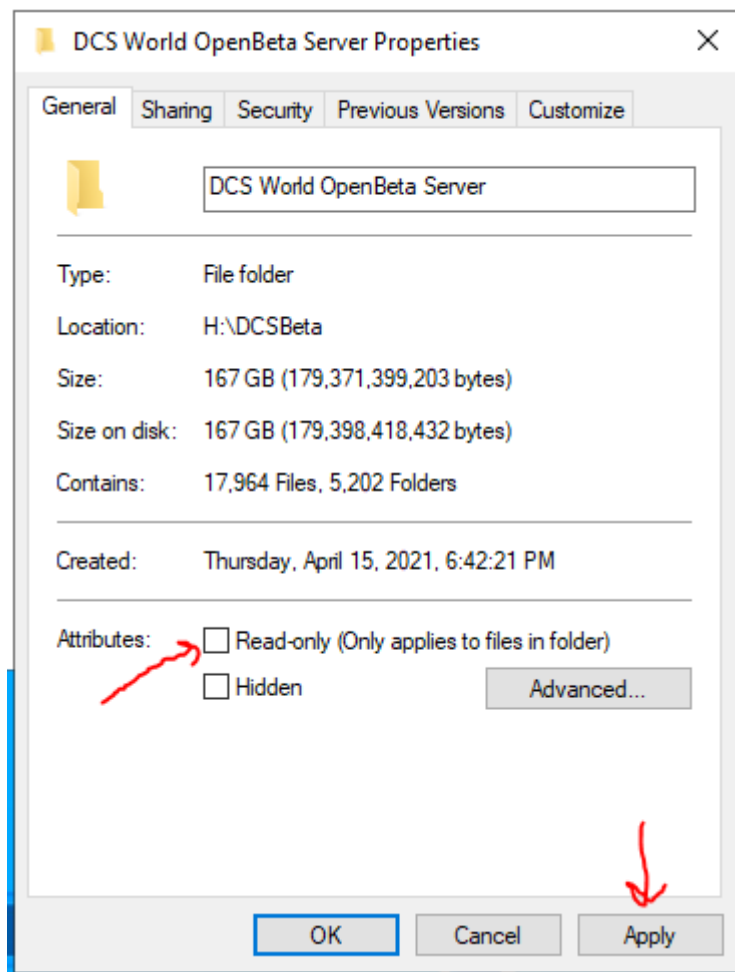
1.3. Updating DCS

This should only be done when all <u>server processes</u> are shut down and not running.

To update your DCS Installation, run "DCS_updater.exe" in the 'bin' folder of your <u>Installation Directory</u>.

Simply close the window when the update is complete, then you may start your <u>servers</u>.

2. **Setup your Saved Games Instance PATH(s)**

We will now set up our <u>Saved Games</u> folder for all of our <u>Server Instances</u>.

Navigate to your <u>Saved Games</u> folder and create a folder named "ServerTemplate" along with any folders for your <u>Server Instances</u>.

For this tutorial, I will also make folders for 2 <u>instances</u>, "PersianGulf" and "Syria"
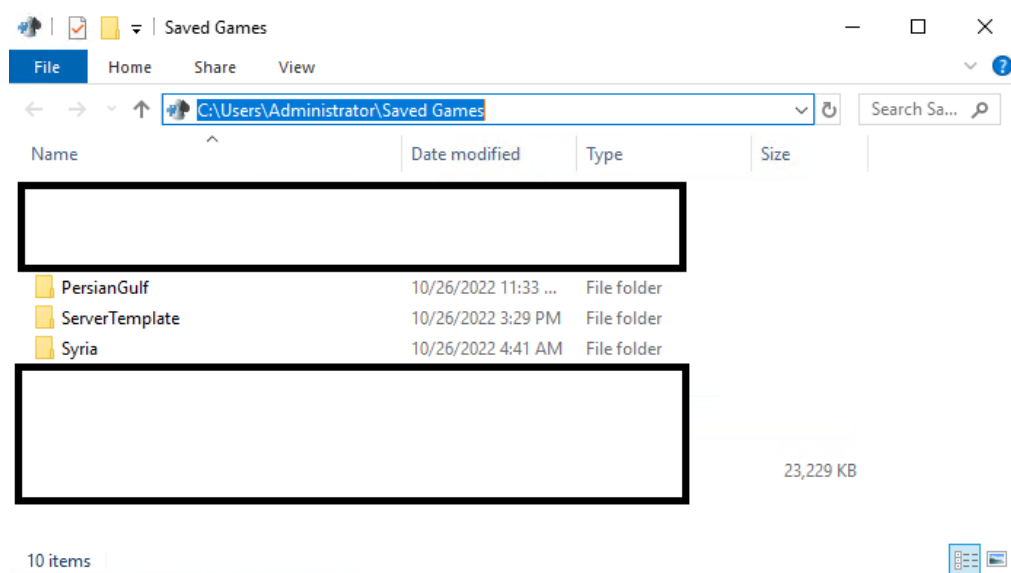
Change each folder so it is no longer "Read Only". This will need to be done after every game update. It may work without this step, but better safe than sorry.

3. **Setup a template <u>Server Instance</u>**

We will now set up a template <u>server instance</u> that will be used to make our life easier when creating new <u>instances</u>.
Go into the "bin" folder of your <u>Installation Directory</u>, in my case: "H:\DCSBeta\DCS World OpenBeta Server\bin"

We will create a StartBat batch file in the 'bin' folder, named: "StartTemplateServer.bat"

The content of the batch file should be as follows, all on a single line:

```
Start "" /high "H:\DCSBeta\DCS World OpenBeta Server\bin\DCS.exe" --server --norender --webgui -w ServerTemplate
```

Where "H:\DCSBeta\DCS World OpenBeta Server\" should reflect **your** <u>Install PATH.</u>

Explanation:

`Start "" /high`
Sets Windows to launch the application in high priority.

`"H:\DCSBeta\DCS World OpenBeta Server\bin\DCS.exe"`
<u>Install PATH</u> to the DCS executable.

`--server`
Tells DCS that we are running a server

`--norender`
Tells DCS that we are do not want to render the actual game on the server

`--webgui`
Tells DCS to enable the WebGUI functions

`-w ServerTemplate`
Sets the Instance PATH for the server process.
"ServerTemplate" is the name of the folder in saved games for this instance.

The following only needs to be done once to generate some template files:

- Run the batch file "StartTemplateServer.bat"
- Login to DCS when prompted in the server process window that opens.
- Wait for the server process to load completely, then close it.

4. **Setup StartBat for each server instance**

   Repeat the steps above in section "3. Setup a template Server Instance" for each instance.

   For simplicity:

   - Duplicate the "StartTemplateServer.bat" and rename it as you see fit for each instance e.g. "StartSyria.bat" & "StartPerianGulf.bat"

   -In each new batch file, change `-w ServerTemplate` to match your instance PATH name e.g. `-w Syria` & `-w PersianGulf`

