USER MANUAL

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:

- 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 relates to mission design?

We hope to answer all those questions in the next pages. Obviously, we strongly suggest reading the **Quickstart guide** first, because, there you will find the basic information about how to run and customize DSMC according to your needs. That is the drive school.

Here, instead, 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.

If you don't feel the need to dig inside the mod structure, history & geek details, you might jump right to the "What are its limitations" chapter!

WHAT IS DSMC?

First of all, it's D.S.M.C., not DSMC. 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 outside. 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 is small, written in Arial 16 to help who can't read that much, to fill void spaces and to let you print in A5 format if you like. But 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.

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 it's never done during the simulation unless you call it by choice using F10 menu options. 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. Sadly, this function is only available if you decide to run DCS in "desanitized mode" editing the missionscripting.lua file in your main DCS folder (google it in ED forum to get details).

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.

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

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.

OPTION MENU… OR OPTION FILE?

We talked about the options you could choose to use. Well, there are two ways: one is for standard usage, the other is for dedicated server mode only. The two options do not update each other, they are separate configurations, so please pay attention; if you intend to change from self-hosting and GUI options, to dedicated server later, then, you must edit the options file.

For standard usage DSMC, there is a special options menu inside DCS. Check the User Manual for details.

For dedicated server mode, DCS removed some useful things like mission editor modules... and main menu. Therefore, when you run a dedicated server, DSMC won't be able to read the options menu. To solve this issue I created a file, that is named very obviously:

DSMC_Dedicated_Server_options.lua

That is placed directly inside your Saved Games\DCS.whatever\ directory. If you run a dedicated server, use that to decide what should be activated or not.

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 files that perform actions inside the mission environment are "EMDB_inj.lua" and "TRPS_inj.lua". The first is the data collector for saving activity, the latter is the CTLD modified code that is loaded only if you activate the corresponding options... we will talk about that later.

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.

WORKFLOW: TRIGGERING 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:

- Get’s the “save now” order
- EMBD_inj took all the events populated tables (logistic, deaths, etc) and saves them
- EMBD_inj took all the units’ position & life 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, naming it “(DSMC_yourmissionname)_001.miz”. If the mission already have the 3 digits sequence at the end of the file, it will add 1 to it. Like if you start a mission called *DSMC_test_024.miz*, the saved file will be *DSMC_test_025.miz*
- 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, it's something between 3.5K and 4K lines of code plus 7K+ for the CTLD clone itself (which is 99.9% Ciribob work)... 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*]

So, what is really relevant? To understand WHEN those complicate things happen. The save procedure is "called" in three different moments:

- Manually using the "F10" radio menu[1]
- When the last client in a server has disconnected leaving the server empty (only in dedicated mode)
- When the simulation is closed, as long as the autosave option is enabled
- When DCS is started again after a crash, if the autosave option is enabled.

If you run a server with desanitized environment and autosave, you can:

- Do a manual save
- DSMC will save anyway on last client disconnect
- DSMC will also save again when DCS stops or crashes

[1] The F10 save call can be performed by ANY client.

The purpose is to have the most updated file possible, so **please understand that each save process will overwrite the previous** one: you can only have one saved mission each time. 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

A simplified matrix of save process call is below. Obviously, we are talking from the player/host/server point of view: clients won't need anything.

		Autosave feature setting	
		Autosave option disabled	**Autosave option enabled** *(desanitized DCS)*
DCS execution mode	**Standard DCS execution; single player**	F10 manual activation	F10 manual activation Autosave
	Standard DCS execution; multiplayer host	F10 manual activation Last client disconnect	F10 manual activation Last client disconnect Autosave
	DCS execution with specials: "--server" or "--norender"	F10 manual activation Last client disconnect	F10 manual activation Last client disconnect Autosave
	Dedicated server executable	F10 manual activation Last client disconnect	F10 manual activation Last client disconnect Autosave

As a dedicated server administrator, you can also choose to disable the F10 manual save option, disabling the corresponding option that is available only in the *DSMC_Dedicated_Server_options.lua*.

The same option isn't available if you run single player or if you host a server using the standard DCS execution, because it is believed that is preferable to let this option always available in those situations.

WORKFLOW: WHERE ARE MY SAVED SCENERY FILES?

