cf/x

Dynamic Mission Library
for DCS

DML

QUICK REFERENCE

MISSION EDITOR MODULE INTEGRATION

Table Of Contents

1 ME INTEGRATION (common abilities)

1.1 Core Abilities (all DML Zones)

All Modules automatically support some attributes when they are present. They can be added to any Trigger Zone and their functionality is always present for all modules

1.1.1 Dependencies

All DML modules require dcsCommon and cfxZones.

1.1.2 ME Integration

Name	Description
linkedUnit	Moves the zone's center with the unit whose name *exactly* matches the value of this attribute. That unit must exist at the beginning of the mission, or the linked zone remains at its ME location until the unit exists and becomes linked (at which point it moves to the unit's location). If the *linkedUnit* ceases to exist after the zone has followed it, the zone remains at the last location it moved to until a new unit with that exact name appears again. This is often the case with player-controlled planes, so expect this to happen and design your mission accordingly. If neither *useOffset* nor *useHeading* attributes (see below) are set to true, the zone always centers on *linkedUnit's* location. **Note:** be advised that all player and client units do not exist at the beginning of a mission and spawn only when a player occupies that slot. This means that zones linked to player or client unit will only move to those locations after the player has entered the game. **Note:** The 'linkedUnit' attribute is one of two methods to link a zone to a unit. The other method is ME's dedicated 'LINK UNIT' pop-up (see description, above). Use only one of the two methods to link a zone to a unit, else DML will give you an error. The advantage of using a *linkedUnit* attribute over ME's pop-up method is that the *linkedUnit* attribute can be used with dynamically spawned units (i.e. units that aren't placed with ME). Its main disadvantage is that it can be prone to spelling mistakes or when you change the name of the linked unit in ME. Defaults to <not linked to any unit>
useOffset	Must be set to "yes" or "true" to have this effect, ignored otherwise. Only has an effect if the zone is linked. Keeps the offset between the linked unit and zone constant. Note that the zone's center remains the same in relation to the unit's center. If the unit turns, the offset does not change with the unit's heading.

Name	Description
	Requires either *linkedUnit* or LINK UNIT be set Not compatible with *useHeading* (below) – use either Defaults to <not set>
useHeading	When set to true, the zone moves and turns in synch with and relative to the *linkedUnit*. **Requires either *linkedUnit* on LINK UNIT be set** Not compatible with *useOffset* (above) – use either Defaults to <not set>
owner	The coalition that owns this zone. Used with some zone enhancements. Do not set this attribute unless you know what you are doing, or a module's documentation requests you to do so. Defaults to neutral

1.2 The "Yeller Model" of module cooperation/integration

DML models love to talk to each other. But how do modules communicate with each other? There is talk of "flags" and "signals" and whatnot, but for something that is supposed to be simple to use, understanding how DML modules internally work with each other, should *not* be required. What we need is a metaphor, a figure of speech, an analogy that reflects the spirit, even if it may technically be somewhat incorrect:

(Please also see the demo mission "demo – The Yeller Model.miz")

1.2.1 Of Old Yellers and El-Aurtians

Instead of getting down into the nitty gritty of techspeak, let's try a more accessible approach. It holds up well and explains how this stuff works without needlessly getting into details. So, no "Flags" or other weird stuff. Let's keep it simple.

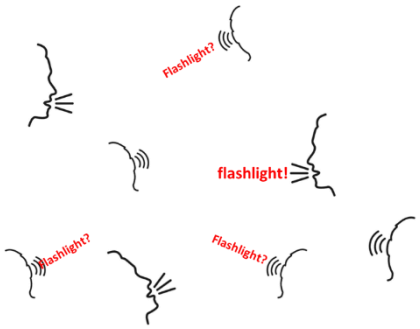
So, how does, for example, a *unitZone* tell a *cloner* to produce new enemies? We can boil it all down to astoundingly simple things: "Yellers" and "Listeners".

When told to, "Yellers" shout out commands to anyone who would listen, and "Listeners" continuously listen to hear a specific command. If they hear that command, they get into action. They ignore everything else.

And here are the important points: Yellers don't care who listens to them, and Listeners don't care who shouts their command. Each do their own stuff, and they neither care nor know about the others. If Listeners hear a command other than the one that they are waiting for, they ignore it.

I call this "**DML's Yeller Model of co-operation**", or "**Yeller Model**" for short. Let's take a closer look:

1.2.2 A room full of people…



Imagine a dark room filled with Yellers and Listeners, all minding their own business. Now imagine that there are three Listeners in there who wait for the command "flashlight", and when they hear that command, they are instructed to turn on their flashlight for 20 seconds.

Now you also place a Yeller into the room and tell it to yell "flashlight" every 90 seconds.

What will happen? If everyone does as instructed, the Yeller shouts "flashlight" every 90 seconds. All Listeners hear that command, and the three of them who are waiting for "flashlight" shine their flashlight for 20 seconds.

So, every 90 seconds, the room is lit by three flashlights for 20 seconds. All the other Listeners in the room ignore the Yeller's command because it's not what they are instructed to wait for.

And that's basically how DML works. Some modules yell, others listen (some do both). A module with an "Output" (an attribute that ends on an exclamation point "!") can yell commands. For example, the *pulseFlag* module has an output "*pulse!*". The value that you put into the field next to it is the *command* that this pulser will yell - in our example it is the command "gogogo" and it is to be yelled every 90 seconds (*time* is set to 90)

gogogo

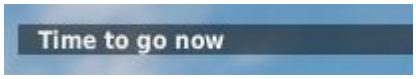
Name	Value
pulse!	gogogo
time	90

Name	Value
messenger?	gogogo
message	Time to go now

A module with an "Input" (an attribute that ends on a question mark "?") listens for a command. If this input hears someone yelling that command, the module reacts.

For example, a *messenger* module listens on its "*messenger?*" input for the command "gogogo". It is set up to print the message "Time to go now" onto the screen when it hears the command "gogogo".

If you run that mission, the output "Time to go now" appears on the screen every 90 seconds.



In a nutshell: if one module wants to tell another module to do something, they use a common command or keyword. When it's time, the yeller blindly shouts its command into the open. So, yeller modules like *pulseZones*, *unitZones*, *radioMenus,* they all yell their commands, not caring who might listen.

The Listeners hear “their” commands only, and react by doing what they do: cloners clone, messengers put up a message, and smoke zones start or stop smoking.

So, if you set up an *LZ* module to yell “playerLanded” from its *landed!* output when a player lands inside it, and a *smokeZone* listens for the command “playerLanded” on its *startSmoke?* input, that smoke zone starts smoking when a player lands inside the LZ zone.

1.2.3 Of Inputs! and Outputs!

And that is how the “Yeller Model” works: “Listener” modules wait until they hear the correct command on their input and then do something. “Yeller” modules look for a certain situation to arise (for example 30 seconds have passed), and then yell a command into the void, not caring if anyone is listening.

And when you put these two groups together, you create a bigger whole. Crucial for Yellers and Listeners to work together is that they use the same command, and that there is no crosstalk (i.e. other modules that use the *same* command for different purposes. Logic inside DML prevents a module can’t hear a command because another module is shouting a different command at the same time – in DML, all commands are always perfectly received by all modules). Yellers shout their command through their *output!* into the blue, and don’t know nor care who receives their command; Listeners listen for commands at their *input?* and they respond to *any* Yeller’s command if it matches, no matter who yelled it. This makes it easy to build complex automatons that work across the entire map – and it makes it equally easy to goof if you by accident make a Listener wait for the wrong command or have a Yeller scream an already-used (existing) command across the mission, triggering unwanted action.

1.2.4 What’s in a name? Commands

There’s one thing that I haven’t mentioned yet, and that perhaps has you scratching your head: What are these ‘commands’ that I speak of, and where do I find them? You don’t. You

make them up. Commands are ‘code phrases’ that you, the mission designer, assign to the inputs and outputs, and I recommend that you aren’t too creative with them. Use commands that are descriptive (like “SAM North destroyed”), and that help you to determine what they are used for. Moreover, make sure that you always spell them *exactly* the same. DML is case sensitive, and if you, for example, get its capitalization wrong by just a hair, a Listener won’t react to it. So be very, very careful how you name your commands, and mind your spelling and capitalization. You are (mostly) free otherwise, and can name your commands like you see fit. I only *recommend* that you do not use blanks in a command’s name, and use “CamelCase” instead: use “SAMNorthDestroyed” instead of "SAM North destroyed”. Oh, and **never use a comma** “,” in a command’s name.

Always remember that commands have to be matched pairs to work: there has to be at least one Yeller and at least one Listener that uses the same command. You *can* have three Listeners for the same command, no problem. And you can have multiple Yellers shouting the same command. But if there is no Yeller that shouts a particular command across the mission, your waiting Listeners do have a problem. Conversely, if your modules yell a command, and nobody is listening to that command, it won’t have any effect.

1.2.5 Hiding the Details: it’s a good thing!

The Yeller Model is a simplification of how DML modules work together; it’s a nice and easy metaphor. The good part is that it works. The great part is that you probably won't need to know more - DML is set up to simply work this way.

Now, like all metaphors, they only work to a certain point before they break. Most mission creators can rest easy, knowing that they'll never reach that point. Occasionally, though, you may want to accomplish something trickier, something that requires that little bit of extra finesse. DML has you covered, but in return it requires that you know more about the underpinnings of the "Yeller Model": Flags and the incredible additional lifting power that DML provides should you want to get that really slick finish.

In this context, simply know that the Yeller Model’s "commands" that modules yell and listen for are abstractions of a DCS concept called ‘Flags’; that they can do much, *much* more. Simply nod wisely when someone mentions “flags”, knowing that if someone uses that term with DCS, they in principle mean the commands that we talked about. Under the hood, they can be used for a lot more than what we do with the Yeller Model. Only few Missions require that kind of deeper knowledge and flexibility. And, if you *really* want to, DML provides you with a rich environment that manages DCS flags for you and can imbue them with additional properties to accomplish lesser magic.

As a result, if your mission requires some sorcery, you will need to resort and know flags. Until then, you should use the "Yeller model" and create great missions. Most of the missions that I have created work *entirely* with the Yeller Model. Yes, that includes “Expansion”.

1.2.6 The Yeller Model in other parts of this Document

The “Yeller Model” is an easy simplification that I introduced to help people understand and more easily enjoy working with DML. The module description and all the demos are written with the (more technical) Flags model in mind. Translating that into the Yeller Model isn’t difficult:

- Inputs and Outputs are the same in both models
- “Banging on a Flag”, “sending a signal” or “receiving a signal”, triggering all mean the same: yelling/hearing a command
- “Trigger” means sending/receiving a command
- DML Method (advanced stuff that wrap commands, ignore)
- Flags, Watchflags are synonymous with inputs/outputs (a *Watchflag* is an input)

Some concepts in the “Flags Model” don’t translate to the Yeller Model and can be safely ignored. If the above translations don’t help you understand what an attribute is for, you can probably ignore it.