5. **Setup *ServerTemplate* <u>Instance PATH</u> folder**

Navigate to the folder for the ServerTemplate instance in your Saved Games
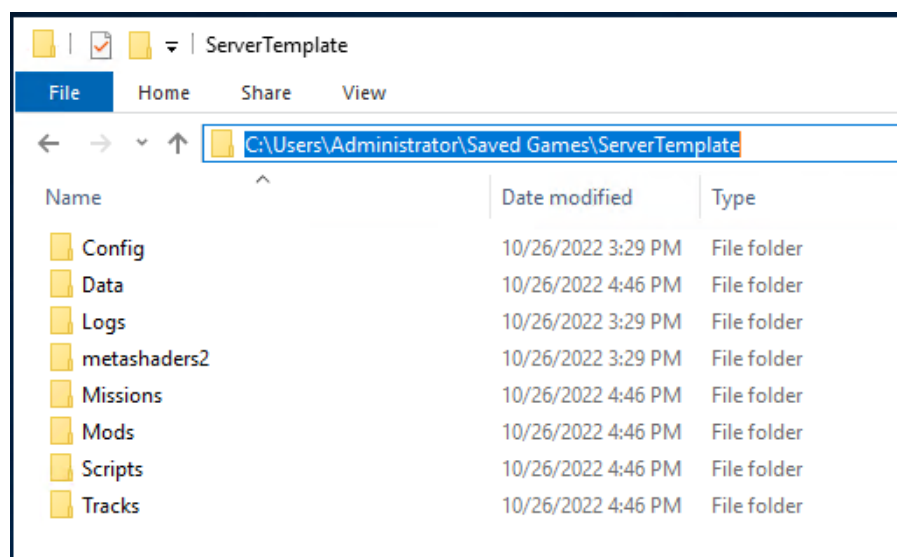
It should look as follows if you followed the previous steps:
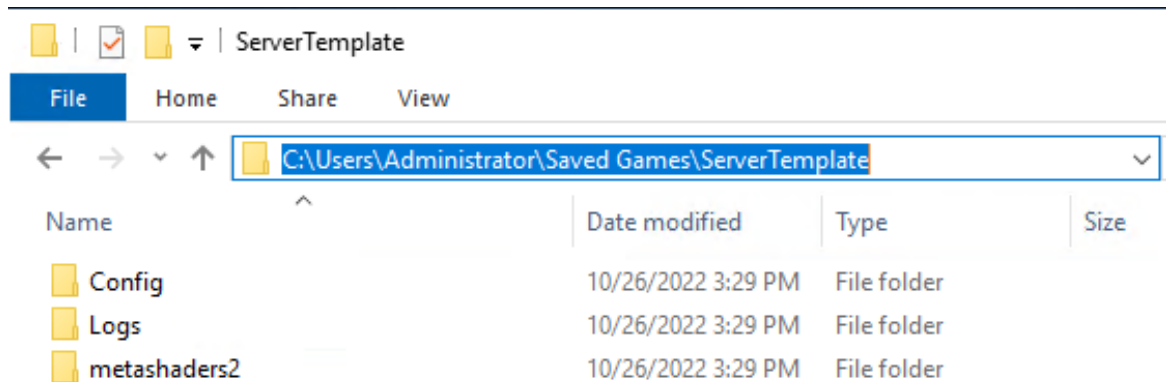
Create the following folders:
- "Data"
- "Missions"
- "Mods"
- "Scripts"
- "Tracks"

It should now look as follows:

5.1.   Setting up ServerTemplate autoexec.cfg

Enter the "Config" folder and create a .cfg file named "autoexec.cfg". The contents of the file should be as follows, each on its own line:

```
options.graphics.fullScreen = false
options.graphics.width = 1280
options.graphics.height = 768
options.graphics.maxfps = 20
options.graphics.render3D = false
if not net then net = {} end
net.download_speed = 131072000
net.upload_speed = 131072000
disable_write_track = true
webgui_port=****
```

Explanation:

`options.graphics.`
settings configure the <u>server</u> to not waste resources

`net.`
settings prevent network throttling of the players on the <u>server</u>

`disable_write_track`
prevents track files from being made on the host machine

`webgui_port=****`
the **** will be replaced with the WebGUI port for the <u>instance</u>

5.2. Setting up ServerTemplate autoexec.cfg

Enter the "Config" folder and create a .lua file named "*serverSettings.lua*". The contents of the file should be as follows, each on its own line:

```lua
cfg = {}
cfg["current"] = 1
cfg["missionList"] = {}
cfg["missionList"][1] = "C:\\Users\\**YOURUSER**\\Saved Games\\**INSTANCE**\\Missions/**MISSION**.miz"
cfg["require_pure_textures"] = true
cfg["listStartIndex"] = 1
cfg["advanced"] = {}
cfg["advanced"]["allow_change_tailno"] = true
cfg["advanced"]["disable_events"] = false
cfg["advanced"]["allow_ownship_export"] = true
cfg["advanced"]["allow_object_export"] = false
cfg["advanced"]["pause_on_load"] = false
cfg["advanced"]["allow_sensor_export"] = false
cfg["advanced"]["event_Takeoff"] = true
cfg["advanced"]["pause_without_clients"] = false
cfg["advanced"]["client_outbound_limit"] = 0
cfg["advanced"]["client_inbound_limit"] = 0
cfg["advanced"]["server_can_screenshot"] = false
cfg["advanced"]["voice_chat_server"] = true
cfg["advanced"]["allow_change_skin"] = true
cfg["advanced"]["event_Connect"] = true
cfg["advanced"]["allow_trial_only_clients"] = true
cfg["advanced"]["event_Kill"] = true
cfg["advanced"]["event_Crash"] = true
cfg["advanced"]["event_Ejecting"] = true
cfg["advanced"]["maxPing"] = "600"
cfg["advanced"]["event_Role"] = true
cfg["advanced"]["resume_mode"] = 1
cfg["port"] = ****
cfg["mode"] = 0
cfg["bind_address"] = ""
cfg["isPublic"] = true
cfg["listLoop"] = false
cfg["require_pure_models"] = true
cfg["password"] = ""
cfg["name"] = "***NAME***"
cfg["require_pure_clients"] = true
cfg["description"] = "***DESCRIPTION***"
cfg["listShuffle"] = false
```

```
cfg["uri"] = "startServer"
cfg["maxPlayers"] = "100"
```

This file is normally generated upon running a <u>server</u>, so the settings should be familiar to you and can be configured to your liking.

The following explanation will help you configure the template for specific <u>instances</u> in later steps.

Explanation for items of note:

```
cfg["missionList"][1] = "C:\\Users\\**YOURUSER**\\Saved
Games\\**INSTANCE**\\Missions/**MISSION**.miz"
```
This will be set to the path of your .miz file.

 YOURUSER & **INSTANCE** should be set to match your <u>Instance PATH</u>
MISSION should be set to match the name of your .miz

```
cfg["port"] = ****
```
This will set the <u>Port</u> for your <u>server</u>.

**** will be changed to the desired port for the players to connect to your server

```
cfg["name"] = "***NAME***"
```
This will set the name for your <u>server</u>.

NAME will be changed to the desired server name.

```
cfg["description"] = "***DESCRIPTION***"
```
This will set the description for your <u>server</u>.
DESCRIPTION will be changed to the desired server description.

6. **Setup <u>instance PATH</u> folder**

We will now use the ServerTemplate we have established to set up the framework for our instances.

-Copy the contents of the ServerTemplate folder into each of your Instance PATH folders.

-Edit the serverSettings.lua and autoexec.cfg in the "Config" folder according to your preferences. (Refer to the explanations in the previous section on setting up the ServerTemplate for assistance.)

-Add your mission .miz to the "Missions" folder

7. **Run your instance process**

At this point, you have now set up and configured your instances to all run at the same time from a single DCS installation.