This time the answer is truly easy. All saved mission 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:

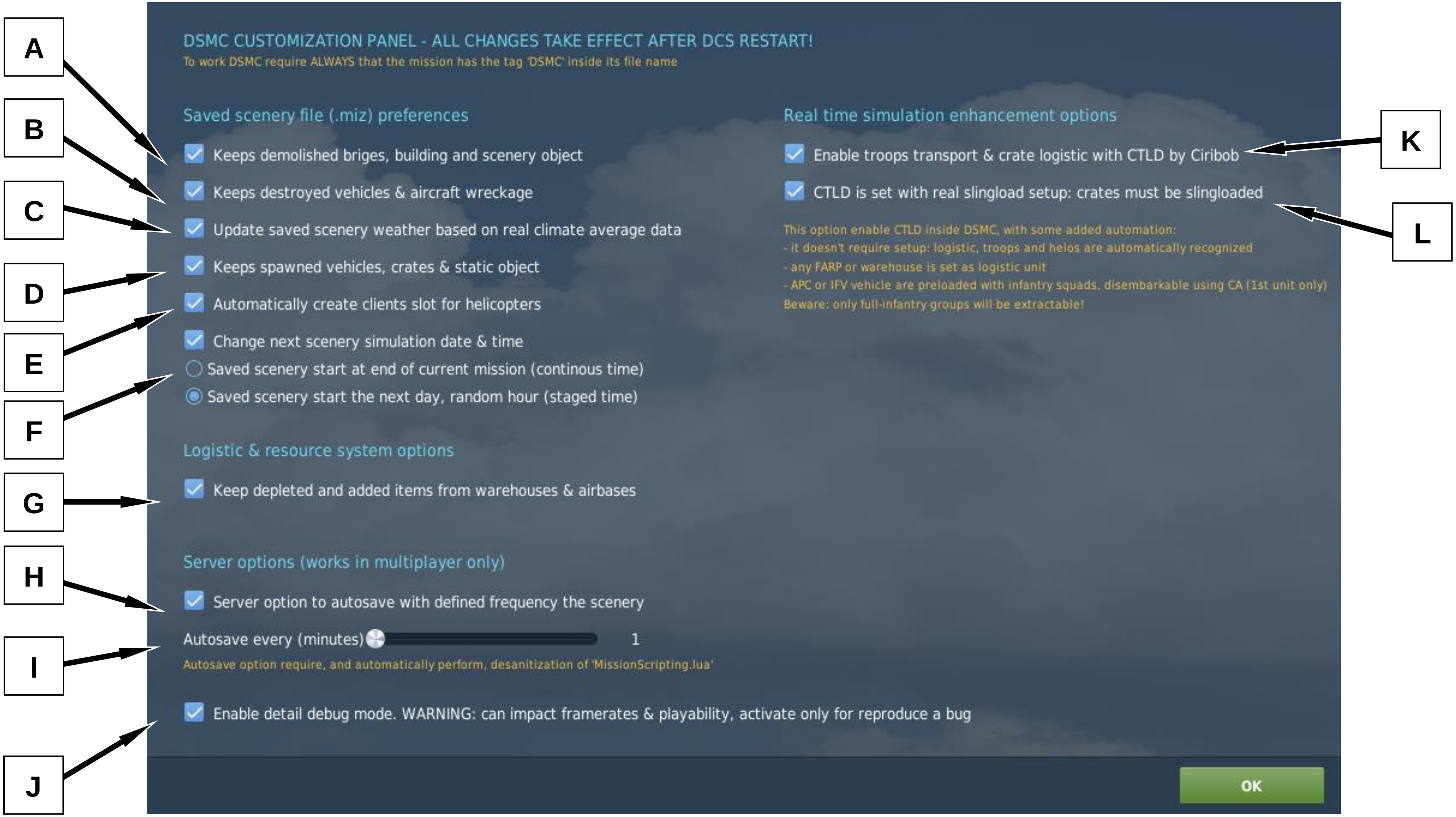
1. Windows won’t let you do that. While DCS is running, the open .miz file is considered “in use”, therefore a VRS error will be prompted in dcs.log and nothing will be saved (that’s a very good reason, you know);
2. Cause if something goes wrong…I bet you would prefer to have the original file to work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

OPTIONS EXPLANATION

In this paragraph I'll try to explain as simply as possible how you should assume that each option is going to work.

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!