1.3 The “Flag Model”: Understanding DML Flag use

The Yeller Model is a great analogy that describes how modules interact with each other and makes a lot of technical details transparent. Underneath the hood, DML implements module communication with something that is called “Flags” in DCS parlance that are used to send signals from “outputs” to “inputs”, and that can be configured for complex behavior. Knowing these details is not required to build great missions with DML. In fact, it is often advantageous to not bother with the details, as mission designers can get lost and start to obsess about some of the many details. My recommendation is that you stick to the “Yeller Model” until your mission requirements outpace the simplifications of the “Yeller Model”.

The DML “Flag Model” uses “Inputs” and “Outputs” for modules to communicate:

- It *sends* signals (the **output (“!”) attribute**, usually designated by the exclamation point "!" at the end of the attribute's name). When sending a signal, the module changes the value of the flags that are listed after the output attribute. The way that it changes the values is described by the ‘output method’ attribute, which is usually “inc” or “increment the current value by one”.
  Mission designers can change the way a module changes values on a zone-individual basis. **A module sends signals (sets output flags) whenever it sees fit**, usually as a response to something happening in the mission (a group enters a zone, an object is destroyed etc.)

- It *receives* signals: modules periodically look at the flag that is listed for their **input (“?”)** flag attribute(s) (usually an attribute with question mark "?" in their name) and compare that value to what they read before. If the value is the same as before, that input is ignored. If the value did change, they then decide if what they read fulfils the trigger condition (also called ‘input method’). If it does, the module triggers its appropriate action. The most common trigger condition is 'change': the input triggers when the flag's current value changes. In DML, trigger conditions are called ‘methods’, and you can tell a module which conditions it should trigger on

  Note:
  Modules periodically look at their input flags, usually once a second (you can set this interval for the module with the “ups” attribute in the module’s config zone). This means that modules do not instantly detect a signal/change, *only the next time they look at their input flags*. This also means that they can miss a signal that was sent if the input flag changes too quickly (a rare occasion).

1.4 Bang! Output Methods (output, sending signals)

DML understands the following methods, each identified by a keyword or expression:

- ‘on’ **[set to 1]**
  Sets the flag’s value to one, no matter what it was before. Same as using the number 1 (one)

- ‘off’ **[set to zero]**
  Sets the flag’s value to 0 (zero), no matter what the value was before. Same as using the number 0 (zero)

- ‘inc’ **[increase by 1]**
  Increases the flag’s value by 1 (one). Same as ‘+1’. If, for example, the flag’s value was previously 10, that is increased to 11. **This is the most common bang! method and is the default for most DML module outputs**; it co-operates best with input flags that are set to trigger on “change” – which is also the default in DML

- ‘dec’ **[decrease by 1]**
  Decreases the flag’s value by 1 (one). Same as ‘-1’. If, for example, the flag’s value was previously 10, this number is decreased to 9.

- ‘flip’ **[alternate between 0 and 1]**
  This is a very effective methods to trigger on flag change. It flips the flag’s value between 0 (zero) and “not 0”: If the flag’s value was anything except zero, the new value is zero. If the flag’s value was zero, the new value is 1 (one). This way you can flip-flop flags, turning them on and off repeatedly. Note that this method can be error prone if one or more modules flip a flag’s value multiple times before a module’s input can detect a change: if two modules “flip” a flag in rapid succession before an input checks the flag’s value, it appears to be the same as before even though it was changed: the second flip returned the flag to its initial value, masking the change. That is why the “inc” method is to be preferred over ‘flip’ for general flag bang!ing.

- #(number or flagName) **[set to absolute value or that of another flag]**
  Sets the flag to the fixed value (when giving a number) or copies the value from another flag, no parentheses.

  Examples:
  - #33 sets the flag to the number 33,
  - #-6 sets the flag to the number -6 (negative six),
  - #kills sets the flag to the value that the flag named ‘kills’ currently holds,
  - #”122” (note the quotes) sets the flag to the value that the flag named ‘122’ (a legal, old DCS flag name) currently holds

- +(number or flagName) **[add amount or another flag’s value**]
  Adds the number give (no parentheses!) or the current value of the flag flagName to the flag.

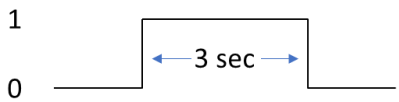
  Examples:
  - “+3” adds 3 to the current flag value, while
  - “+killScore” adds the current value of flag “killScore” to the flag, and

- `+"22"` adds the current value of DCS flag named 22 to the current value

- `-`(number or flagName) **[subtract amount or another flag's value]**
  Subtracts the number given (no parentheses!) the or current value of the flag flagName to the flag.

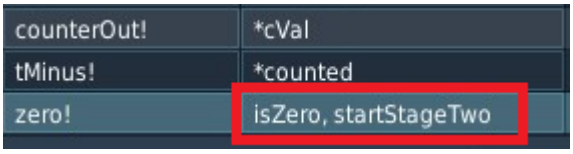
  Examples:
  - `-3` subtracts 3 from the current flag value, while
  - `-penalty` subtracts the current value of flag "penalty" from, and
  - `-"22"` subtracts the current value of numbered flag "22" from the current value

- `'pulse'` or `'pulse, <number>'` **[set to 1 and reset automatically]**
  "pulses" the flag by setting its value to 1 (one) for some time, and then re-setting it to 0 (zero) some time later. If you do not specify any time, the flag is reset after three (3) seconds. You can supply your own pulse time by adding a comma and a number,



  Example:
  - `"pulse, 4"` will keep the pulse up for four (4) seconds before dropping back to zero.

1.5 Multiple Output Flags



DML can bang! multiple flags at the same time. Unless otherwise specified, all outputs (those attributes that end on an exclamation point "!") support this ability. To bang! multiple flags, simply list them as the attribute's value and separate them by comma; leading/trailing blanks are ignored.

Please note the following:

- All flags are banged! with the same method.
- All flags are set at the same time, meaning that there is no guaranteed order in which they are changed.
- Should you list the same flag multiple times, it will get set multiple times; this means that the value changes internally multiple times; that does does not mean that inputs who watch that flag can detect multiple changes; from an external point of view the value of that flag changes at maximum once between inspections.

1.6 Input Methods (input, receiving signals)

DML understands the following conditions to look for in a flag, each defined by a keyword. Watchflags are inspected regularly and can only trigger after a flag's value changes; they trigger when the following conditions are met:

- `'change'` or `'#'`
  trigger whenever the watched flag's value changes. No other conditions need be

fulfilled. **This is the default**

- `'off'` or `'0'` or `'no'` or `'false'`
  triggers when the watched flag's value changes to zero

- `'on'` or `'1'` or `'yes'` or `'true'`
  triggers when the watched flag's value changes from zero to non-zero
  (**Warning**: with this method, DML will *not* detect a transition between two non-zero numbers e.g., 3→4, it only triggers on a change from ZERO to a non-zero value. To trigger again, the flag must first return to a value of zero)

- `'inc'` or `'+1'`
  triggers when the watched flag's current value is greater than the previous value

- `'dec'` or `'-1'`
  triggers when the watched flag's value is less than the watched flag's previous value

- `'lohi'`
  triggers when the watched flag's previous value was zero (0) or less and the new value is greater than zero. Often used with pulses.

- `'hilo'`
  triggers when the watched flag's previous value was greater than zero (0) and the new value is zero or less. Often used to detect a countdown reaching zero.

- `'>(number)'` or `'>(name)'`
  triggers when the watched flag's value changes, and the value is larger than the number given or flag identified by name

  Examples:
  - `>4` triggers when the watched flag's value is larger than the number 4
  - `>*landings` triggers when the watched flag's value is larger than the value of local flag 'landings'

- `'=(number)'` or `'=(name)'`
  triggers when the watched flag's value changes, and the value is equal to the number given or flag identified by name

  Examples:
  - `=4` triggers when the watched flag's value is equal to the number 4
  - `=*landings` triggers when the watched flag's value is equal to the value of local flag 'landings'

- `'<(number)'` or `'<(name)'`
  triggers when the watched flag's value changes, and the value is less than the number given or flag identified by name

  Examples:
  - `<4` triggers when the watched flag's value is less than the number 4

- `<*landings` triggers when the watched flag's value is less than the value of local flag 'landings'

- `'#(number)'` or `'#(name)'`
  triggers when the watched flag's value changes, and the value is not equal to the number given or flag identified by name

  Examples:
  - `#4` triggers when the watched flag's value is not equal to the number 4
  - `#*landings` triggers when the watched flag's value is not equal to the value of local flag 'landings'

Quoting Numbered Flags
Early versions of DCS used flag names that entirely consisted of number. For example, "22" was (and still is) a legal flag name. This can create confusion when using some trigger methods: DML can't tell the difference between a number and a flag whose name happens to be a number.

To allow DML to distinguish between a number and **flags whose name happens to be a number**, such a flag's name **must be put into double quotes** '"' and '"' to be interpreted as a flag number. Hence, if you want the input to trigger only if the connected flag was equal to flag named 22, that condition would be

`="22"`

Note the two quotes. DML then (and only then) recognizes "22" as meaning **the flag named 22** rather than the number 22.

1.7 DML Flag Naming Rules

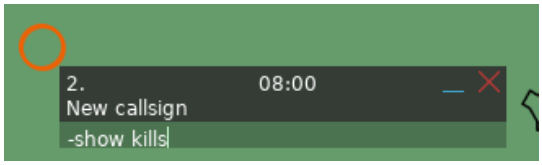
	DML	**DML Zone-Local**
Format	• Alphanumeric • must not contain comma ',', • must not start with asterisk '*' • must not start with double quote '"'. *should* not start with a digit ('0'...'9')	**Starts with asterisk** '*', alphanumeric, must not contain comma ','
Examples	• A12 • With blank • F***d up Yup "quotes" too	• *1 • *A12 • *fireCloner *ok multi **
Scope / Visibility	DML modules, Entire DCS (newer versions)	Only DML modules attached to the same zone
Invisible to	n/a	Everyone outside Zone

2 DEBUGGER

2.1 Summary

"The Debugger" is an interactive (in-mission) debugger that sports a 'console' via the Add Map Mark Label' mission feature to give commands. Commands all start with a hyphen.

"Mark Label"



and will elicit a response when accepted

2.2 Dependencies

The debugger requires dcsCommon and cfxZones

2.3 ME Integration

Name	Description
debug?	List the flag names that the debugger is to observe. All flags listed here are accessible from the debugger under the observer with the same name as the trigger zone **MANDATORY**
triggerMethod debugTriggerMethod inputMethod sayWhen	Trigger condition for the flags (the observer's "condition" that triggers a report for the flag) Defaults to 'change'
method outputMethod debugMethod	DML Method for the debugger's output flags. Rarely used. Defaults to "inc"
notify!	DML flag to bang! when a flag listed in debug? triggers
debugMsg	Message to output when a flag listed in debug? triggers. Supports wildcards, including <f> for the flag name that triggered, and <z> for the zone name. **Note that this allows you to provide individual message formatting per observer**, a feature that is not available for the interactive debugger. Defaults to "`---debug: <t> -- Flag <f> changed from <p> to <c> [<z>]`" which results in a message similar to `---debug: 08:00:12 -- Flag t1 changed from 2 to 4 [many flags]`

2.4 Demos

- Bug Hunt

3 Airfield

3.1 Summary

This module provides signals from airfields/FARPS when they are captured, and can provide control over airfield/FARP ownership to assign an airfield to a coalition (RED/BLUE)

3.2 Dependencies

Airfield requires the modules dcsCommon and cfxZones.