Simply run the StartBat batch files that were made in step "4. Setup StartBat for each server instance" to start each Server Instance

Adding DSMC to Multiple DCS Server Instances

Adding DSMC to an <u>Instance</u> is **EASY**!!! (If you followed the instructions in 8.3.5)

Simply copy the DSMC files into your <u>Instance PATH folder</u> and configure as you normally would for a dedicated server manual install. All done.
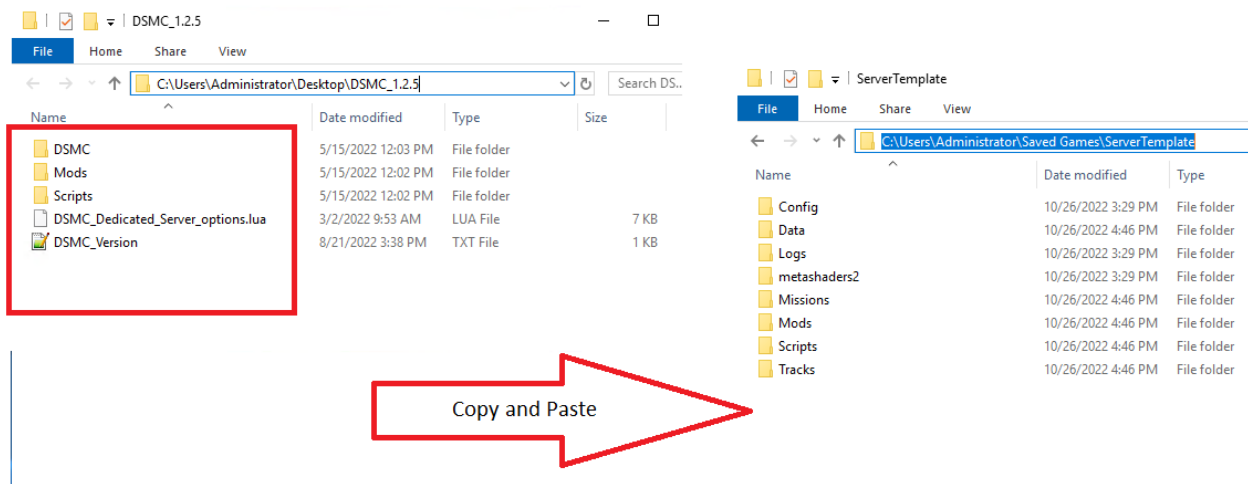
10   DSMC LIMITATIONS

Ok, here I'm trying to list some of the limitations of the mod. Basically, some are about its design or "vision" while others are about DCS code. Some, also, are due to my skill limitation.

So, this is going to be a "No, you can't" list almost, ordered per feature, as complete as possible. Let's start:

10.1 MOD RELATED

- You need your saved games directory and DCS main directory to be not read-only, and you should execute DCS as administrator. If you don't comply, the mod could fail.
- If you're using a dedicated server host or a server host with "--nographic" and "--server" custom load string enabled, you must have a desanitized server or the save process will fail. DSMC perform the desanitization automatically, while preserving possible other modification you added to the file (it does not overwrite the original file, it will search & change the code lines for desanitized);
- DSMC doesn't "freeze" or "pause" a mission: it will save an updated mission scenario
- DSMC does not add any automation or enhanced behaviour of DCS units, groups, controller
- DSMC does not support officially any other mods, except for MOOSE, CTLD (standard version and DSMC's CTLD), mist. In particular, some couple of mods that open & close the file just before mission start (i.e. weather injection tools) or overwrites missionscripting.lua have issues;
- DSMC does not keep routes or paths that you created in the M.E. or routes added during the simulation with CA or scripts or otherwise
- Once you choose your special options in the options menu, those are kept as long as you don't uninstall the mod. As said, you shouldn't even when using a server: if you don't want DSMC

to work, simply don't name the miz file starting with "DSMC". After an unistall/reinstall, options are reverted to default

10.2 SPECIAL OPTIONS RELATED

- DSMC do not load the ".ogg" files needed for CTLD for beacons usage in the first mission. Those files are, anyway, kept in subsequent saved files. Those files are stored, to help you, in the "DSMC\Files\" Folder
- DSMC does not edit DCS warehouses in real time (1): you can't add or remove aircraft, weapons or petrol from DCS warehouse using functions, scripting or otherwise
- DSMC does not edit DCS warehouses in real time (2): as I can't check warehouses content in real time, any supply logic provided by DCS won't take effects. Items will be added and removed as if the warehouse didn't have any supplies from other warehouses
- DSMC warehouse tracking can't track "fuel tank" & "misc" items such pods, side guns. The POL stored inside the fuel tank is tracked, the object "fuel tank" is not. This is a limitation DSMC can't overcome due to DCS getAmmo() function limits.
- DSMC scenery object persistence can't prevent any AI aircraft spawned at mission start to load fuel & ammo from an airbase which has all the fuel tank and/or ammo buildings destroyed. Else, right after mission start, those items or fuel tons won't be available. My design suggestion is therefore to set any aircraft "late activation" for at least 1 second.

11 MISSION DESIGN GUIDELINES

The very first thing you need to double check is the missionscripting.lua file or user write permission in the main DCS folder. In fact, **when running in a dedicated server or in a server with "--nographic" or "--server" custom string enabled, DSMC needs to have os, lfs and io library available in the SSE**. DSMC itself have a good automatic desanitization check & mod function, but it will run only if your DCS and PC user has write permission. Else, you will need to modify those strings using notepad++ or similar code editor (google about it in the ED forum to understand what I'm talking about).

In the next paragraphs I'll do a rundown of every relevant thing that mission designers should pay attention when preparing a mission or a scenario to be used with DSMC. That is cause DSMC itself does not have specific requirements and it's designed to be used almost without particular set-up in the mission editor, but users had never the "problem" to think at the "saved mission", because we don't have it provided by DCS.

After the first small & brief paragraphs, I'll go in depth with some design guidelines related to the specific features of DSMC, like warehouse tracking and automatic slot creations. These even will work without specific settings… but if you want them to work properly and with a predictable behaviour aligned with your needs, you will need to set up the scenario properly.

First, I want to specify that, in this manual, *mission* and *scenario* are slightly different things:

- A *mission* is the mission file that is going to be used, it can be the first mission out of a sequence or a mission saved by DSMC from a previous file;
- A *scenario* is the "main" mission file which you as a mission designer prepare as a base layer, used to built up the subsequent *missions.*

I'm telling this cause some of the most complicates things needed to employ warehousing or slot creation requires you to work on the *scenario* file, but once done that the first time... It's done! No effort will be required in all the subsequent *missions* generated from that scenario.

11.1 Ground groups & units

The following instructions about ground groups are more about building a future-proof scenario, and they're not a "must" today. But you can take them as good design guidelines.

- Keep ground groups in a size of 2-to-6 units each for vehicle and 2-to-10 units each for infantries.
- Try to use always the same unit type inside the group: it's much better to have 1 group of MBT + 1 group of APC + 1 group of IFV instead of 3 groups made by 1 MBT + 1 APC + 1 IFV.

11.2 Ships & Carriers

DSMC update ship position in the same way it does with ground units, **except** from the carrier group (every ship in the group included). That is a necessary compromise to be able to keep flights starting over a carrier in their proper position & parking slots.

So, carrier group won't move from a mission to another, while any other ships will. If you want to move the carrier group, it's obviously possible but will require to edit the saved mission before re-launch it.

11.3 Custom files inside the .miz file

You can add any external file inside the mission if you want, but I recommend to use winzip and NOT 7zip to pack the .miz file, because I saw a couple of compatibility error while saving due to the packing procedure, that happen only when 7zip is used for this kind of operation (like adding sound .ogg files).

11.4 Using scripts with naming conventions

One of the most common issue is about scripts or trigger that use group or unit name convention, that works like 'Pick the group named "charlie" and do things'.

Eh. All wise mission designers knows a rule: if "Charlie" might be killed during the mission, the scripts must be "protected" against code errors… like: trying to pick "charlie" while "charlie" has been *picked* several times by a couple of A-10 few minutes before. Result: SSE error, mission halted. To avoid it, mission designers do scripts with some checks before, so that the original actions can be described like this: 'if a group named "Charlie" exist, then pick it and do things'. This won't cause errors.

When using DSMC, you should follow this rule **always** for any group that use naming conventions. That is because even if the call is done at mission start… with DSMC that groups might be killed the mission before!

11.5 Spawning (MOOSE or SSE)

This is a **<u>very important point</u>**. DSMC do not know what you spawn or not, but if you spawn anything and it reaches alive the end of the mission, this thing will be saved and you will find that in the saved mission (if the spawned option is enabled).

What does this mean? That if you have any sort of automation that spawn object at mission start… **you will cumulate those objects at any subsequent start**! So, pay particular attention about this.