Next page you will have a picture of the in game options menu, with all the options included there highlighted. As said in the previous pages, server options are handled using a lua file that features a little difference: you won't have the updated position of a landed flight while you will get in exchange the possibility to remove the F10 manual save option.



DSMC CUSTOMIZATION PANEL - ALL CHANGES TAKE EFFECT AFTER DCS RESTART!

To work DSMC require ALWAYS that the mission has the tag 'DSMC' inside its file name

Saved scenery file (.miz) preferences

☑ Keeps demolished briges, building and scenery object

☑ Keeps destroyed vehicles & aircraft wreckage

☑ Update saved scenery weather based on real climate average data

☑ Keeps spawned vehicles, crates & static object

☑ Automatically create clients slot for helicopters

☑ Change next scenery simulation date & time

○ Saved scenery start at end of current mission (continous time)

● Saved scenery start the next day, random hour (staged time)

Logistic & resource system options

☑ Keep depleted and added items from warehouses & airbases

Server options (works in multiplayer only)

☑ Server option to autosave with defined frequency the scenery

Autosave every (minutes) 1

Autosave option require, and automatically perform, desanitization of 'MissionScripting.lua'

☑ Enable detail debug mode. WARNING: can impact framerates & playability, activate only for reproduce a bug

Real time simulation enhancement options

☑ Enable troops transport & crate logistic with CTLD by Ciribob

☑ CTLD is set with real slingload setup: crates must be slingloaded

This option enable CTLD inside DSMC, with some added automation:
- it doesn't require setup: logistic, troops and helos are automatically recognized
- any FARP or warehouse is set as logistic unit
- APC or IFV vehicle are preloaded with infantry squads, disembarkable using CA (1st unit only)
Beware: only full-infantry groups will be extractable!

All the options are explained below. Each explanation is also provided with its equal variable name in *DSMC_Dedicated_Server_options.lua*

A. Enable/Disable the feature that keeps track of destroyed scenery objects. A scenery object is a 3D model which is not a static or a unit, like bridges, Buildings, factories, etc.
If this option is enabled, a destroyed bridge will stay destroyed in the saved mission. 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.

This is done by keeping track of every "death" event related to a scenery object, which is recognized using a getCategory() function. In the saved mission there will be a single trigger created (it will be a single one even in subsequent missions) that contains a very small "zone" for each object, in which anything is set to "dead". Easy trick. So, you can simply get to the zone list and remove those zones related to the object you want to be alive again. It's harder to explain than to try.

An important side note: 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 that are not late activation. 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.

Another important side note: 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 in Sochi.

Variable name: *DSMC_MapPersistence*

B. Enable/Disable the feature that creates a "dead" static object of the same shape that was the real unit where this unit died. Purpose? almost nothing, it's useful for scenery immersion. Also, those objects are created for downed helos & planes. That might be useful for a CSAR mission.

Variable name: *DSMC_StaticDeadUnits*

C. Enable/Disable the option to track automatically update 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, 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.

An important sidenote: due to the current status of dynamic weather and for the sake of mission planning, DSMC use static weather type.

Variable name: *DSMC_WeatherUpdate*

D. Enable/Disable the option to track spawned objects or not. If false, everything you added into the mission using scripts, won't be saved.

BEWARE: as better explained in the mission design guidelines section, you should be very aware that with this option enabled your persistency will do its best, but also 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!

Variable name: *DSMC_TrackSpawnedUnits*

E. Enable/Disable the option to automatically create client helicopter flights in FARP & Airports.

If this option is "on", DSMC will check if airport/FARP warehouse has flyable[2] helicopters available. If so, it will create slots for clients available on parking. This will follow some rules:

- All flights will be 2-ship, no wpt added, unless the airbase/FARP warehouse has only 1 or 3 airframes: in that case, a 1-ship or 3-ship flight will be created;
- If the warehouse has more than 4 airframes, only 2 flights (both 2-ship) will be created;
- If the airbase/farp is set as “unlimited”, then it will be created one 2-ship flight for each helicopter model[3].

As some of you could know, airports parking is very very difficult to manage. For the sake of “safety”, DSMC will check if all the possibile precondition to create slots are met. If not, it will skip that flight. So,

[2] Currently supports: Ka-50, Mi-8MTV2, UH-1H, all Gazelle variants
[3] For the Gazelle only the 342-L and 342-M variant will be created

expect that sometimes flights won't be created in certain airports. You should simply try and see…

Ok… so, what's the advantage of this thing? Easy: imagine a "progressive conquer" campaign: you may have a red farp with no airframe. If you conquer it, and then land helos there (let's say a couple of UH-1). In the saved mission DSMC will:

- Change FARP coalition from red to blue;
- See the 2 UH-1H in the warehouse;
- Create 1 flight of UH-1H, 2-ship, ready to be used or edited.