If you are using persistence and cloners or spawners that take their ownership from airfield zones, airfield should load before cloneZones and spawn zones.

3.3 ME Integration

Name	Description
airfield	Tells DML that this trigger zone should associate with the closest airfield (or FARP, see config settings) **The owner of this zone is always the same as the owner of the airfield/FARP that airfield associates with.** The value of this attribute is ignored. **MANDATORY**
farps	When set to true, airfield zones also associate with FARPs. When set to false, FARPs are ignored. When enabled, care must be taken when placing FARPS close to airfields: an airfield zone associates itself with the *closest airfield or FARP*, and the in-game location of an airfield may not be where you assume it is. Hint: enable verbosity and see which airfield zone attaches to which object. Defaults to false (associate only with airfields, disregard FARPs)
fixed	Defines which side holds the associated airfield/FARP **at mission start.** The airfield/FARP is made **unconquerable**, and remains in that coalition's possession until the zone receives a signal on one of the following inputs: • `makeRed?` – ownership is turned over to RED • `makeBlue?` – ownership is turned over to BLUE • `makeNeutral?` – ownership is turned over to NEUTRAL • `autoCap?` – makes the airfield/FARP capturable by mission ground capture rules. Valid values are 0, 1, 2, red, blue, neutral Defaults to <none>, airfield can be captured normally, and initial ownership is as set up by mission editor.
method	Output method for all outputs. Defaults to 'inc'
triggerMethod	DML input method for all inputs

Name	Description
	Defaults to 'change'
red!	Flags that should be sent a signal when the associated airfield/FARP is captured by RED faction Defaults to <none>
blue!	Flags that should be sent a signal when the associated airfield/FARP is captured by BLUE faction Defaults to <none>
makeRed?	When a signal is received on the flag that connects to this input, the zone takes control of airfield/FARP ownership (no longer mission ground capture rules), and forces ownership to RED faction. If the airfield/FARP was previously owned by a different faction, a signal is sent to red! Output. To have the airfield return to being able to be captured by other factions, send a signal to the autoCap? input. Defaults to <none>
makeBlue?	When a signal is received on the flag that connects to this input, the zone takes control of airfield/FARP ownership (no longer mission ground capture rules), and forces ownership to BLUE faction. If the airfield/FARP was previously owned by a different faction, a signal is sent to blue! Output. To have the airfield return to being able to be captured by other factions, send a signal to the autoCap? input. Defaults to <none>
makeNeutral?	When a signal is received on the flag that connects to this input, the zone takes control of airfield/FARP ownership (no longer mission ground capture rules), and forces ownership to NEUTRAL faction. To have the airfield return to being able to be captured by other factions, send a signal to the autoCap? input. Defaults to <none>
autoCap?	When a signal is received on the that connects to this input, the zone relinquishes ownership control over the associated airfield/FARP and ownership is determined by the mission ground capture rules Defaults to <none>
directControl	When set to true, at mission start the zone assumes control over the associated airfield/FARP's ownership: it can no longer be captured by mission ground capture rules. Use the `autoCap?` input to relinquish control back to the mission, or `makeRed?`, `makeNetral?` and `makeBlue?` inputs to explicitly set ownership. Defaults to false (at mission start, the associated airfield's ownership is determined by mission ground capture rules)
ownedBy#	Flags connected to this output are set to the value of the currently owning faction of the associated airfield/FARP: 0 (neutral), 1 (red) or blue (2) Defaults to <none>
redFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the airfield's capture zone when owned by RED coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors)

Name	Description
	Defaults to config zone's "redFill" value
redLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the airfield capture zone's outline when owned by RED coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to config zone's 'redLine' value
blueFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the airfield's capture zone when owned by BLUE coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to config zone's "blueFill" value
blueLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the airfield capture zone's outline when owned by BLUE coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to config zone's 'blueLine' value
neutralFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the airfield's capture zone when owned by NEUTRAL coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to config zone's "neutralFill" value
neutralLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the airfield capture zone's outline when owned by NEUTRAL coalition. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to config zone's 'neutralLine' value
show	If set to true, the airfield's 2km capture zone is shown on the F10 map in the colors described above, by owning coalition. Defaults to false (capture zone is not shown)

3.4 Demos

- Airfield mine
- Send in the Clones

4 Airtank

4.1 Summary

This module allows player-controlled aircraft to extinguish fires controlled by the inferno module

4.2 Dependencies

Airtank requires dcsCommon and cfxZones

Only makes sense with inferno.

4.3 ME Integration

Add airtank zones to enable airtank-enabled aircraft to refill quickly inside refill zones

Name	Description
airtank	Marks this zone as an airtank refill zone The value of this attribute is ignored **Mandatory**
capacity	Maximum amount of Flame retardant that can be stored in this refill zone. Defaults to 99999999 (essentially infinite)
amount	Amount of flame retardant that is stored in this refill station at the start of the mission. Once this amount is reduced to zero, aircraft can no longer be refilled from this refill station. Defaults to the amount of capacity

4.4 Demos

5 Artillery Zones

5.1 Summary

Simulates an artillery barrage inside the zone.

5.2 Dependencies

dcsCommon, cfxZones

5.3 ME Integration

Name	Description
artilleryTarget	Marks this zone as an artillery zone. Value is ignored **MANDATORY**
coalition	Used with Artillery UI – the coalition that can give a fire command (the explosions are completely coalition agnostic – they kill anyone). When the artillery zone is marked on the map, only this side will see it. Defaults to 0. Supports “red” and “blue” as values
spotRange	Used with Artillery UI – the maximum range at which an FO can give a fire command. Measured from center of zone. Defaults to 3000 meters
shellStrength	Average power of **each** exploding shell. Defaults to 500. 3000 is enough to level big buildings, so be conservative.
shellNum	Number of shells (salvo) per fire cycle. Defaults to 17 shells per cycle
transitionTime	The time (in seconds) the shells take on average to reach the target zone. Note that not all shells arrive at once but are usually spread over a couple of seconds. Defaults to 20
addMark	Add the artillery target zone to the F10 map of coalition (see above). Defaults to **true**.
shellVariance	Difference in shell’s explosion power, in percent. Defaults to 0.2 (20%)
f? in? artillery?	**DML Watchflag**. When triggered, the artillery bombardment starts. Defaults to <none> You can use any synonym, but only one per zone
triggerMethod artyTriggerMethod	Defines the trigger condition for the DML Watchflag. Defaults to “change”
cooldown	Used with Artillery UI: Number of seconds before the next fire cycle can be initiated. Is ignored when initiating fire via ME flags. Defaults to 120 (= 2 Minutes)
baseAccuracy	The radius (in meters) around the center of the zone in which the projectiles will land. Defaults to the ME zone’s radius (meaning all projectiles will land inside the zone if this attribute is missing and fire cycle is invoked via trigger flag)
silent	Used with Artillery UI: if true, suppresses communication responses from artillery

5.4 Demos

- Artillery with UI
- Artillery zones triggered
- Pulsing Fun

6 Artillery UI

6.1 Summary

Provides forward observation features for helicopters, along with a UI to trigger artillery zones.

6.2 Dependencies

Tcb

6.3 ME Integration

Tbc

6.4 Demos

- Artillery with UI

7 ASW (asw, aswZones, aswGUI, aswSubs)

7.1 Summary

ASW provides Anti-Submarine Warfare mechanics to your mission, allowing players to stock up on ASW munitions, and hunt for submerged aircraft by dropping buoys and torpedoes.

7.2 Dependencies

- **asw** requires dcsCommon, cfxZones.
- **aswZones** requires dcsCommon, cfxZones and **asw**
- **aswGUI** requires dcsCommon, cfxZones, **asw** and **aswZones**
- **aswSubs** requires dcsCommon, cfxZones

7.3 Configuration

Most ASW* features are set up via configuration zones:

7.3.1 ASW (main)

Name	Description
verbose	A value of “true” turns on debugging for the entire module. Default is “false”
buoyLife	Duration (in seconds) how long a buoy is active. Defaults to 1800 (=30 minutes)
fixLife	Number of seconds that a sub fix is reliable and can be used by nearby torpedoes to get a better homing chance. Defaults to 180 (=3 minutes)
detectionRange	Sensor range (in meters) for a buoy. Reliability deteriorates with distance, and beyond this range, no subs are detected. Only enemy subs are detected when in range Does not detect vessels on the surface, even if those vessels are of the submarine class Defaults to 12000 (12km)
sureDetect	“point blank’ range for buoys. If a sub is closer than this range they are always detected. Defaults to 1000
detectionDepth	Maximum depth (in meters) in which submarines are detected. Submarines that are deeper than that are invisible and will not be detected. Defaults to 500
fixSound	Sound effect to play when a new fix on a submarine is made Defaults to "submarine ping.ogg" (file is part of DML’s library)
sonarSound	Sound effect to play when a buoy finds a contact. Defaults to "beacon beep-beep.ogg" (file is part of DML’s library)
redKill!	Output flag to bang! when a red submarine is killed by ASW means. Defaults to <none>
blueKill!	Output flag to bang! when a blue submarine is killed by ASW means. Defaults to <none>
method	Method for output flags

Name	Description
smokeColor	Color that marks a buoy. Can be a number (0 to 4) or “red”, “green”, “blue”, “white”, “orange.” Defaults to “red”
killScore	Only relevant when PayerScore is active in mission Score for killing a submarine with ASW munitions Defaults to 0 (no score)
killFeat	Only relevant when PlayerScore is active in mission Description of feat when killing a submarine with ASW munitions, supports all PlayerScore wildcards.