Since DSMC keeps the group names, this could be easily workaround by checking if that group exist before spawning another one with the same name, or else performing a "world.searchObjects" inside the spawning area to see if something is still there.

11.6 Airbase running out of fuel

If you use functions that spawn aircraft from bases, such as MOOSE dispatcher, you will need to check if that base has enough fuel. Sadly, that is not possible in DCS.

To prevent issue, with the help of Pikey, I added a small check to DSMC. In the scripting environment you will find a table called EMBD.airbaseFuelIndex. This table will be populated with the name of the airports once a birth event happens and the aircraft spawned has no fuel. The aircraft will be immediately removed, you will find a couple of messages into the dcs.log file and the table will be added with the airbase name.

i.e. if a Mig-29S try to spawn at Gudauta, but Gudauta don't have fuel, the Mig-29 will immediately disappear and EMBD.airbaseFuelIndex will be like this:

EMBD.airbaseFuelIndex = {

    ["Gudauta]" = true,

}

PS: important thing. When you spawn aircraft with fuel tanks, **those are fulfilled with fuel even if the airport is at 0 tons**.

11.7 Resource management: logistics comes into play

If you like logistic, this is going to be a key point. As said, all supply operations will be performed at the end of the mission. First thing: **DSMC won't resupply airframe**. It will resupply jet fuel or any type of ammunition, but no aircraft will be resupplied. That is by design decision.

There are 3 ways to resupply an airbase or a heliport:

- By landing there (this will also "move" the aircraft item): everything that is on board the aircraft, fuel & munitions, will be added to the warehouse;
- Using the supply chain, described below.

What are the differences between those 3 solutions?

Timing: Landing somewhere will add items & fuel to the warehouse both using DMSC and the Resource Management System of DCS. That means that resources added by landing are available immediately to any other user in that airbase/heliport.

Instead, both airlift crates and supply using the supply chain in DSMC happens at the end of the mission: first the airlift, then the supply chain. Items & fuel will be available in the subsequent run and not real time.

Amount of items & fuel: Both landing and airlift crates adds item and fuel to the warehouse. By adding them, they also "set" new required values of the warehouse. To make it easier: if I land in a heliport that currently has 10 tons of fuel, and I add other 2 tons, in the subsequent mission I'll have 12 tons of fuel… and when depleted, the heliport will try to replenish all the necessary fuel to go back to 12. To 12, not 10! That means that by using airlift crates or landing, you're also increasing the airbase/heliport operational capabilities.

Beware: the "increase capability" take effects only in those cases:

- By landing, **if no items were used** in that airbase/heliport in the same mission;
- By airlift crates, always.

Ok, that said… you know what means to land somewhere (I presume, else… you might probably check flight manual before this one).

But WTF is the supply chain?! The supply chain is, by order of magnitude, the most powerful way to replenish used resources in airbase/heliport. Basically, it will check the number of items depleted during the mission and try to find a supplier that has them. If found, those items are removed from the supplier instead of the airbase, that will stay on its initial values. This is also recursive: If the supplier has a supplier itself, it will remove the items from the highest level available.

BEWARE: here it comes the only real risk: do not set a loop. If you set the warehouse A to be supplied by B, then B to be supplied by C.. and C finally supplied by A, you will have a loop. Which leads to a DCS mission crash once you save, no saved file, etc.

Then, who is "supplier" of the airbase? Those that are set as supplier in the mission editor. Yes, I made it simple.

But since it's not possible to track the RMS of DCS, to be consistent with fuel & items you must accept a compromise: **no real time resupply must be used with the RMS**. To be sure that that's not happening, every mission save the timing/size/speed of the RMS resupplies are minimized to prevent them happening.

Let's dig a bit into how the supply chain works.

11.7.1   Warehouse objects

DSMC don't require any particular setup to define a warehouse object. It uses all the inbuilt DCS info: if something is a warehouse in DCS, it will be in DSMC, along with its resources information.

Therefore, any airport/FARP/Oilrig and so on are warehouses. Any ship with helipad or carriers are warehouses. Any FARP resupply object, are warehouses.

Also, any static object of "warehouse" category is a warehouse: you will recognize them because instead of the classic icon, you will see the warehouse icon for fuel or ammo:

Standard Static object — Fuel tank Warehouse object — Ammo depot Warehouse object
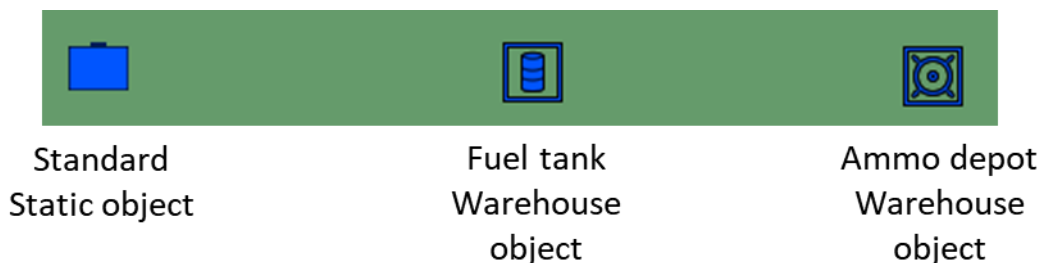
Remember, DCS is a sandbox: Every time you place a warehouse object on the scenery, it can be everything: a fuel tank (with that icon) can be an ammunition supplier, unless you manually void it. So, as

per units' placement, flight, etc... **it's up to you to set up that correctly**. Sadly, if you copy & paste a warehouse in DCS, you still have to manually set its "warehouse content" table.

The warehouse properties you need to know are only 2:

- Amount of items, that can be a number that you set or "unlimited";
- The "size" value.
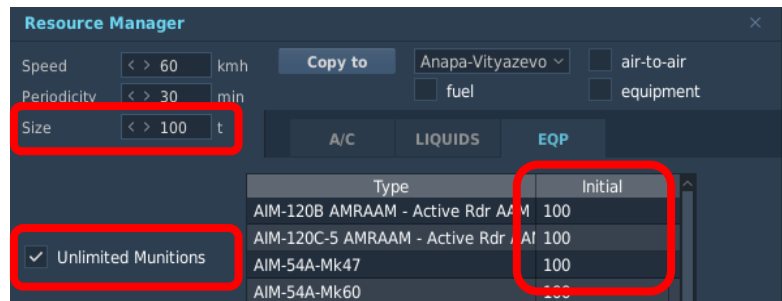
These parameters define the amount of items available and how much of it can be delivered each mission.

11.7.2 Supply mechanics

If 3 aircraft consumed 10 tons of fuel and 6 AIM-120B, then the airport where they operate will check for available suppliers of the same coalition (that you set into the mission editor).

DSMC will check the best suppliers: one that can resupply all the quantities or the biggest part of it. If you set a warehouse (or another airbase) with 40 tons available, then DSMC will transfer the 10 used tons to the airbase, removing them to the supplier. If you set a warehouse with 6 tons, DSMC will take all those 6 and will also remove 4 from the airbase.

If one supplier has fuel, and another the AIM-120, DSMC will take resources from both.

Since DSMC do not track info about "first mission situation", you surely want to carefully track the initial amount of an airbase resources, because you can supply them only as explained before.

11.7.3 Basic vertical supply chain example

In all the examples below, I'll use fuel (jet fuel) as main resources as example... cause it's easier to explain the mechanics.

In the mission editor, you can set a supply net, like:

Airbase 80 t	→	Wh «A» 20 t	→	Wh «B» 500 t	→	Wh «C» 300 t

Airbase is the place where your flights consumed resources. The subsequent "Wh" can be another airbase as well as any warehouse object. Let's say that they are three fuel tank objects where the only resources available is jet fuel (no munition, aircraft are ignored by design).

In our example, Airbase consumed 5 tons of jet fuel at the end of the mission.