Also, the same thing apply if Ka-50 where already inside the warehouse: DSMC will create blue flights for those.

This feature is mostly intended for servers' owners who run DCS in a loop, restarting the saved mission each time.

Variable name: *DSMC_CreateClientSlot*

F. Enable/Disable the option to modify the starting hour and date of the next mission. Here you have 3 choices:

a. Do not change the start time/date (unclick the checkbox)
b. Change it, using the start time as the exact simulation date & time of the save command
c. Change it, using a randomized hour in the next day relative to the mission flown and saved[4]

Nothing else.

Variable name: *DSMC_UpdateStartTime*
Variable option: *DSMC_UpdateStartTime_mode* (must be 1 or 2)

G. Enable/Disable the option to track all warehouses items, when they aren't set as "unlimited". That was a very complex task to achieve,

[4] For lua guru: since DCS has some issue about mission starting daytime and ending nighttime with runway lights, I decided to add a filter: no mission will start after 16:00 and before 21:00. If you want to change-remove this limit, you should manually edit the TMUP.lua file inside DSMC directory… there are a couple of local variable that speaks from themselves: if you can't understand them, you probably should't modify the file.

and it sticks to **a couple of rules you must adhere**. Resource management in DCS is not accessible by scripting, therefore the only way to track that data is "running" is via parallel code outside, that tracks everything that is removed from a warehouse via takeoff event and also everything that is added via landings into the airbase/FARP.

This system is <u>not designed to work during the mission</u>: as said, I can't track warehouses items during the mission. Instead, <u>everything is tracked at the end</u>.

The main differences you must take into account are two:

- Real time supplies performed by DCS engine <u>will not be tracked</u>;
- All the changes into the items amount you will see during the DCS simulation, is not directly tracked into DSMC, but using takeoff & landing events.

In DCS, items are removed from the warehouse when you spawn and re-added when you de-spawn. DSMC, instead, will track takeoff & landings events.

Also, supplies will be performed accordingly to the removed items from any airbase & FARP, and it will be performed accordingly to the supply chain you set into the mission editor.

So, what changes? Simple: everything will be calculated and performed at mission end. You will find the result into the saved file.

Again, <u>DCS maths is performed real time</u>, during the simulation, therefore it's all about available items & petrol in the running mission. Else, <u>DSMC maths is done after the mission</u>, so it's about the starting items & petrol available for the next mission.

Another important point: 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 to use this properly:
As said this warehouse tracking system runs outside of DCS, using DCS gathered information to add or remove items & petrol. Ok, but how? It uses event handlers:

- When you takeoff from the Airport/FARP/Carrier/Helipad, items are removed.
- When you land, items are added.

If you set anything in the warehouse of the Airport/FARP/Carrier/Helipad as "unlimited", nothing will be added or removed.

It should work on anything that has a warehouse, from airports to FARP to oilrigs to ships to carriers etc etc.

Variable name: *DSMC_TrackWarehouses*

H. Enable/Disable the autosave procedure, we already talked about that earlier. If disabled, autosave won't do its work. If enabled but *missionscripting.lua* is sti ll unmodified (not desanitized), it won't take effect again.

Remember: this mod should "survive" at almost any update except for induced bugs, so you aren't required to do the mod dance disabling it before launching the update and then re-enabling it after update. Else, missionscripting.lua is ALWAYS reverted to its original "desanitized" state each DCS update: remember to desanitize it again.

Variable name: *DSMC_AutosaveProcess*

I. This slider decides how often you want DSMC to take updated data for the autosave procedure, in minutes. You can slide any number from 1 to 480 minutes.