7.3.2 aswZones

Name	Description
verbose	A value of “true” turns on debugging for the entire module. Default is “false”

7.3.3 aswGUI

Name	Description
verbose	A value of “true” turns on debugging for the entire module. Default is “false”
aswCarriers	Comma-separated list of types of player aircraft that are allowed to perform ASW. Defaults to DCS Common’s list of troop carriers (`Mi-8MT, UH-1H, Mi-24P`), and you can provide your own list: Example: “`Mi-8MT, UH-1H, C-130J`” removes the Hind and adds the Hercules fixed-wing transport to the list of legal ASW carft. Supports wildcard type endings: if a type ends on an asterisk (“*”) all types that match whatever precedes the asterisk are accepted. For example, `"Mi-*"` will match both `"Mi-8T"` and `"Mi-24P"`. You can supply the type `'any'` or `'all'` to allow all aircraft to perform ASW. Default <none> (use dcsCommon’s list of troop carriers for ASW)
buoyWeight	Weight (in kg) per ASW buoy Defaults to 50kg
torpedoWeight	Weight (in kg) per ASW torpedo Defaults to 700kg

7.3.4 aswSubs

Name	Description
verbose	A value of “true” turns on debugging for the entire module. Default is “false”
critDist	Distance (in meters) to trigger the submarine’s attack on a target vessel (see *targets*, below). The target vessel will receive multiple (see *salvoSize*) hits (see *explosionDamage,* below)
explosionDamage	Damage inflicted by attacking aswSub per hit. Defaults to 1000

Name	Description
salvoSize	A range (e.g., 2-4) that determines how many hits a target receives when attacked. Once a target ship has been hit, no other aswSub will attack that vessel. Defaults to '4-4' (always take four hits)
targets	Comma-separated list of group names that are targets for aswSubs. The group must exist in ME when the mission starts. Defaults to <none>

7.4 ME Integration

Name	Description
asw	Marks this zone as an ASW supply/drop zone **MANDATORY**
buoys	Number of buoys in stock in this zone. Setting this number to -1 sets the supply to infinite. Defaults to -1 (infinite supply)
torpedoes	Number of torpedoes in stock in this zone. Setting this number to -1 sets the supply to infinite. Defaults to -1 (infinite supply)
coalition	Coalition which owns the asw that are being dropped by this zone. If this zone is linked to a unit, that unit's coalition is used instead of any value given to this attribute. (only required when dropping ASW and zone is not linked to a unit) Defaults to 'neutral'
buoy?	Input flag that triggers the drop of an ASW buoy. Will only drop a buoy if there are enough left in the stores. Defaults to <none>
torpedo?	Input flag that triggers the drop of an ASW torpedo. Will only drop a torpedo if there are enough left in stores. Defaults to <none>
triggerMethod aswTriggerMethod	DML Method for input flags Defaults to 'change'

7.5 Demos

- Davy Jones' Rocker

8 BaseCaptured

8.1 Summary

This module generates a signal on the output flags when a base (Airfield, FARP, Ship) is captured (note that currently, ships cannot be captured).

8.2 Dependencies

dcsCommon, cfxZones

8.3 ME Integration

baseCaptured!	Marks this zone as a baseCaptured zone. It lists the flags that should be banged! when the closest base (FARP, Airfield, Ship with Helipad) to this zone is captured by another faction. **MANDATORY**
method captureMethod	DML method for output flags Defaults to 'inc'
blueCaptured! blue!	Flags to bang! when blue faction captures the closest base to this zone Defaults to <none>
redCaptured! red!	Flags to bang! when blue faction captures the closest base to this zone Defaults to <none>
contested!	Flags to bang! when closest base becomes contested and belongs to neither blue nor red Requires handleContested be true in configuration (default) Defaults to <none>
baseOwner	Flag that is set by the module to the current faction (0 = neutral, 1 = red, 2 = blue, 3 = contested) that currently holds this base.

8.4 Demos

- Count Base's Blues

9 BombRange

9.1 Summary

A module that provides bomb range training services to the mission

9.2 Dependencies

bombRange requires dcsCommon and cfxZones

9.3 ME Integration

Name	Description
bombRange	Marks this zone as a bombRange target zone. **MANDATORY**
percentage	If set to true, the target zone reports in percent how close to the center the impact was located. 100% means bullseye, 0% means outside. If set to false, any hit inside counts as 100% hit. Only works for circular target zones. If you try to enable percentage for quad-pased target zones, you will receive a warning at mission start, and the zone still reports all hits inside as 100% Defaults to true for circular zones, always false for quad-based zones.
details	When set to true, reports of an impact include the details of weapon type, pilot name, airframe type, total travel distance, impact velocity Defaults to true
reporter	When set to true, impacts are reported. Defaults to true
reportName	If true, the target zone's name is reported with "INSIDE target area' Defaults to false (zone name not added)
smokeHits	If set to true, impact locations inside a target area are marked with smoke Defaults to false (not marked with smoke)
smokeColor	Color of the smoke that marks hits inside the zone. Possible values for the smoke effect are: • "green" or "0" • "red" or "1" • "white" or "2" • "orange" or "3" • "blue" or "4" Only relevant if you set smokeHits to true. Defaults to "blue"

Name	Description
flagHits	Similar to 'smokeHits', except that an object is placed on the impact point instead of smoke. Called so because the default object that is placed on the impact location is a flag. Defaults to false (no objects placed on impact point).
flagType	Type Name of the object that is to be placed on the impact location. See here https://github.com/mrSkortch/DCS-miscScripts/tree/master/ObjectDB/Statics for a collection of possible objects. Only relevant if you set flagHits to true. Defaults to "Red_Flag"
clipDist	Maximum distance from center of the trigger zone that a hit is reported. Defaults to 2000 (meters = 2km = 1.2 miles)
method	DML method to bang! output flags. Defaults to "inc"
hit!	DML output. List of flags to beng! when a weapon fulfils the hit condition Defaults to <none>
markBoundary	If set to true, the outline of the target zone is marked with a number of objects for visual cues for pilots. All objects belong to Neutral Defaults to false (no marks)
markType	Type Name of the object that is used along the zone's boundary. See here https://github.com/mrSkortch/DCS-miscScripts/tree/master/ObjectDB/Statics for a collection of possible objects. Defaults to "Black_Tyre_RF"
markNum	Number of repeats for the boundary marker per quarter. The number of objects placed is one higher per quarter, so the total numbers of objects placed is (1+markNum) * 4. If, for example, you set markNum to 5, a total number of (1 + 5) * 4 = 24 objects are places around the target zone's outline. Defaults to 3 (a total of (1 + 3) * 4 = 16 objects along the outline)
markCenter	If set to true, the center of the bomb zone is marked by an object. The object belongs to Neutral. Default to false (no object to mark center)
centerType	Type Name of the object that marks the center. See here https://github.com/mrSkortch/DCS-miscScripts/tree/master/ObjectDB/Statics for a collection of possible objects. Defaults to "house2arm"
markOnMap	If set to true, the bombZone is marked on the F-10 map. Default to false (no mark in F-10 Map)
mapColor	Four numbers, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to

Name	Description
	draw the zone's outline on the map. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to "0.8, 0.8, 0,8 1.0" or #CCCCCCFF – a light gray, fully opaque
mapFillColor	Four numbers, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone on the map. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to "0.8, 0.8, 0,8 0.2" or #CCCCCC33 – a light gray, 80% transparent

9.4 Demos

- Bombs Away

10 Cargo Receiver

10.1 Summary

A zone designed to generate an event / signal when cargo is landed inside the zone. Cargo must be registered with CargoManager (happens automatically when using an objectSpawner)

10.2 Dependencies

dcsCommon, cfxZones, cfxCargoManager.

10.3 ME Integration

Name	Description
cargoReceiver!	Output flag to bang! when the cargo object is delivered onto the ground inside the zone. Marks this zone as a cargo receiver zone. **MANDATORY**
~~cargoReceiver~~ **deprecated**	~~Marks this zone as a cargo receiver zone. Value is ignored~~ **deprecated**
autoRemove	Delete (despawn and remove from game) a cargo object <removeDelay> seconds after it was successfully delivered. This is helpful for most ObjectSpawnZones set-ups to trigger their spawn cycle Defaults to false
silent	Set to true to turn off this zone's directions. Defaults to false (zone will talk to pilots)
method cargoMethod	DML Method for output Defaults to "inc"
~~f!~~ ~~cargoReceived!~~ **deprecated**	~~The flag to bang! when the object is destroyed. Use only one synonym per zone.~~
removeDelay	The delay (in seconds) after which after a successful delivery of a cargo object is removed. Requires that autoRemove be set to true. Defaults to 1 (second), Minimum is 1
noBounce	Prevents multiple deliveries of the same cargo object to the same zone are counted multiple times (e.g. bounces). When set to true, each cargo can only be delivered once per receiver zone. Defaults to true (one delivery per zone)

10.4 Demos

- Helo Cargo

11 Changer

11.1 Summary

This module provides a convenient way to transform flags on-the-fly, and to provide a flexible gated switch

11.2 Dependencies

dcsCommon, cfxZones

11.3 ME Integration

Name	Description
change?	The input flag whose value is used to create the output signal **MANDATORY**
out! changeOut!	The output flag.
triggerMethod triggerChangeMethod	Watchflag method that is used to interpret change? when inEval is set to true Defaults to 'change'
inEval	When set to true, the input flag change? is interpreted as a watchflag under triggerChangeMethod's rules Defaults to false
changeTo to	Operation to apply to the input flag to create the ouput value. The following operations are defined: • `'bool'` Output value is 0 if input is 0, 1 otherwise (conversion to bool) • `'not'` Output value is 1 if input is 0, 0 otherwise • `'sign'` Output value is -1 if input is <0, 1 otherwise • `'abs'` Output value is the absolute of input value (e.g. outputs '3' for '-3' and '3') • `'negative'` Output value is input value multiplied by -1, turning the sign. • `'direct'` Output value is the same a input value. Used primarily with when changer is functioning as a gated switch Defaults to "direct"
min	When defined, ensures that the output value has this value at minimum
max	When defined, ensures that the output value has this value. If max is less than min, output is always set to max

Name	Description
paused changePaused	When set to true, the changer starts in paused/off mode and the output flag is not changed. The only way to turn a paused changer on when off/paused is via a signal on the changeOn? input Defaults to false.
on? changeOn?	Turns a paused changer on, enabling transmission of the input signal (after processing) to output, opening the 'gate'. Triggers on "change" Defaults to <none>
off? changeOff?	Turns a changer off, pausing it. Input signals are no longer processed nor propagated to output, closing the 'gate'. Triggers on "change". Defaults to <none>
On/Off? changeOn/Off?	Flag that when defined controls transmission of input to output when the changer in running (i.e. not paused). The value of this input controls the gate as follows: • 0: gate closed, no transmission • Anything else: gate open. Defaults to <none>

11.4 Demos

- Gate and Switch

12 CivAir

12.1 Summary

Drop-in to generate civilian air traffic. Runs out of the box, can be easily customized.

12.2 Dependencies

dcsCommon, cfxZones

12.3 ME Integration

CivAir is customized via a config zone "**civAirConfig**" with the following attributes:

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
DCS	By default, civAir adds a set of standard DCS aircraft types (Yak-40, C-130, C-17A, IL-76MD, An-30M, An-26B) to its list of civil aircraft. You can turn that off by setting the attribute's value to false Defaults to 'true' (add above mentioned standard DCS types)
CAM	If you set this attribute to true, civAir also adds the following airfcaft types to its list of available aircraft: A_320, A_330, A_380, B_727, B_737, B_747, B_757, Cessna_210N, DC_10 These aircraft types aren't included in a standard DCS install, and using such an aircraft type simply results in a non-spawning civil aircraft. The types listed above are aircraft that come with a popular mod called "CAM". If that mod isn't installed when DCS runs a mission that tries to access such an aircraft type, the relevant flight simply will not spawn. Defaults to false (no CAM types added tp civAir)
aircraftTypes	A comma-separated list of Types (as defined in https://github.com/mrSkortch/DCS-miscScripts/tree/master/ObjectDB/Aircraft) that define the aircraft types used for civilian flights. These must be fixed wing aircraft (i.e not helicopters). **When present, all airframe types are picked from this list**, and each entry has the same chance to be picked. This means that if you list the same type twice, you increase the chance of that type to be picked. If, for example, you list `Yak-40, Yak-40, IL-76MD` During the mission two thirds of all civilian flights will be performed by Yak-40, and the remaining third by IL-76MB. The list that you provide here overrides any list that was assembled by the attribute values for "DCS" and "CAM" (see above), or provided in a civil_liveries zone.