Since all of them has sufficient jet fuel, <u>all the tons will be removed from warehouse C</u> cause that is the "mother" of the chain: in fact, at the end of the mission this will happen:

- Airbase require 5 tons from "A". "A" has them, so they move 5 tons to airbase (you won't see them depleted in the airbase);
- Warehouse "A" require 5 tons from "B", same thing;
- Warehouse "B" require 5 tons from "C", same thing;
- Warehouse "C" does not have suppliers available: the 5 tons will be depleted to it;

End situation? This:

Airbase 80 t	→	Wh «A» 20 t	→	Wh «B» 500 t	→	Wh «C» 295 t
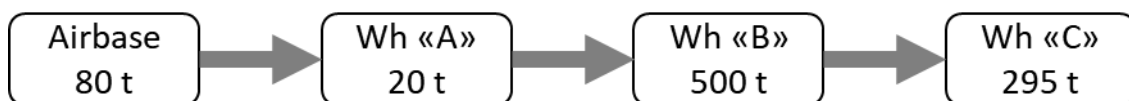
That is the key of "vertical" side of the chain. But… what happen if the airbase consumes 50 tons?

- Airbase require 50 tons from "A". "A" do not have them, so they move the maximum amount available (20 tons) to airbase. The remaining amount, it will be depleted from the Airbase.
- Warehouse "A" require 20 tons from "B", same thing;
- Warehouse "B" require 20 tons from "C", same thing;
- Warehouse "C" does not have suppliers available: the 20 tons will be depleted to it;

Final situation:

Airbase	Wh «A»	Wh «B»	Wh «C»
50 t	20 t	500 t	280 t

Note the "50 tons" at airbase, which has 80, while warehouse "C" took the 20 tons transferred by warehouse "A".

What does this mean? <u>That you can set a limit of the expendable fuel per mission</u>, which is given by the warehouse that has less resources. If you use more, this will be depleted from your airbase… and to re-charge that, you will now be forced to land something on it.... which is obviously much harder to do.

11.7.4   Basic horizontal supply chain example

Obviously, you can also set horizontal supply. What does this mean? That an airbase can have multiple warehouses to check for. Let's say this:
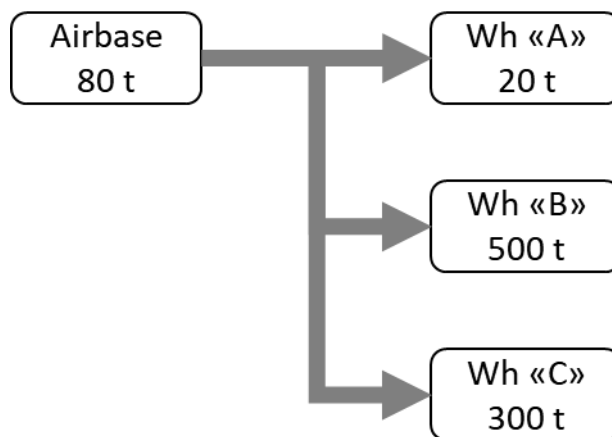
In this case, DSMC will use the better suited warehouse, which is "B" because it has more fuel than "A" and "C". This will be the result:

Airbase	Wh «A»	Wh «B»	Wh «C»
80 t	20 t	450 t	300 t
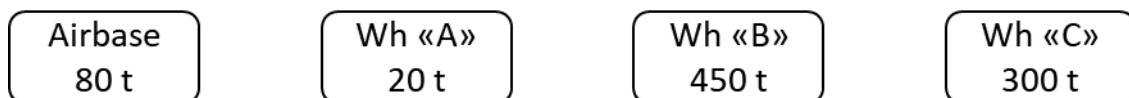
So, what does DSMC not allow? Multiple resupplies to the same base from the warehouses. To explain that easier: DSMC will choose the supplier with more fuel, but only that. In this situation:

Airbase	Wh «A»	Wh «B»	Wh «C»
80 t	20 t	10 t	15 t

DSMC will use full "A" warehouse (cause it's more than B and C), but will also remove 30 tons of the 50 you asked for from the airbase. It won't remove anything from B and C. This will be the result:

Airbase	Wh «A»	Wh «B»	Wh «C»
50 t	0 t	10 t	15 t

11.7.5   How many items/tons a warehouse can deliver?

DSMC makes it simple: any warehouse can deliver an amount of fuel or ammo proportional to the "size" parameters that you can set in the mission editor (standard value is 100, when you create a new object).

Fuel: same as "size", as jetfuel tons;

Dumb weapons: same as "size", as items number, for each type (that means 100 S-8 rockets *and* 100 Mk-82, for example);

Smart weapons: 1/5 of the "size", as items number, for each type (so if size is "100", it will be 20 of each weapons type);

BEWARE: In the end, there a situation you MUST avoid at all cost. As explained in the vertical chain, DSMC supply chain function will search the best supplier at each step.

But if you accidentally create a loop, like:

Airbase X → Wh A → Wh B → Wh C → Airbase X

That will lead to a DCS & DSMC crash due to endlessy resupply run. So, check this condition very well. I know, I already wrote this… but this time twice is better than once.

11.7.6   Production sites, coalition depot

So… if you get the basic examples I wrote above, you should easily understand that you can build a couple of interesting warehouse items without having to code anything.

You can create coalition depot of a resource (ammo, fuel) and also create production sites!

Coalition depot

Using fuel as example, to create a coalition depot you can simply create one (or more, in proximity) "tank" warehouse object and place them where you want on the map, filling them with a high value of jet fuel only and no ammunition. Let's say you place 3 fuel tanks in proximity with 1000 tons of fuel each. Now you set the Tank #001 to be supplied by Tank #002 and Tank #003. Now you have a depot of 3000 tons of fuel.

Then you can set all your airbase to have low fuel amount (let's say 100 tons) and Tank #001 as supplier. Every consumed fuel in those airbases will be removed from your depot, since it becomes empty.

What's the point? To have the depot as a valuable target! If enemy destroy the depot, then you will have your resources depleted.

Production sites

To create a petrol production site, you must simply place a warehouse object and set it as "unlimited". A production sites won't require suppliers, but it's still limited! In fact, unlike for deposits, it will be capable of resupply only the amount of item (i.e. petrol) defined in "size" value. If you create a tank object set as unlimited with size equal to 80, you will have something as a "refinery" capable of creating endlessy 80 tons of fuel each mission.

11.7.7 Standard supply net

First: the standard supply net can be used automatically activating the *DSMC_WarehouseAutoSetup* options in multiplayer, and it's active by default for single player. That means that if you're using DSMC in single player or host with graphics, **you must build a scenario that follows these supply rules** or else that is not using warehouse tracking system to prevent unexpected behaviour and supplier changes.

Second: standard supply net, and its activation, **do not change the content of the warehouse**. So, while this option will provide an automatic update of all the supply net without your need to create it in the mission editor using all the possible combination since start, you still have to perform an accurate initial setup of your scenery before launching "mission 000" that comply with the basic scheme explained below.

What autosetup does: this feature works after all units & static objects have already been updated, so it takes action when all the warehouse static objects had changed coalition if necessary.

The autosetup use some fixed rules that you need to consider when building a new scenario if you want to take advantage of this option:

- Airports is recognized as "customer" and can only be supplied by depots, but they're not supplier of anyone;
- The very standard FARP object (with 4 helipads) is recognized as "heliport" and therefore will work the same as airports;
- All the others heliport objects (single helipad, invisible, oilrig, etc) won't be set as customer and won't ever have any supplier;
- Warehouses with the shape of a tank will be recognized as fuel depot;
- Warehouses with the shape of an ammunition dump or a standard warehouse will be recognized as ammo depot;
- Warehouses (of any kind) that are set with unlimited fuel or unlimited weapons will be recognized as refineries/ammunition factories.