Variable option: *DSMC_AutosaveProcess_min* (from 1 to 480)

J. 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.

Variable name: *DSMC_DebugMode*

K. Enable/Disable the CTLD by Ciribob clone code. Enabling this option you will get nothing more than the fantastic CTLD script working inside your mission, with some small additional features:

a. Weight feeling added for infantries! if you load a squad team in your Huey, you will feel the weight;

b. Reworked spawnable crates & group, now you can spawn FARP support units and also you will be able to spawn combined arms platoon depending on the country you choose: Georgian will have only Georgian vehicles, US only US ones and so on. You will be able to choose any vehicle type in service since 30 years before the mission date (in 2010 you won't have MTLB in Russian units, but instead you will have BTR-80 or BMP-3);

c. You can use any units in a group, you're no more forced to create flights with 1 unit only to comply with DCS limitation. Any pilot in a flight will have commands for each flight member;

d. Ground vehicle group will be able to load/unload troops & crates all units at once, not only the first unit of the group anymore. So you can transport 3 crates at once using a group with 3 trucks inside;

e. Trucks will be able to handle crates & troops by default: one crate or 20 soldiers each. APC will handle squad of troops (6-8 soldiers). IFV will handle fireteams (3-4 soldiers). All APC and IFV will have a soldier team pre-loaded by default;

f. Name coding for pilot units, extractable groups and logistic object is not necessary anymore. DSMC CTLD clone will automatically add to the corresponding tables (still there and editable if you wish) any helicopters, spawned or included in mission into pilots table, and any groups of infantry-only units into extractable groups and warehouses object (static-warehouse category) into logistic units;

g. FOB will be recognized in the mission editor, see advanced mission design guidelines for details;

h. JTAC units by default are Stinger and SA-18: those are still controllable by client but also valid troops for helos!;

i. Static units (SAMs) and FOB crates and JTACs can be unpacked everywhere, while instead all “movers” combined arms groups can be unpacked only nearby a warehouse object of “Ammunition deposit” or “Warehouse” type, that act like production sites.

Additional info: If you have an updated CTLD or a modified CTLD and you want to use that instead of the inbuilt version, no issue! But please remember to uncheck the option in the menu!

Variable name: *DSMC_automated_CTLD*. If you use it in servers, so with the variable option, you will also find some additional customization setup in the “advanced server options” (see mission design guidelines chapter).

L. Enable/Disable the "real slingload" option of CTLD, as it is. Basically if disabled you will be able to "load" crates virtually in your helicopter without the need to sling load them.

Variable option: *DSMC_CTLD_RealSlingload* (true/false)

WHAT ARE DSMC’S 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:

MOD RELATED

- You can't change anything in the options menu and then immediately start a mission. You always have to close and restart DCS
- You can't load the *DSMC-SavedMission_auto.miz* as a mission. You need at least to rename the file
- 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 and CTLD (the built in version). In particular, some couple of mods that open & close the file just before mission start (i.e. weather injection tools) have issues sometimes
- 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
- For dedicated server or server using the "--norender" or "--server" executable, it' required to desanitize *missionscripting.lua*

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

- Even with autosave feature enabled, DSMC does not provide code or apps to close & restart a DCS server: you must provide that. If you kill the server with task manger is going to be ok, but I might suggest the software "*RestartOnCrash*", tested and working fine with a gracetime setting of at least 120 seconds (better 300).

- Ok this one is gonna scare you… it scare me, for sure. About inbuilt CTLD… every 1st unit of a ground group, if It's an APC/IFV vehicle, is "pre-loaded" with a soldier squad. And that's normal. This team is going to stay inside the vehicle unless you enter using CA and use "unload/extract troops" command. And that's normal. BUT… i*f you* *have a custom kneeboard folder inside the miz file*… well.. all ground squad will immediately extract outside the vehicles at mission start creating a real mess. Why? You tell me: I don’t know. God don’t know. I mean… the “kneeboard” folder inside the miz file: it has nothing to do with real time CTLD commands… but it’s a confirmed bug and behaviour, and you simply have to remove the custom kneeboard to revert back to normal.

MISSION DESIGN GUIDELINE

DSMC is available to some testers since months, something after January or so. This fact allow 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[5] 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.

HELICOPTER POSITION UPDATE

If you have the *DSMC_SaveLastPlanePosition* server variable to "true" or the corresponding options "update position of landed helicopters" checkbox checked, the thing described at the letter "E" of previous chapters will happen.

To better design missions where you expect to move helicopters from place to place, you should plan the clients behaviour accordingly:

- Don't expect clients to have flight plans: they should plan accordingly to objectives you assign them on their own (and should have the time before takeoff to do that);
- Consider the options to use separate groups even in the same flights. That, a required things also for CTLD, will let them to keep their helicopter in separated place in the saved miz file if they land in different airbase.