Name	Description
	If you accidentally misspell a type name, or that type does not exist for your DCS installation, DCS will to either of two things: • Create no flight at all (some versions of DCS) • Create a civilian flight of a SU-27 with fantasy “placeholder” livery – or some other random airframe/livery combo)other versions of DCS) Neither will disrupt your mission in any way, so using this feature is safe.‘ Defaults to <none> (aircraft used are taken according to the settings of the attributes DCS and CAM, and whatever you supply in the civil_liveries zone
owner	Country ID that all aircraft spawned by civAir belong to. Defaults to 82 (UN Peacekeepers)
ups	Number of updates per second that civAir checks on its flights. By default, this is 0.05, or once every 20 seconds.
maxTraffic maxFlights	Maximum number of civilian flights at the same time. Defaults to 10
maxIdle	Number of seconds of an aircraft idling that can elapse before it is removed. CivAir determines that an aircraft is idling by checking if it is moving. If you set this number too low, a cold-starting aircraft may be removed before it can move. Defaults to 480 (seconds = 8 minutes)
initialAirSpawns	Controls if at mission start half of maxTraffic immediately spawn in mid-air to start a mission with planes in the air.

civil_liveries
With this zone you can add any aircraft type and a list of liveries to be used for that type

Name	Description
Type_name Example: YAK-40	List if livery names, separated by comma. Example: Aeroflot, Algeria GLAM, Olympic Airways

You can add airfields to the inclusion / exclusion set with trigger zones as follows:

Name	Description
civAir	When present, the airfield closest to this trigger zone can be added to civ air’s airport list. The following values are supported: • ‘`closed`’ or ‘`exclude`’ No civilian flights originate or land here. This overrides any other settings for this airfield

Name	Description
	• ‘`departure`’ or ‘`depart only`’ Civilian flights can depart only from this airfield • ‘`arrival`’ or ‘`arrive only`’ Civilian flights can only land here • ‘`in`’ or ‘`inbound`’ – DOES NOT ASSOCIATE AN AIRFIELD Civilian flights can appear in this trigger zone, in-flight and at altitude. This is intended to be used to simulate off-map long-distance flights that now enter the map arena. Instead of an associated airfield’s name, this zone’s name is used for the flight’s origin • ‘`out`’ or ‘`outbound`’ – DOES NOT ASSOCIATE AN AIRFIELD Civilian flights that reach this trigger zone disappear. This is intended to simulate an off-map connection, where a long-distance flight leaves the map. Instead of an associated airfield’s name, this zone’s name is used for the flight’s origin • ‘`in/out`’ or ‘`in/outbound`’ – DOES NOT ASSOCIATE AN AIRFIELD Civilian flights can both appear in this trigger zone, and disappear when they reach it as their destination. This is intended as both a source and destination point for flights that have an off-map source and/or destination. Instead of an associated airfield’s name, this zone’s name is used for the flight’s origin or destination • *Any other value, or nothing* The airfield closest to this zone is an open airfield for departing and arriving civilian flights Note that you can add the same airfield multiple times by adding multiple trigger zones with a civAir attribute in proximity the same airfield. This will proportionally increase the likelihood that the airfield is picked for destination or departure Note also that only functioning airfields are chosen. FARPS or ships are disregarded. **MANDATORY**

12.4 Demos

- Virgin (Civ) Air
- Air Caucasus II
- One-Way Air
- Civ Air International
- Types and civil liveries

13 Civ Helo

13.1 Summary

Provides civilian (non-aggressive) helicopter traffic to your map. Flights depart one civHelo zone and fly to another civHelo zone

13.2 Dependencies

civHelo requires dcsCommon and cfxZones

13.3 ME Integration

Name	Description
civHelo	Tells DML that this zone is a civHelo zone that can be used for flights **Mandatory**
land	If set to true, civHelo flights can land here (use the zone as a destination for a flight) Defaults to TRUE (civHelo flight can use this as destination)
start	If set to true, civHelo flights can take off from here (use the zone as a starting point for a flight) Defaults to TRUE (civHelo flights can use this a departure point for a flight)
hot	If set to true, civHelo flights that depart from here start hot. Otherwise, the flight is cold-started Defaults to FALSE (flight is cold-started)
types	The DCS types of helicopters, separated by comma (“,”) that can start from here. Example: `Mi-8MT, SA342L, UH-1H` Defaults to the same types that are supplied with the config zone.

13.4 Demos

- Civvy Heli

14 Clone Zone

14.1 Summary

Allows you to clone groups and static objects, any time, anywhere.

14.2 Dependencies

dcsCommon, cfxMX, cfxZones

14.3 ME Integration

Name	Description
cloner	Marks this ME Zone as a clone zone. The **Value of this attribute is ignored**, use it to describe this cloner's function. **MANDATORY**
source	The source for the clone template: must be the name of a clone zone. When a clone cycle is initiated, the template is fetched from the source zone, and the units are then spawned around If this zone's center. If this attribute is present, this zone is not scanned for units to create a template from, and no units are removed at start. If you supply **more than one template zone names**, separated by comma (e.g., "SAM 9 small, SAM 9 big"), each time a clone cycle is initiated, the clone zone randomly picks a template from that list. Defaults to <not present, zone scanned for units to create a template from>
turn	Degrees in which the clones are turned relative to the template's original position, relative to the zone's center. Defaults to 0 (zero)
moveRoute	If this attribute's value is true, all waypoints are move the same amount as the cloned units upon spawn. Only relevant if the zone is cloning another zone's template. When not present or false, all spawned units use the template's waypoints. When the clone zone is using *linkedUnit* (moving DML zone, a Core Attribute of all DML zones) in conjunction with *useHeading* (another Core Attribute), the entire route is also rotated to coincide with the linked master unit's heading differential to its original ME heading. Defaults to false
onStart	When set to false (default), the cloner will not spawn during start. Note that if this spawn zone is used to create a template, this results in an empty zone, as all units used for the template are destroyed during template creation. Defaults to false (**no spawn on start** of mission). **To spawn units at mission start, set this attribute to true.**

Name	Description
masterOwner	If **not** present, all *cloned units retain ownership from the template*. This can be problematic if you need to change the ownership of the spawned units (for example when you import templates via source, or the clone zone can be captured). If present, all cloned units are spawned for the faction (red/blue/grey) **that owns the zone named masterOwner**. There's a convenient shortcut to set the masterOwner to this clone zone (see below) **CONVENIENCE SHORTCUT:** When the master owner is set to "*" (Asterisk) wildcard, it is set to the same zone as the cloner. This is a convenience shortcut for cloners that import foreign templates and convert them to their own faction, and to enable a cloner that can be conquered to spawn its own template according to the faction that owns the spawning cloner. Note that when you use the masterOwner attribute, the coalition for which this cloner spawns units can change with the ownership of masterOwner. This can be used with cloners that import templates to easily control spawning the same template that aligns to different factions by setting the spawning cloner's 'owner' to a faction, set masterOwner to itself Defaults to <none, retain **template** ownership>
spawn? ~~f?~~ ~~in?~~ clone?	Input to trigger a clone cycle. Defaults to <none> This input has multiple synonyms. Use only one per zone.
despawnIn	If you add this attribute, all clones created by this cloner will be automatically de-spawned (removed from the mission) after the amount of seconds that you specified here. If the clones were already destroyed before the time runs out, nothing happens. *despawnIn* is compatible with the *empty!* output (meaning: if the last batch is auto-destroyed due to *despawnIn*, that will create the appropriate output signal. despawnIn supports a single number and range. If you supply a range (e.g. "600-1200"), each group (but not unit inside the group) or static object is assigned their own despawn time. Defaults to: <none> (no automatic despawn)
triggerMethod cloneTriggerMethod	Watchflag method to tell what to look for in inputs. Defaults to "change"
preWipe	If this attribute is true, any remaining units from the previous cloning cycle are removed from the game when the next clone cycle starts. Use this to 'refresh' groups like SAMs or Tanks that can run out of ammo.

Name	Description
	If units are preWiped, removing the remaining units happens immediately, while spawning of new units is delayed for 0.5 seconds to allow DCS to internally resolve ground height and prevent spawned new units to 'fall to the ground' or spawn suspended in mid-air. Default: false (no pre-wipe)
declutter	If this attribute is set to true, all debris and wrecks inside the clone zone are removed before a clone cycle is started. Use this to remove debris from destroyed units before re-spawning units in the same zone. Default: false (no declutter)
empty!	Output to send a signal to when all units from the last spawn have been destroyed (this includes all static objects). **Note:** There are other outputs (for example damaged! and health#) that you can use to monitor the health of the last batch of clones and can get alerted if one of more units in your last clone batch get destroyed. See those attributes below. Defaults to <none>
method cloneMethod	DML Method for output flags Defaults to "inc"
deSpawn? deClone? ~~wipe?~~	Input that if triggered causes all remaining units / static objects from the previous spawn to be removed. Note that if you trigger deSpawn?, empty! will **not** trigger. **Please note that the wipe? attribute is deprecated as it conflicts with the "Wiper" module's key attribute.** Note that in the same cycle, deSpawn? is triggered **before** spawn?. This means that you can easily use common command for all cloners to wipe the last spawn before spawning a new cycle in setups that use randomized groups of cloners. Defaults to <none>
trackWith:	List of groupTracker zones. All spawned groups are added to these groupTrackers. If you have stacked the tracker on the same zone as the cloner, you can use a single asterisk '*' as zone name. Supports a comma-separated list of trackers if you simultaneously want to pass the cloned groups to multiple trackers, e.g. "GroundTrack, HeloTrack" This is useful if your cloner clones more than one group, and your trackers use filtering. Defaults to <none>
useDelicates	Name of a Delictates Zone that is used to assign delicate status when this spawner spawns objects. As with the *trackWith:* attribute, you can use "*" to refer to this zone.