So, it appears important that setting up the scenario the mission designer will follow these rules:

- All warehouse objects must be set with their content consistent to their shape. It's also important that the other items column will be set as void. For example: set 0 to each aircraft and ammo for the fuel depot, and vice versa for the ammo depot;

- If you set a depot as unlimited in its main item, this will be recognized as production site (with will have unlimited supply over time, but limited supply for each mission!).

The standard supply net for fuel is built following this scheme:
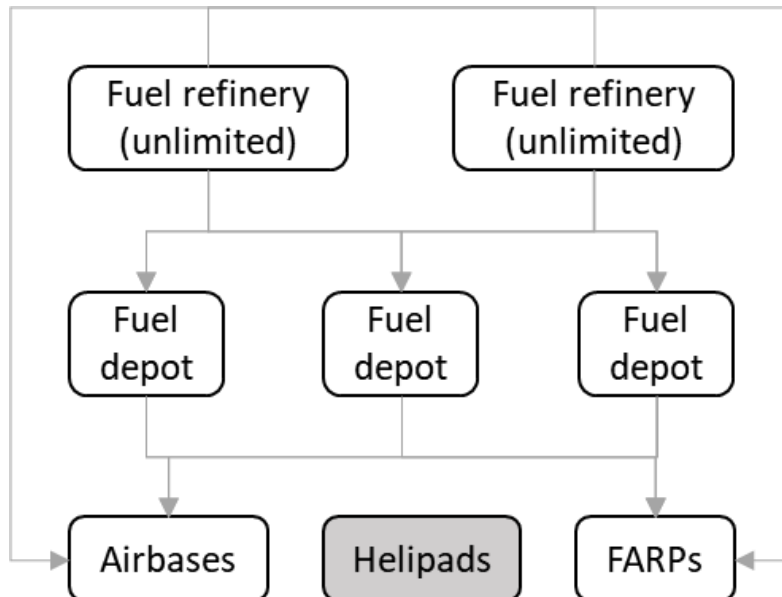
Ammo supply net is the very same as for fuel. The scheme follows these rules:

- Customers (Airbases & FARPs) are supplied by both depots and productions sites;
- Depots (warehouse, not unlimited) are supplied only by production sites;
- Production sites (warehouse, unlimited) does not have suppliers.

The automatic supply chain setup therefore provides an update in each mission where the available suppliers for each coalition are effectively used in the supply net.

11.8 Automatic helicopter & planes slot creation

DSMC is designed to fulfill two types of service: allow virtual squadron's mission designer to be able to design a sequential mission campaign with much less pain than usual… and allow 24/7 servers to create a continuously developed scenario.

These very different conditions have very similar needs: have an updated scenario, don't mess up with AI flights… and feature updated available client's slots.

As discussed in paragraph 8.2.3 DSMC provide automatic slot creation based on the airbase/FARP warehouse content, **with up to 8 slots for each flyable aircraft** (8 single-ship flights). But now it comes the unwritten rules of DCS that you must deal with.

When you uncheck the "unlimited aircraft" checkbox, each aircraft type (flyable or not) will be set with 100 items… unless you change using the "Fill Col" button below the aircraft list. That is **not** good, cause since there are many flyable aircrafts, this means **a lot** of slots for each airbase/FARPs that is set properly. We're talking about one hundred of them for each base. So, you must manage this situation, that comes down to two cases:
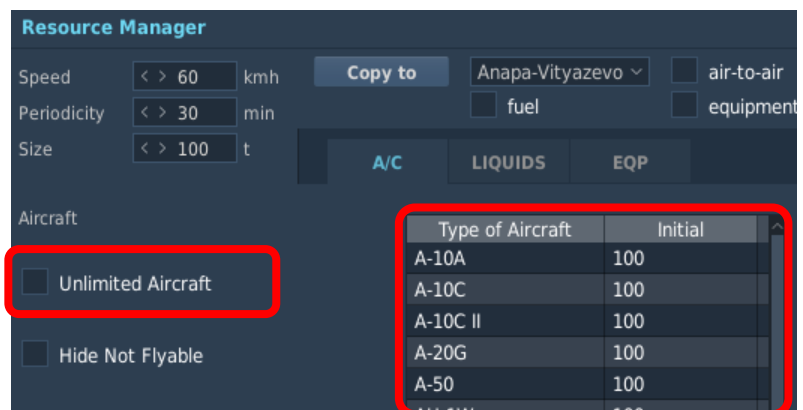
- In each FARP or base that does not have multiple parking positions, it will eventually generate all the possible slots. There are no limits, and we're talking about hundreds of them;
- Instead, in the airbases you can have thousands of aircraft in each airport warehouses, even if it has 10 tiny parkings… and here it comes the problems.

It's easy to understand what is the problem with this situation. Try to answer this question: *if DCS has at least 30 flyable aircraft, each one is set to "100" for a total of 3000 potential slots available, but I have only 42 free parking spots… which one should I create?*
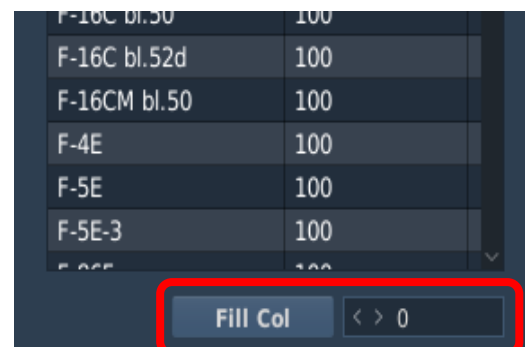
This awful question **has no right answer**. It's on alphabetical order.

Also, another issues, is about exiting clients and AI slots (flights): what should I do with them?

We need rules to make this suitable. Those rules are something that I don't like, cause people (*yeah, you included*) usually do not read rules… and even more rarely comply to them. But I trust you, and I tried to make those rules as easy as possible.

So, **those are the rules that applies** to any airbase or heliport when using those options:

- Ships are ignored;
- The warehouse of the airbase/heliport must have the *Unlimited Aircraft* checkbox unchecked. Unlimited warehouse will be ignored and considered unsuitable;
- The airbase must not be set as "Neutral";
- All slots (groups with client's units) on each suitable airbase will be deleted;
- Slots **will try** to add up to 8 slots for each flyable aircraft that has sufficient items in the warehouse.

I wrote "will try" cause, in heliports objects, slot creation will eventually happen, and it's only up to you to avoid having 8 x slots x each flyable type (something less than 100 slots today) by leaving that "100" items each default. Instead, in airports, we have to deal with parkings… and it's not that straightforward! this is, in short term, the procedure done by DSMC for every airbase or heliports with parkings (those present in the scenery):