SHIP POSITION UPDATE & CARRIER GROUP

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

[5] 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.

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.

FARP PLACING

Do not place FARPs or Helipads inside Airports perimeter, to be more specific keep a distance of at least 5 km (or 3 nm) from any other possible landing location.

Reason: Once landed, DSMC use a proximity function to identify the nearest allied things with parking. If you're too close, you might find your helicopter in the wrong place.

Also, if you have flights starting from a "blue" airbase and that becomes "red", those flights would still be there.

WAREHOUSE SUPPLY CHAIN

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.

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 a suppliers have fuel, and another the AIM-120, DSMC will take resources from both.

Since DSMC do not track infos about “first mission situation”, you surely want to carefully track the initial amount of an airbase resources, cause you can supply them only in two ways:

- By supplier logic, that anyway will re-integrate any expended items but never more than the “initial amount” at the mission start
- By landing aircraft with fuel/items, that can add resources to the “initial amount”, increasing it.

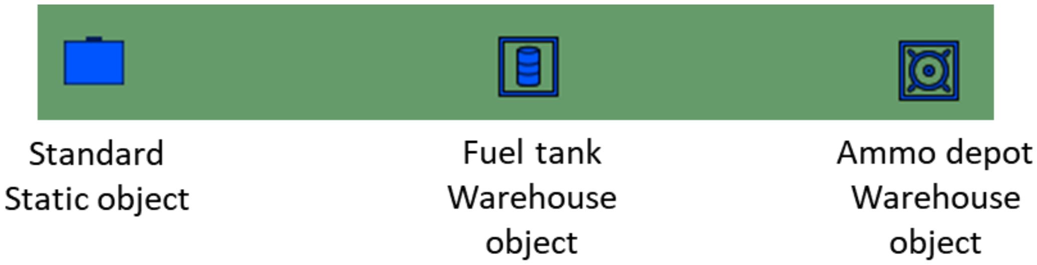
Since this is a delicate object, I’m going to explain in details how it’s intended to work.

Warehouse objects

DSMC don’t require any particular setup to define a warehouse object. It use 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:



Remember, DCS is a sandbox:

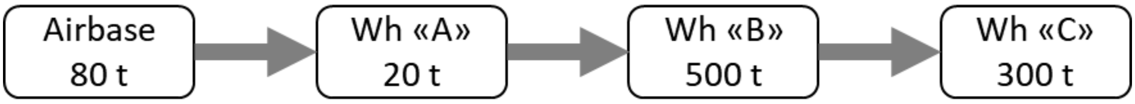
Every time you place a warehouse object on the scenery, it can be everything: a fuel tank (with that icon) can be an ammunition supplier, unless you manually void it. So, as per units placement, flight, etc… it’s up to you to set up that correctly.

Sadly, if you copy & paste a warehouse in DCS, you still have to manually set its “warehouse content” table.

<u>Basic vertical supply chain example</u>

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:



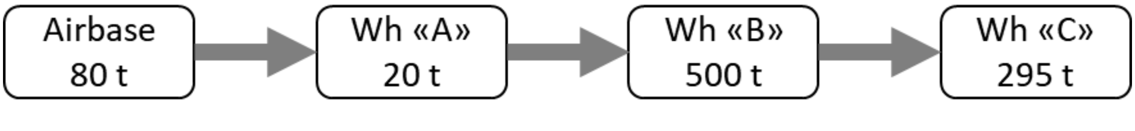
Airbase is the place where your flights consumed resources. The subsequent "Wh" can be other airbase as well as any warehouse object. Let's say that they are three fuel tank object where the only resources available is jet fuel (no munition, aircraft are ignored by design).

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:

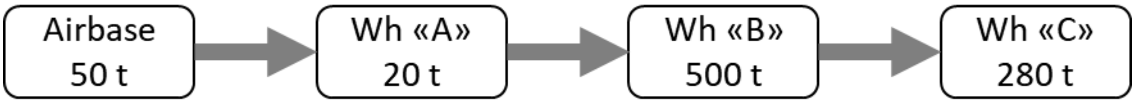


That is the key of "vertical" side of the chain. But… what happen if the airbase consumes 50 tons?