Name	Description
	Defaults to <none>
randomizedLoc rndLoc	Upon cloning, each cloned unit/object is placed randomly inside the zone. Works with moving zones. Works with quad-based zones. Defaults to false (use template's placements)
rndHeading	Upon cloning, each cloned unit/object assumes a random heading. Defaults to false (keep template's heading)
triggerMethod cloneTriggerMethod	Watchflag method for input flags. Defaults to 'change'
onRoad	Set to true to have any cloned unit deploy on the nearest road to the location it would spawn otherwise. This is very likely to change the template's formation, and can be a long way from the clone zone's intended spawn location, so use carefully. May also have other effects, such as bunching up the spawned units in unrealistically close proximity to each other. If set to true, it overrides whatever settings are used for onPerimeter and inBuiltup. Defaults to false (keep template's position)
wholeGroups	A modifier for the rndHeading, onRoad and rndLoc attributes. When set to true the clones are affected as follows: • rndLoc - randomizes the location of unit 1 of the group, and keeps all other vehicles in formation relative to that group. In fact, it only randomizes the group's position, not individual units • onRoad - now only the first unit is placed on the center of a road, all other units remain in location • rndHeading - with centerOnly, all units rotate randomly around unit 1's location Defaults to false (all units individually)
onPerimeter	**A modifier for rndLoc.** If this attribute is set to true, the units are distributed along the edge of the trigger zone. Can be used alongside *wholeGroups*. Is overridden when *onRoads* is set to true. When set to true, overrides any setting of *inBuiltup*. Defaults to false (keep template's positions).
inBuiltup	**A modifier for rndLoc.** If set, the units are placed on the map with at least as much clarence (in meters) to the center of any map objects in the vicinity. Use this when you want to spawn units on a free spot in map regions that have lots of houses like cities (hence the 'builtUp' name). **Note:** There is no guarantee that DML can find a sufficiently open (free) space inside the destination trigger zone. For example, some densely populated areas on the "Syria" or "Sinai" map are so densely packed with objects that it is impossible to quickly find a free space that is 20 meters wide. After several tries, it simply gives

Name	Description
	up and places the unit randomly without guarantee that the inBuiltup constraint (clearance) is kept. **Recommendation**: use 15 (meters = 45 feet). If that works well, you can try and decrease. Is overridden if either *onRoads* or *onPerimeter* is set to true. Defaults to <none>
nameScheme	By default, a clone zone uses a simple name scheme to ensure that all cloned units have a mission-unique name. The default name scheme is the original unit's name (as taken from the template), plus a hyphen "-", plus a DML-unique number (e.g. "76791") In certain situations, mission designers need or want a tighter control over how cloners name units: for better (easier to debug) "speaking" unit names, to achieve special effects (intentionally replace existing units), or to be able to interface with more predictably named units (e.g. linkedUnit attribute for moving zones) or other scripts. When you are using a name scheme for a clone zone, the onus to ensure that you don't accidentally replace an existing unit of the same name is on you. See 'Custom naming of cloned units' in this section for details. Clone Zone's name schemes supports wildcards that are filled in at runtime as follows: • `<o>` the original (as defined in the template) name of the unit • `<z>` name of the spawning cloner's zone • `<s>` name of clone zone to whom the template belongs that is currently spawning. If the cloner does not use a *source* attribute, it is the same as <z> • `<uid>` creates a DML-wide unique number. It is guaranteed to be unique across all clone zones. • `<i>` creates a group-local unique number valid only for this spawn cycle. It increases by one (1) with each use, and reset to 1 whenever a new group is cloned • `<lcl>` creates a zone-local unique number. It increases by one (1) with each use (unit or groups). This number is not shared among other clone zones, so it is not guaranteed that this number is unique across multiple cloner zones. • `<g>` creates a module-local unique number. It is increased by one (1) with each use. This number is guaranteed to be unique across all clone zones.

Name	Description
	Defaults to <none> (this means that clone zones uses its default naming scheme which is equivalent to `<o>-<uid>`
groupScheme	Like for units, a clone zone uses a simple name scheme to ensure that all cloned **groups** have a mission-unique name. The default name scheme is the original group's name (as taken from the template), plus a hyphen "-", plus a DML-unique number (e.g. "76791") Like for units, you can change the way how the cloner names groups (this is always independent from how it names units) Clone Zone's name schemes supports wildcards that are filled in at runtime as follows: • `<o>` the original (as defined in the template) name of the unit • `<z>` name of the spawning cloner's zone • `<s>` name of clone zone to whom the template belongs that is currently spawning. If the cloner does not use a *source* attribute, it is the same as <z> • `<uid>` creates a DML-wide unique number. It is guaranteed to be unique across all clone zones. • `<i>` always returns "1" • `<lcl>` creates a zone-local unique number. It increases by one (1) with each use (units or groups). This number is not shared among other clone zones, so it is not guaranteed that this number is unique across multiple cloner zones. • `<g>` creates a module-local unique number. It is increased by one (1) with each use. This number is guaranteed to be unique across all clone zones. Defaults to <none> (this means that clone zones uses its default naming scheme for groups which is equivalent to `<o>-<uid>`
identical	If this attribute is set to true, each time a clone cycle is run, the cloned units receive identical name and ID as the template. This results in clones that • Replace any previous clones from this template that were cloned with 'identical = true' • Can still be different in location, coalition etc. *identical* and *nameScheme/groupScheme* are mutually exclusive. When identical is true, it overrides any nameScheme settings. Defaults to false
requestable	If set to true, this clone zone can be controlled via other modules, such as HeloTroops. Availability of this cloner to other modules is

Name	Description
	controlled by those modules (for example, HeloTroops will only control cloners that clone units that can be carried with a helicopter) Note that “requestable:true” and “source:<templates>” are usually mutually exclusive, so you will se a warning when the cloner module starts up and a zone sports both “requestable” and “source” attributes Defaults to <none>
cooldown	If set to a value greater than zero, a cloner can only start a new clone cycle after <cooldown> seconds have passed since the last clone cycle. Defaults to -1 (no cooldown)
useAI	Only works for ground and naval units. If set to false, the spawned groups have their AI disabled and will neither move nor shoot nor otherwise react to enemies. Defaults to true (AI is turned on)
damaged!	Whenever the current batch of clones receives so much damage that one or more units are destroyed, a signal is sent on the connected flags. Use this when you want to be notified whenever your last clone batch receives significant damage, but that damage is not enough to entirely destroy all units. Note that only AI units are counted, static objects are not monitored. Defaults to <none>
health#	Direct output. Always contains current health in percent of the current batch of clones (AI units only, does not include static objects). 100 means 100% (all units are still alive), 0 means all units are dead. Defaults to <none>

14.4 Demos

- Attack of the CloneZ
- Once, twice, three times a maybe
- Clone Relations
- Frog Men Training
- Flag Fun
- Track This!
- Gate and Switch
- Reinforcements A La Carte
- Forever-looping Spawners
- Recon Mode – Reloaded
- Willie Nillie
- BFM Combat Trainer
- Cleanup Crew
- Airfield Mine
- Send in the Clones

15 Convoy

15.1 Summary

A module that manages ground vehicle based convoys and their helicopter escorts.

15.2 Dependencies

Convoy requires *dcsCommon*, *cfxZones* and *cfxMX*.

Optional: *radioMenu* (when using the *"attachTo:"* attribute)

Convoy automatically integrates with *twn* if it is present in the mission

15.3 ME Integration

Name	Description
convoy	Marks this zone as a convoy Zone. All Ground groups that have at least one vehicle inside this zone become convoys. You define the route and the orders for those groups using Mission Editor. If more than one ground vehicle group is inside the convoy zone, one is chosen by random each time the zone is signaled to spawn. All helicopter groups inside this zone become possible helicopter escorts. Only one group (or less, see *pEscort*) is chosen to escort the convoy in its way. Remember to set up the helicopter's loadout. **MANDATORY**
coalition	The coalition that the convoys spawned by this zone belong to. Defaults to NEUTRAL
dynamic	EXPERIMENTAL – do not use
Identical unique	EXPERIMENTAL – do not use
preWipe	When set to true, all previously spawned convoys are removed from the mission. Defaults to false (spawns do not remove previous convoy)
endWipe	When set to true, a convoy that reaches their last waypoint is removed from the map. This will prevent vehicles massing around the last waypoint when multiple instances of the same convoy reach their destination. Defaults to true (convoys are removed when they reach the last waypoint)
killWipeDelay	Delay (in seconds) for escorting helicopters to be removed after all ground vehicles of the convoy are destroyed. This allows you to have the escort helicopter loiter like 'angry hornets' for a while before they, too, are removed. Defaults to 300 (5 Minutes)
pEscort	Probability (in Percent) that the convoy is escorted by helicopters. This requires that at least one helicopter group is present inside

Name	Description
	the convoy zone. For example, if pEscort to 25, there is a 25% chance that a convoy when spawned also receives an escort. If pEscort is smaller than 1, that convoy never receives a helicopter escort. Defaults to 100 (always escort if helicopters are present)
onStart	If set to true, a convoy is spawned from this zone at the start of the mission. Defaults to false (no spawn on start)
wpUpdates	If set to true, coalitions receive an update each time that a convoy reaches a waypoint. There are separate messages for the first waypoint (when the convoy is spawned), last waypoint (at least one ground unit of the convoy reaches destination), and in-between waypoints. When a convoy reaches the penultimate waypoint (and assuming that there are at least 3 waypoints in the convoy's route), the enemy coalition also receives a separate notification. Defaults to true (waypoint updates are sent)
attackWarnings	If set to true, the module sends out warnings to the coalition that owns the convoy when it gets attacked and at least one unit is destroyed. Only one warning per attacked convoy is sent within 5 minutes. Defaults to true (coalition receives warnings)
spawn?	Input. When triggered, this zone spawns a new convoy Defaults to <none>
dead!	Output: flag to change when all ground vehicles of a convoy spawned by this zone are destroyed. Defaults to <none>
attacked!	Output: flag to change when ground vehicles of a convoy have been destroyed since the last check, and the convoy isn't completely destroyed. Defaults to <none>
arrived!	Output: flag to change when the convoy (i.e at least one living ground unit) has reached its destination. Defaults to <none>
convoyMethod	DML method that is sent to the output. Defaults to "inc"
convoyTriggerMethod	DML method that triggers the input. Defaults to "change"

15.4 Demos

- Escort convoy to town

16 Count Down

16.1 Summary

A module that counts down flag signals (not time!) and can do various tricks with these signals.

16.2 Dependencies

dcsCommon, cfxZones

16.3 ME Integration

Name	Description
countDown	Marks this as a count down. The value of this attribute defines the number times that this module can be triggered until the count reaches zero (“limit”) This value supports ranges: if you specify a range (e.g., “3-5”) each time that the count down is initialized (at start, at reset, and when looping), a random number in the range (including upper and lower limit) is chosen as the start value. Defaults to 1 (one) **MANDATORY**
loop	If this attribute is true, a count down restarts after reaching zero. If the count down is given as a range, a new random start value is taken from that range (including upper and lower limit)
method ctdwnMethod	DML Output method. Use only one synonym per zone. Defaults to “inc” (increment flags connected to output)
~~**count?**~~ in? clock?	Main input. Every time that a command/signal is received on this input, the count down proceeds down by one. Use only one synonym per zone. Defaults to <none>
triggerMethod ctdwnTriggerMethod	DML Input method that determines how inputs trigger. Defaults to “change”
zero! out!	DML output. The command/signal to send when the countdown reaches zero. Use only one synonym per zone. Defaults to <none>
tMinus!	DML output. The command/signal to send when count is lowered, and has not yet reached zero. Defaults to <none>
belowZero!	DML output. the command/signal to send every time that the count is decreased *and* is below zero. Note that if the ‘loop’ attribute is set, this can’t happen. Defaults to <none>
counterOut#	Direct value output. Flags that are connected to this output are set to the current value of the count. The flags are updated each time that the counter is triggered via the clock?/in? input Defaults to <none>
disableCounter?	DML input. When triggered, this turns off (halts) the count down. Signals on the clock? Input are ignored while the counter is disabled

Name	Description
	Defaults to ‘<none>’
enableCounter?	DML input. When triggered, a disabled counter continues the count down. The count down picks up where it left off when it was disabled. Has no effect if the counter is enabled. Defaults to ‘<none>’
reset?	DML input. When triggered, it restarts the count down. If the count down is randomized, a new random start amount is selected. Defaults to <none>

16.4 Demos

- Once, twice, three times a maybe
- The Zonal Countdown

17 Counter

17.1 Summary

Counter simply changes the output as often as it triggers on the input. This can be a simple counter, but if you choose the correct input and output methods, this can be pure magic.