- Check if the warehouse is not set as "Unlimited Aircraft";
- Clean every client slot;
- Check available parkings by "removing" from availability all those used by AI flights;
- Loop through any flyable aircraft and try to assign available parkings that are compatible for aircraft type & size, couple by couple, till the higher value between warehouse items and "8" is reached. For each couple verified, a couple of slots (one flight) will be added;
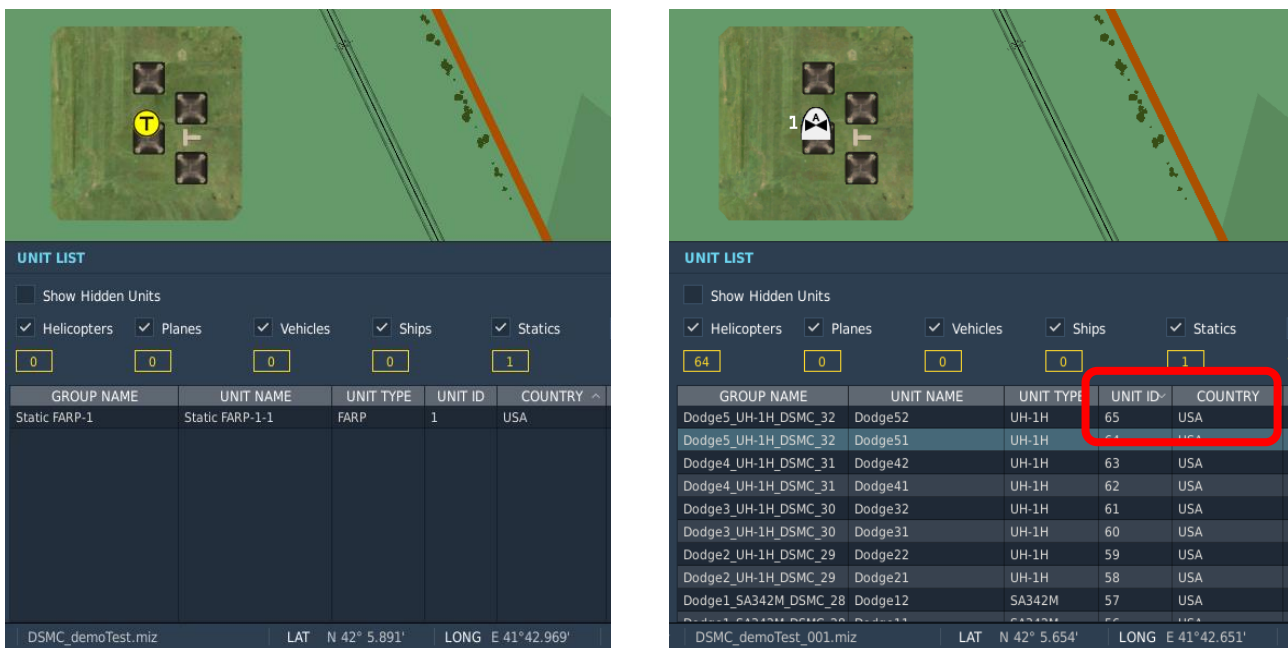
- Once no available parkings are left, no slots will be added anymore. So… the first are found in the "loop", the first are created. The others… **might not**.

Now that you know what are the rules, let's see how to bend them to your needs.

11.8.1 Heliports setup

Heliports can be divided in to 2 types: those you manually add using the mission editor and those that has been created using special versions of CTLD (FOB in the DSMC's CTLD version).

To demonstrate what happens when you add a heliport if you don't change anything but unchecking the "unlimited aircrafts" checkbox, I created a new mission with only a 1 heliport, south of Poti, and the box unchecked. Once started the mission, I waited few seconds and then save it. Left & right pictures are the before & after save.

Yeah, 64 helicopters slot plus 1 heliport object. Heliports does only generate helicopters (sorry, harrier), and at the date this manual is written there are 8 flyable helo types. When they will be 10, there will be 80 slots and more.

To have a properly scenario, you should limit the clients slot in each heliport. This is done in the simplest way, by making available only the type of helicopter you want there. Let's say you want to have Ka-50 only in that airport. You will do this:
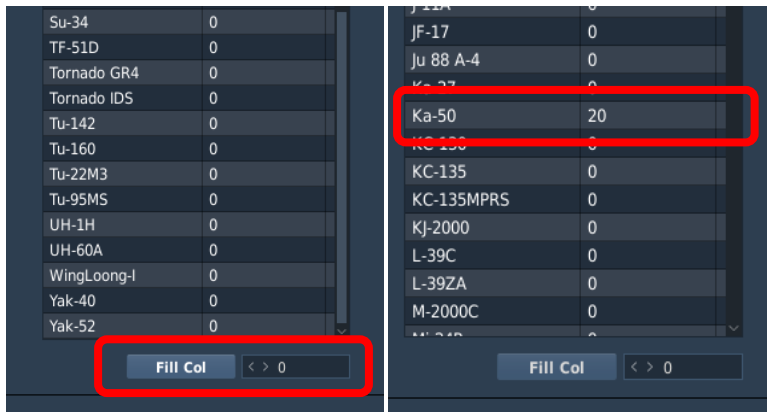

With this setup, restarting the mission and re-making the save, you will have 8 slots of Ka-50, plus the heliport objects.

You can set the initial amount of helicopters that you want to have, up to 1000. Obviously, if you set an amount of "4", you will have only 4 slots.
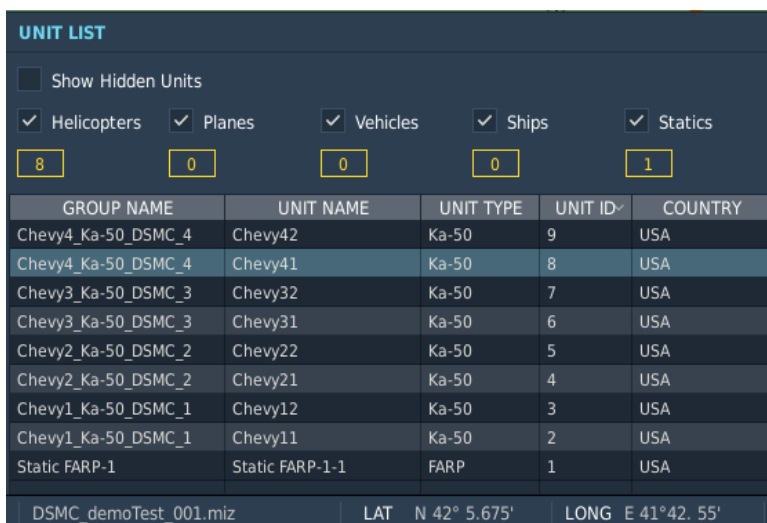
If you use more than one type of aircraft, slots will reflect your choice. For example, you set 6 Ka-50 and 4 Mi-24P, you will find these exact values (see picture here on the left).
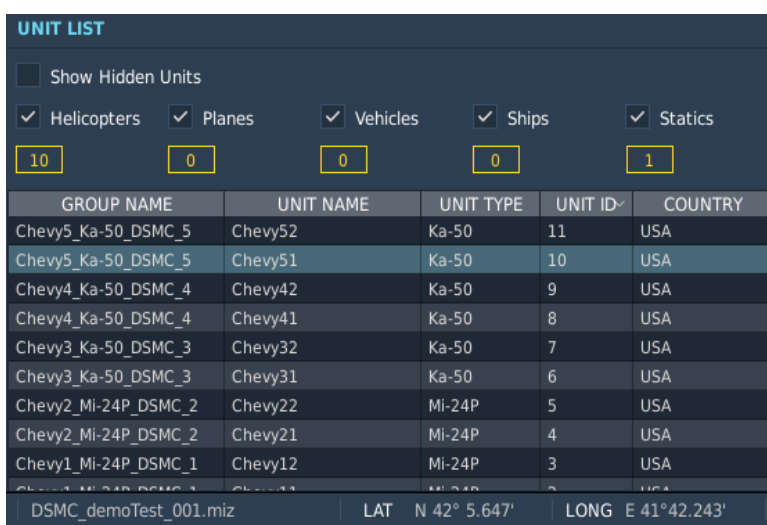
Another important thing: the country of each slot will be the country of the object owning the heliport. This is important, cause when you conquer a heliport, the coalition will change and therefore also the created slot will.

FOB's heliport could have two different behaviors. When spawned, they will be "unlimited" by default and this is a DCS thing. But, once you save the mission, if *resource tracking* is disabled, they will be

"unlimited" and therefore no slot will be added. Instead, if *resource tracking* is enabled, newly spawned heliport will be "zeroed": they will have 0 fuel, 0 munition and obviously 0 aircraft. And they will be set to limited, in every aspect. Anything that happens in the very same mission you spawned them, will be calculated after the "zeroing".

For example: if you create a FOB, and then land there with a couple of Mi-8s, you will have a FOB with exactly 2 Mi-8s, the fuel in these helicopters (plus few tons by default) and the exact munitions carried by the helos.