- 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:

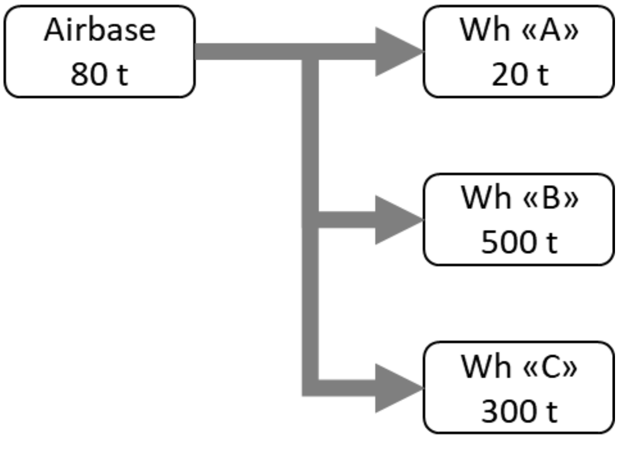


Note the “50 tons” at airbase, which has 80, while warehouse “C” took the 20 tons transferred by warehouse “A”.

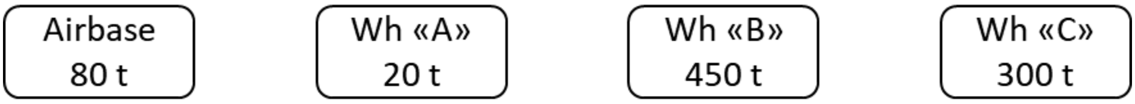
What does this mean? <u>That you can set a limit of the expendable fuel per mission</u>, which is given by the warehouse that has less resources. If you use more, this will be depleted from your airbase… and to re-charge that, you will now be forced to land something on it.... which is obviously much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

<u>Basic horizontal supply chain example</u>

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” cause it has more fuel than “A” and “C”. This will be the result:



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

Production sites, coalition deposit

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 deposit of a resource (ammo, fuel) and also create production sites!

Coalition deposit

Using fuel as example, to create a coalition deposit you can simply create one (or more, in proximity) “tank” warehouse object and place them where you want on the map, filling them with an 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 deposit 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 deposit, since it becomes empty.

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

Production sites

To create a petrol production site, you must simply place 2 warehouse object and set them like this:

- The first will be set as “unlimited fuel”. For the example, we call this “refinery”;
- The latter will be set as limited, with an “initial amount” equal to the daily production you want… let’s say 60 tons. For the example, we call it “refinery tank”

All your airbases, or deposit, should have “refinery tank” as supplier, so that every mission 60 tons will be added to the requester… since the enemy destroy the site.

Supply chain main example

A very basic supply chain setup is provided in an example mission that you can find under the “Examples” folder, DSMC mod folder.

CUSTOM FILE INSIDE MISSION

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, cause 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)

SPAWNING, IN ANY POSSIBLE WAY (MOOSE OR CODE)

This is a **very important point**. 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 object 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.

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 happen 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.

FARP & AIRBASE COALITION CHANGES

The helicopter landing position save option do not play well with coalition changes. If you plan to land in a conquered airbase or FARP and find your helicopter there in the saved mission, be sure that something else perform the coalition change before you land.

An important point about base/FARP coalition change & helicopter land position update (*DSMC_SaveLastPlanePosition* variable for servers). Imagine this situation:

If you land as a blue helo in a blue farp, then disconnect… so that helicopter group “slot” will be set in that farp in the saved file. But, before the save, some red units come nearby your farp and change its coalition!

In that case, you will see a blue helicopter slot over a red farp. And you can use that helo, only… you won’t be able to use the farp services.

To avoid erratic behaviour, I suggest to set all aircraft/helicopter with 0 kg of fuel on start: that way, you may spawn… but you won’t be able to load fuel and turn on the aircraft!

CTLD ADDITIONAL CONSIDERATION

CTLD modified version is entirely coded inside the file “TRPS_inj.lua”, DSMC directory. You can check inside and customize standard CTLD options as you like, but you should consider this chapter before touching. I can (almost) guarantee a “good behaviour” of the mod if you leave the file as it is.

Crates

All created crates will retain their original “content”, unless you change their weight using mission editor. You can complete an SAM installation or FOB in three consequent mission, if you’d like.

Unpack Crates

Here comes something new, pay attention. **You won’t be able to unpack any crates where you want**. You will be able to unpack SAM sites and support vehicles (FARP, rearming) where you like. And that’s the same as before.