17.2 Dependencies

Counter requires dcsCommon, cfxZones.

17.3 ME Integration

Name	Description
count?	Tells DML that this is a counter that is triggered by this input **MANDATORY**
triggerMethod triggerCountMethod	Watchflag – the condition that triggers the input. Defaults to “change”
method countMethod	Output method. Defaults to “+1”
out! countOut!	The output to change when the counter is triggered. Defaults to <none>

17.4 Demos

- Formation Trainer

18 CSAR Manager

18.1 Summary

CSAR Manager is a stand-alone module that adds CSAR (Combat Search And Rescue) operations to your missions. It consists of multiple ME parts, and in-game UI.

18.2 Dependencies

CSAR Manager requires dcsCommon, cfxZones, cfxPlayer, nameStats, cargoSuper

Optional: commander (when using CSAR Zones), names (when using random mission names)

18.3 ME Integration

ME Integration consists of two parts: placing CSARBASES, which are zones in which a successful landing completes the CSAR mission (and returns a lost pilot when using “limited airframes”), and placing CSAR Zones that start CSAR Missions by placing downed pilots on the map on-demand

Name	Description
CSARBASE	Must be present to identify this zone as CSAR Base where CSAR Missions can end. A helicopter that lands inside this zone completes CSAR missions for those pilots that it is carrying. Supports linked zones (for example if the BSAR Base is a ship). Each side that has CSAR Missions must have at least one such zone, or CSAR Missions cannot be completed. There is no upper limit on the number of CSAR Bases a side can have. The value of this attribute defines which sides can complete a CSAR mission here. • “red” or 1 means that only red pilots can complete their missions here, • “blue” or 2 means blue only, and • “neutral” or “0” means that all sides can complete CSAR missions in this zone. Defaults to “neutral” – all sides can complete CSAR Missions in this zone. **MANDATORY**
name	Optional name for CSARBASE. Defaults to trigger zone’s name

Name	Description
CSAR	Identifies this as CSAR Zone that is converted into a CSAR mission upon mission start or when the value of the startCSAR? flag changes.

	Contains the name of this mission, recommended is to use a personal name, e.g. “Lt. Wesley Crasher” Names can be randomized by setting the name to “*rnd”. In this case, each mission has a random name (first and last name). Using this option requires the “**names**” module to be present. **MANDATORY**
coalition	Faction (red/blue) for which this mission is generated
freq	Frequency for the ELT (radio to home in on) in KHz. Random if not set
timeLimit	(currently not used)
weight	Weight of pilot (tbc)
deferred	If true, CSAR missions are only created when the startCSAR? flag changes. Default is false (a CSAR mission is automatically created when the main mission starts up)
in? start? startCSAR?	When the value of this flag changes, a new CSAR is created based on this Zone’s attributes. Defaults to <none>
score	Number of points to award when rescued successfully. Only relevant when PlayerScore is installed. If not present, csarManager’s default score is used.
triggerMethod	DML Method that triggers inputs. Defaults to “change”
rndLoc	If set to false, the evacuee spawns at the CSAR Zone’s center, or at a random location inside the CSAR zone otherwise. Note that CSAR Zones support polygonal zone shapes Defaults to TRUE (random spawn location inside the zone)
onRoad onRoads	If set to true, the evauee spawns on the nearest road closest to the intended (even when randomized) spawn location. Note that this can cause evacuees outside of the CSAR zone. If you enable *onRoad*, that overrides any setting you many have for *inBuiltup* / *clearance* (see below). Defaults to false (units are not moved to the nearest road)
clearance inBuiltup	If given, the units are placed on the map with at least as much clearance (in meters) to the center of any map objects in the vicinity. Use this when you want to spawn units on a free spot in map regions that have lots of houses like cities (hence the ‘builtUp’ name). **Note:** There is no guarantee that DML can find a sufficiently open (free) space inside the destination trigger zone. For example, some densely populated areas on the “Syria” or “Sinai” map are so densely packed with objects that it is impossible to quickly find a free space that is 20 meters wide. After several tries, it simply gives up and places the unit randomly without guarantee that the inBuiltup constraint (clearance) is kept. **Recommendation**: use 15 (meters = 45 feet). If that works well, you can try and decrease. Is overridden if *onRoad* is set to true.

	Defaults to <none>

18.4 Configuration

The lion's share of CSAR manager is controlled via the central configuration zone:

Name	Description
verbose	Set to true to turn on debugging information. Defaults to false
ups	Updates per second. Defaults to 1
useSmoke	When approaching a mission target, activate smoke or not. Smoke can have performance impact when close in a helicopter, so be mindful of this option Defaults to true (activate smoke)
smokeColor	Color of the smoke to pop when helicopter is in range and enabled. Understands numbers (0-4) and names (green, red, white, orange or blue) Defaults to "blue"
useFlare	When a player approaches the evacuees, release a flare. This is especially helpful in night-time recovery. The flare is released 5-10 seconds after the player comes into range Defaults to true (launch a flare)
flareColor	When flares enabled, the color of the flare that is launched. Possible colors are -1, "random" or "rnd" for random, or 0 (zero) to 3 or one of "green", "red", "white", or "yellow" Defaults to "red"
csarRedDelivered!	Flag to change when a red coalition-flown CSAR mission ends successfully
csarBlueDelivered!	Flag to change when a blue coalition-flown CSAR mission ends successfully
csarDelivered!	Flag to change when a CSAR mission ends successfully
rescueRadius	Helicopter must land within this distance (in meters) to the target to pick up. Recommended Value: 70
hoverRadius	When attempting a hover rescue, helicopter must stay within this range (in meters). Recommended value: 30
hoverAlt	When attempting a hover rescue, helicopter must stay below this altitude (in meters). Recommended value: 40
rescueTriggerRange	When approaching a mission target, the mission triggers a message from the evacuees at this range. This is also the range at which smoke is triggered if enabled
beaconSound	Name of sound file (ogg or wav) to play on the ELT frequency. Includes extension. Example: "Radio_beacon_of_distress.ogg"
pilotWeight	Weight for an evacuee in kg. Recommended Value: 100
hoverDuration	Time required to hover above pilot to secure winch and complete rescue
rescueScore	Default number of points awarded for a successful rescue. Awarded upon delivery in CSARBASE. Requires PlayerScore module to have effect
vectoring	If set to false, the mission report will no longer give range and bearing to the downed pilots. Default is true (range and bearing is provided) Defaults to true (vectoring enabled)

actionSound	Sound file to play on notifications
troopCarriers	A list of helicopter types that are allowed to carry/rescue troops in this mission. Defaults to DCS Common's list of troop carriers (which is usually `Mi-8MT, UH-1H, Mi-24P`), but you can provide your own list (for example to add non-official types). Example: "`Mi-8MT, UH-1H, SA342Minigun`" removes the Hind and adds the Gazelle in Minigun configuration to the list of legal troop carriers. Supports wildcard type endings: if a type ends on an asterisk ("*") all types that match whatever precedes the asterisk are accepted. For example, "`Mi-*`" will match both "`Mi-8T`" and "`Mi-24P`". You can supply the type '`helo`' to allow all player helicopters (including unofficial Mods like Blackhawk) to carry troops. You can supply the type '`any`' or '`all`' to allow all player units to carry troops. Default <none> (use dcsCommon's list of troop carriers)

18.5 Demos

- CSAR of Georgia
- On the record

19 Delay Flags (“Timer”)

19.1 Summary

Change a flag on the output side after a timer runs down. Can be paused, continued, reset.

19.2 Dependencies

dcsCommon, cfxZones

19.3 ME Integration

Name	Description
timeDelay	Marks this as a delayFlag/Timer module. The value of this attribute defines the number of seconds to wait after activation before the output flag is set. Value can be a range in which case delayFlag picks a random number inside the range (including bounds). Defaults to 1 second **MANDATORY**
out! delayDone!	The flag to bang! after the delay has passed. Use only one synonym per Zone
method delayMethod	DML Flag method for output Defaults to “inc”
f? in? startDelay?	Watchflag for a change that starts the delay. Use any synonym, but only one per zone.
triggerMethod delayTriggerMethod	DML Method for Watchflags
stopDelay?	Watchflag that stops a running delay. Has no effect on a stopped delay
pauseDelay?	Watchflag that when triggered pauses the delay to be continued later. Has no effect when triggered when the timer isn’t running or is already paused. Defaults to <none>
continueDelay?	Watchflag that when triggered continues a paused delay. Has no effect when triggered when the timer isn’t running or not paused. Defaults to <none>
~~delayLeft~~ delayLeft#	Flag that carries the number of seconds left on the timer (including when delay timer is paused). Carries -1 when timer is not running. Defaults to <none>

19.4 Demos

- Attack of the CloneZ
- Bottled Messages
- Clone Relations
- Flag Fun
- Track This!

20 Delicates

20.1 Summary

Makes units/objects/cargos brittle: they immediately explode when they receive some damage.

20.2 Dependencies

Delicates requires dcsCommon and cfxZones.

20.3 ME Integration

Name	Description
delicates	Marks all static objects and units that are inside this zone when the mission starts up as 'delicate'. If they get damaged at all, they immediately explode. Note that if these units move later outside this zone, they remain delicates, while units that move inside this zone later do NOT become delicates. Value of this attribute is ignored **MANDATORY**
power	Strength of the explosion when the delicate object is triggered. Defaults to 10
f! out! delicatesHit!	Flag to bang! when one of the units/objects defined by this zone explodes. Defaults to <none>
method delicatesMethod	DML Method for output flags Defaults to 'inc'
remove	When set to true, the delicate object/unit is removed from the game when it explodes. Defaults to true
triggerMethod delicateTriggerMethod	Watchflag method for all inputs
blowAll?	Sending a signal on this input causes all surviving delicates defined by this zone to blow up. When you feed back the delicatesHit! Signal into this input, one hit kills all.
safetyMargin	A number that defines how much damage the object can sustain (relative to its initial life) before it explodes. Expressed as a fraction from 0 (0%) to 1 (100%). For example, 0.1 means that the object can sustain 10% damage before it explodes. Defaults to 0 (any damage makes it blow up)

20.4 Demos

- Delicate Subjects

21 Factory Zone

21.1 Summary

This zone ‘produces’ units to defend the zone and attack other zones.