11.8.2 Airbase setup

Airbase comes with the same principle than heliports, except for a couple of things:

- Every flyable module is considered (except mods);
- There are parkings! 1 slot for each park position is allowed!

Those two little details mean… a revolution. If you leave the aircraft table to its default of "100" items each, this *bad thing* will happen:

Only Mi-24Ps! Why? **Cause we have only 10 parkings in Batumi**, so if we have 100 items for each airplane, the first "flyable" things DSMC found will definitely fill any slot. This make not "optional" but **necessary** that you choose wisely in the very first mission what you exactly need from that airport.

Important note: DSMC will register landings if you have the resource tracking active, so you might want to keep some parkings free.

I'm going to make it much less complicate by a couple of step-by-step rules (suggestions):

1. Use different airports for AI & client's slots. This is not mandatory, but will simplify a lot the following steps (because you won't need to take AI slot in account). Choose airbases with many parkings for client's flights;
2. Take the maximum parkings value from the mission editor and then subtract 4-to-8 of them as reserved spare. The result will be our *available parkings*;
3. Think like it's real: if you have an airbase with 50 parkings, how many airplanes could stay there? How many airplanes you expect to be in one squadron? Set the warehouse to be sure that you won't have more aircraft items than available parkings!

If you follow these rules, you'll be ok. Let's follow them.

Rule **1**: I decided to use Batumi for AI support flights, because it has few parkings. Client's slots will be generated in Vaziani, which have plenty of slots (92). How can I know this? With an easy trick: go to the mission editor, add a flight of 1 **small** acf, like A-10A. Set the first waypoint to "Takeoff from ramp" and then… look at the PRK list:

The highest number will effectively be the parking number.

PS: there are some airbases with different parking names, like Nellis in Nevada… with them, you have to be patient and do the math manually. If you really need that. Remember, you can leave even 100 spares if you want.
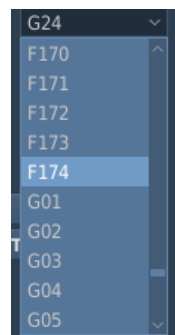
Rule **2**: Since Vaziani has 92 slots, I'm going to be generous with the reserved spares and I'll give them 12, so I can use 80 slots for clients.

They're a lot? Yes, but I can effectively use less. In the meantime, rule 2 is done: no more than 80 items in the warehouse!

Rule **3**: In my scenery, Vaziani is going to be the airbase for A-10C and F-16C block 50. I think that 24 F-16s and another 24 A-10Cs could fit my needs, for a total of 48 items in the aircraft warehouse:

Type of Aircraft	Initial
A-10A	0
A-10C	24
A-10C II	0
A-20G	0
A-50	0
AH-1W	0

Type of Aircraft	Initial
F-16C bl.52d	0
F-16CM bl.50	24
F-4E	0
F-5E	0
F-5E-3	0
F-86F	0

This is what DSMC will do once you save the mission:

18 total units, cause 2 were already there (ground units): 8 slots for each aircraft type, F16s and A-10s, 16 client's slots.

As you can see, only the first unit in each flight will have the proper icon, and if you try to move them in the mission editor both parkings will be re-assigned by DCS. That said, all slot works properly.

If you need more than 8 slots per type, you have 2 solutions: the first is to use more than 1 airbase for that type, the latter is to contact me on the discord server and ask what harcoded parameter could be changed.

11.8.3 AI slot management

Ok now we know how to properly set-up automatic slots creation for clients. But let's say that I still want to "limit" my AI aircraft items, even for tankers and AWACS or else. But if I set my warehouse both with "unlimited aircraft" unchecked and in any coalition but "Neutral", it will try to generate slots. How can I avoid this?

Very easy answer: set the warehouse content with all aircraft to "0" except those you need to be there in the proper quantity, just like you do for clients.

If you set an airbase to have only E-3S or KC-135 or else F-15E, no client's slot will be generated cause these aircraft are not flyable type. Obviously, here it comes the most important limit of automatic slot creation: **you cannot prevent slot generation for flyable types**.

Ok then, how can I have my F-15C AI flights to start from (Any airport name)? You don't have to change anything. DSMC won't interfere with AI aircrafts, both if you add them in the mission editor or dynamically spawn them by script. You only have to be sure that:

- There are some spare parkings! AI will use it to spawn aircrafts;
- There are enough items in the warehouse! If you set 8 F-15C in an airbase, and you have 8 client's F-15C in flight from there… well, DCS will try to spawn your AI's F-15Cs… but they won't be available in the warehouse. So, they won't spawn.

11.8.4 In-flight clients & AI spawning aircrafts

Just one line in this manual to say that if you add any aircrafts from the ME or by scripts and set them to start already in flights, by "turning point", they won't be filtered in any way.

This can be a very reliable solution to be sure that one or more specific flights won't miss the scenery in any possible case.

BEWARE: some scripts make AI aircrafts to start in flight but still "using" airbase warehouse: in that case these aircrafts still need to comply to airbase limitations (items, parkings).

CREDITS

If DSMC works as well as you can see, you have to congrats to all the testers that tried it knowing that it wasn't working yet and tirelessy reports any strange behaviour, bug, or even compatibility issues with MOOSE, CTLD and MIST.

Else, if DSMC doesn't work as expected, here you can find a fairly accurate list of everyone actively helped me or tried to make this work: fell free to blame them. Not me obviously, I'm clearly innocent.

SPECIAL MENTIONS

https://simitaliagames.com/

The day I started asking for "beta testers" I never imagined I could find an entire community of DCS simmers ready to launch the last DSMC built on their 24h/24 servers. SIG_Webber, who manages the servers, was fantastic both in willingness to test but also in reporting every anomalous condition in detail.

One of the most dedicated tester is alt.sanity, he tested DSMC extensively, with particular focus on helicopter operations. He also helped a lot with dedicated tests aimed to reproduce certain undesidered behaviour or bugs.

TESTERS

Besides of the communities above, a lot of effort to make DSMC works has been done by all testers, which tried the mod both in single player and multiplayer and make me aware about many issue that could be solved before the release. Here a summary list of those who directly reported as testers during the "beta" stage, by nickname:

- Dax;
- Enkas;
- ESA_Furia;
- Gh0st
- Neon;
- Panthir;
- Paranoid;
- PVI_Eagle;
- Gunny;
- Frosties;
- KaptinKaos;
- [SG] Idefix;
- Special K;

MASTERS

If DSMC even exist is due to the skills that the following people gave to me, directly or even by reading their code. Many of them is always there for sharing knowledge, and that is the core of a working community. From some of them I simply reverse engineer their code to learn, and rebuilt them as I needed.

So, many many thanks to:

- Rider;
- Ciribob;
- Grimes;
- MBot;
- Pikey;
- Pravus;
- Speed;
- Xcom;

VERY SPECIAL ONES

There are at least two very special person I would like to thank for their patience and dedication to this project:

- My wife[5], Nicoleta. She always support me with her infinite patience, jokes, fantastic cooking skills, endless love;
- You, cause if you're reading here it means that you care about this small project very much. And even if there will be bugs and sometimes issues to be solved, I know that I can count on you to make DSMC better every day.

DSMC MOD SUPPORT (OR THANKS)

I made this mod for passion, not money. I'm not a software designer, I don't plan to become one, because I love my current job. I put an incredible amount of hours in coding DSMC (in the range of the thousands), but that is mostly cause I started that I didn't knew what means "local", for who could understand, and I had to re-done it at least three time.

I'm already working on the next step... it might take years, maybe less, but I'm not in a hurry. That said, this mod-creation hobby takes hours, usually night hours, so if you'd like to help me completing you can do it by giving me a coffee!

How? I set up a paypal donation and you can add 1 € (the standard espresso here in Milan). If you really liked the mod and you find me to deserve this additional coffee, then click here.

---

[5] Yeah that's the most valuable update to DSMC: we got married on 4th June 2021 :P