But any other vehicles, like IFVs, Tanks, APCs can be unpacked only nearby a warehouse object (static object -> warehouse). Once unpacked, you can move them where you need. This will make your warehouse not only a resource production site (if you use warehouse tracking) but also a “military factory”, that you will try to protect for sure.

“simulated” sling load action

This option has been disabled by design. You can try to look into TRPS_inj.lua file to modify that, but I strongly suggest to avoid.

FARP support group

A “FARP Support group” has been added as spawnable for blue & red coalition. Will require the proper vehicles as DCS needs, one crates per vehicle. FARP support group won’t have anything special: it’s a ground vehicle group with the right composition to allow ATC, rearm & refuel when placed in proximity to an heliport.

“In transit” crates

It could happen that the mission will save or close while a crate is ongoing a transport, “inside” an APC or an helicopter (if you don’ use slingload mode). In that case, the crates is lost. DMCS won’t retain in transit crates.

FOB

Created FOB and its proximity beacon are retained in the saved mission.

JTAC

Due to design reasons JTAC is performed by “manpad” units: Stinger for blue coalition and SA-18 for red coalition. The main reason is that currently JTAC are unarmed unit that is required for FARP support units, creating bugs.

Anyway, that is a technical feature: “manpad” units can be used by player/client just as vehicles, but they can also be transported by helicopters.

BEWARE: any “single soldier manpad” group will be inspected at mission start, set as JTAC unit, and set ROE to “**return fire**”. Therefore, if you want normal manpads to behave correctly, you simply need a group of at least 2 units. You can add a single soldier, or another manpad, or anything you want. But don’t leave the manpad unit alone!

Logistic zones/object

As said internal CTLD version automatically recognize any previously built FOB & any warehouse object as a logistic object, allowing you to spawn crates, load squads, etc etc.

Also, FOB recognition is not done by adding properties to the object. Instead, it's done by checking static object in the saved file: each time you place a beacon TTS unit (Fortification class) within 150 meters from an outpost static object (not road outpost, not a fortification), those will be recognized as "FOB" and full ctld functionality will be added to them, including beacon.

PS: that beacon won't retain the same frequency as the previous mission: it will change each time the mission is restarted.

ADVANCED SERVER CUSTOMIZATION

Inside the DSMC_Dedicated_Server_options.lua file there are also some "advanced server options" that should be used **only** if you clearly know what you're doing. In particular, you will find:

DSMC_cleanDamagedSAMsites: This option will check for "damaged" sam sites, where damaged means any group that enlist a "launcher" unit without having a related "radar" unit. Those group will disappear after some seconds from mission start and won't be in the saved file. This option is very much "beta", with a couple of "it removed single SAM also!" reports, so I suggest to leave it off.

DSMC_CTLD_AllowCrates: This option will enable/disable all the crates transport & building features in inbuilt CTLD. If it's false, you will have only troops transportation.

DSMC_CTLD_AllowPlatoon: This option will enable/disable a new ground unit listing for CTLD crates logic. Instead of having a fixed ground vehicle list that you would be able to create using crates, the list will vary from country to country, regardless of coalition, and depending on everything available in that country units as per DCS database. That won’t work with complex SAMs, only for armoured vehicles!

DSMC_CTLD_UseYearFilter: Since for many countries there are A LOT of possible units, but we don’t see why you would build a T-55 in 2015, if you enable this filter you will be able to built units that have been created in the last 40 years before the mission start date. If your mission start in 1970s, you won’t see T-90 in the CTLD platoons crate list, for example.

*DSMC_CTLD_UnitNumLim*its: this option is done to prevent a server to become unbalanced in ground units availability. If It’s set as “true” both coalition won’t be able to have more than “n” units category available in the scenery: if that number is more than the limit, you won’t be able to use crates to create more vehicle of the same category (you won’t be able to unpack the crates, but you can spawn and move these crates).

DSMC_CTLD_Limit_(unit category): This is the number I spoke in the previous option. For example, if you set *DSMC_CTLD_Limit_Tanks = 50* then both coalition will be able to have 50 tanks, no more. If the starting scenery has 200 tank units in the blue coalition, then you will be able to create them using crates only when the alive tanks count goes below 50.

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;
- Neon;
- Panthir;
- Paranoid;

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 (or simply modifying them, like for inbuilt CTLD version) as I needed.

So, many many thanks to:

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