21.2 Dependencies

dcsCommon, cfxZones, cfxOwnedZones, commander, groundTroops

21.3 ME Integration

Name	Description
factory	Enables production abilities and attributes for this zone. The value is a string, coma separated, that tells the factory what types of troops to spawn. Can be overridden by other attributes like ‘*production*’, ‘*defenders*, ‘*productionXXX’* or ‘*defendersXXX’* Example: "`Soldier M4,Soldier M4`” produces two Infantry soldiers for defenders and attackers. **Warning:** these types need to *exactly* match DCS’s types. Be sure not to accidentally insert blanks. Special meta-type: “none” – no troops Defaults to “none” (no troops produced) Note that you must also supply a separate ‘owner’ attribute (which will also turn this zone into an owned zone); otherwise this factory belongs to the neutral faction and can’t be captured **MANDATORY** **Note:** In pre-2.x versions of ownedZones/factoryZones, the ‘factory’ production ability was integrated into ownedZones. If you are upgrading a mission with pre 2.0 version ownedZones to 2.x or later, you gain 99% backward-compatibility simply by adding this ‘factory’ attribute to the owned zone and leaving all other attributes as they are.
owner	**MANDATORY** each Factory must also be an owned zone, the owner attribute must be managed by the ownedZones module For value description, please see “ownedZones”
production	A string, coma separated, that tells the factory what types of troops to spawn. Can be overridden by other attributes like ‘*defenders*, ‘*productionXXX’* or ‘*defendersXXX’*

Name	Description
	Example: "`Soldier M4,Soldier M4`" produces two Infantry soldiers for attackers. **Warning:** these types need to *exactly* match DCS's types. Be sure not to accidentally insert blanks. Special meta-type: "none" – no troops Defaults to current value of '*factory*' attribute
defenders	A string, coma separated, that tells the factory what types of defensive ground units to spawn. Can be overridden by the side-specific attributes 'defendersRED' or 'defendersBLUE' Example: "`Soldier M4,Soldier M4`" places two red infantry soldiers to defend the factory. Defaults to current value of '*factory*' attribute
defendersRED	A string, coma separated, that specifies the types of defensive troops to spawn when the zone is owned by RED. Providing this attribute overrides the settings that you may have supplied in '*defenders'* or '*production'* Example: "`Soldier M4,Soldier M4`" places two red infantry soldiers to defend the factory when it is owned by the red faction. Defaults to current value of '*defenders'* attribute
defendersBLUE	A string, coma separated, that specifies the types of defensive troops to spawn when the zone is owned by BLUE. Providing this attribute overrides the settings that you may have supplied in '*defenders'* or '*production'* Example: "`Soldier M4,Soldier M4`" places two blue infantry soldiers to defend the factory when it is owned by the red faction. Defaults to current value of '*defenders'* attribute
productionRED	A string, coma separated, that specifies the types of offensive troops to spawn when the zone is owned by RED. Offensive troops seek out enemy owned zones. Providing this attribute overrides the settings that you may have supplied in '*production'* Example: "`Soldier M4,Soldier M4`" places two blue infantry soldiers to defend the factory when it is owned by the red faction. Defaults to current value of '*production'* attribute
productionBLUE	A string, coma separated, that specifies the types of offensive troops to spawn when the zone is owned by BLUE. Offensive troops seek out enemy owned zones. Providing this attribute overrides the settings that you may have supplied in '*production'* Example: "`Soldier M4,Soldier M4`" places two blue infantry soldiers to defend the factory when it is owned by the red faction. Defaults to current value of '*production'* attribute

Name	Description
redP!	Output to bang! whenever production cycles for “Production” units for the RED faction. Defaults to <none>
redD!	Output to bang! whenever production cycles for “Defender” units for the RED faction. Defaults to <none>
blueP!	Output to bang! whenever production cycles for “Production” units for the BLUE faction. Defaults to <none>
blueD!	Output to bang! whenever production cycles for “Defender” units for the BLUE faction. Defaults to <none>
formation	Formation of the defenders group. See dcsCommon for supported group formations. Defaults to ‘`circle_out`’.
attackFormation	Formation of the attackers group. See dcsCommon for supported group formations. Defaults to ‘`circle_out`’.
spawnRadius	Radius of circle that the defenders are placed on. Defaults to slightly less than zone radius, so defenders are always inside the zone they are defending. Defaults to 0.
attackRadius	Radius of circle in which the attackers spawn after they are produced. Defaults to zone radius
attackPhi	Angle (direction) in degrees from zone center where attackers are spawning. Defaults to 0.
paused	Pauses the zone’s production. “true” or “yes” means that the zone is paused. A paused zone produces no attackers nor defenders, but will detect capture normally. Capturing a paused zone will not unpause the zone. Since paused zones will signal their capture normally, so you can wire the “conquered!” output into “activate?” to automatically activate paused zones upon capture. Defaults to “no”
pause?	DML Watchflag to set paused status! Triggers on change Defaults to <none>
activate?	DML Watchflag to un-pause a paused owned zone. Defaults to <none>
helpMe?	DML Watchflag that triggers a ‘defend’ cycle (will produce new defenders). Defaults to <none>
triggerMethod factoryTriggerMethod	DML method to trigger inputs Defaults to “change”
method factoryMethod	DML method to bang! on outputs Defaults to ”inc”

21.4 Demos

- My first factory
- Clone Factory

22 FARP Zones

22.1 Summary

A zone linked to a FARP (and thus conquerable) that automatically provides service vehicles.

22.2 Dependencies

dcsCommon, cfxZones

22.3 ME Integration

Name	Description
FARP	Indicates that this zone is a FARP zone. Value is ignored. **MANDATORY**
rPhiHDef	Radius (in m), Phi (degrees) and Heading (degrees) of the center point around which the **defenders** deploy. Defaults to 0, 0, 0
rPhiHRes	Radius (in m), Phi (degrees) and Heading (degrees) of the center point around which the **resource** vehicles deploy as a line. Defaults to 0, 0, 0
redDefenders	typeStrings of defender vehicles. Example “ZSU-23-4 Shilka, ZSU-23-4 Shilka”. Defaults to “none” Special encoding: “none” – no vehicles
blueDefenders	typeStrings of defender vehicles. Example “Roland ADS, Roland Radar, Roland ADS”. Defaults to “none” Special encoding: “none” – no vehicles
formation	Formation of the defenders group. See dcsCommon for supported group formations. Defaults to ‘circle_out’.
rFormation	Radius of the circle that the defenders assemble in. Defaults to 100m
hidden	Set to “no” if FARP is visible on the F10 map (and colored according to owner). Defaults to “no”
hideRed hideBlue hideGrey	For any of these three attributes, the FARP is hidden if it belongs to that faction. For example, if hideRed is set to true, the FARP is shown on the map while it belongs to neutral or blue, but disappears when it is owned by red.

22.4 Demos

- FARP and away

23 FireFX

23.1 Summary

This module is similar to ‘smoke zone’, except that it creates a fire/black smoke effect that can be turned off much quicker and that can be controlled in size and visual intensity.

23.2 Dependencies

fireFx requires dcsCommon, cfxZones.

23.3 ME Integration

Name	Description
fireFX	Tells DML that you want fire/smoke effects inside the zone. The value of this field specifies the size of the effect itself. Currently DCS recognizes the following: • “small” or “S” • “medium” or “M” • “large” or “L” • “huge” or “H” or “XL” • “rnd” for random size Smoke is always colored black. Flames are optional (see the ‘flames’ attribute) Note that *this effect is visual only*. Vehicles or troops inside the fire aren’t damaged by the fire at all. Defaults to “small” (if none of the given values is recognized) **MANDATORY**
agl	Altitude **in meters** above the ground where the effect should start. Often used with oil rigs or chimneys to create torch/smoke effects at their top. Defaults to 0 (directly on the ground)
flames	If you supply “false” for this attribute, only smoke (no fire) is displayed. **Using “rnd” as value is fully supported and randomized for each and every effect.** Defaults to true (flames and smoke)
density	“Thickness” or visibility” of the smoke produced by this effect.
start?	DML watchflag (input) for when the effect should start. Defaults to <none>
stop?	DML watchflag (input) for then the effect should stop

Name	Description
	Note: unlike the smoke zone module, which can take several minutes for the smoke to stop, this fireFX's smoke and flames take roughly 10 seconds to peter out) Defaults to <none>
onStart	If set to true, the mission starts with the effect on Defaults to false (no effect on start)
triggerMethod fxTriggerMethod	DML method for inputs Defaults to change
num	The number (e.g. "3") or range (e.g. "2-7") of fires to start. If you supply a range, each time that you start fires in the zone, a random number from lower to upper range (inclusive) of fires is started. Defaults to 1 (one fire)
rndLoc	If true, the location(s) of the fire(s) is randomized. If you are allowing more than 1 fire to be started, this value is set to true. Defaults to pause (fire starts at center)

23.4 Demos

24 Flare Zone

24.1 Summary

Launches flares from a location on the map on command. Supports heavy randomization.

24.2 Dependencies

fireFx requires dcsCommon, cfxZones.

24.3 ME Integration

Name	Description
flare	Tells DML that this trigger zone is a flare zone. The value of this attribute specifies the flare's color. Valid values are "green", "red", "white", "yellow", "random", "rnd", or "-1" (for random), "0" (for green), "1" (for red), "2" (for white) or "3" (for yellow) Defaults to "green" **MANDATORY**
launch? launchFlare? f?	Watchflag. When the value of this flag changes, the flares are launched. If no flag is defined, no flares can be launched. Defaults to <none>
triggerMethod flareTriggerMethod	DML Method by which the flag is triggered. Defaults to "change"
direction	Compass heading in degrees (or a range) in which to launch the flare. The heading must be a positive number. When you supply a range, the direction will be randomized between the lower and upper bounds. Note that both numbers in the range must be positive numbers, and the upper bound must be larger than the lower. So if you want the flares to launch from 270 to 90 degrees, you need to add 260 to the second number. Examples: • 45 (launch in NE direction) • 90-180 (launch somewhere between East and South) • 270-450 (launch between West and East) Defaults to 0 (North)
altitude flareAlt agl	Height (in meters) above ground where the flare is launched from. Defaults to 1 (m above ground)
salvo	Number of flares to launch when the zone is triggered. Can be range. If you supply a range, a random number is chosen inside the range. Examples:

Name	Description
	• 3 (launch 3 flares • 2-5 (launch from 2 to 5 flares) Defaults to 1
duration	Only applicable when salvo contains more than 1 flares. The time (in seconds) over which the salvo is to be launched. If, for example salvo size is 4 flares, and duration is 2 seconds, a flare will be launched every 2/4 = 0.5 seconds, so that all 4 flares are launched of the span of 2 seconds. Duration can be a range, in which case the duration is randomized to a value inside the range. Defaults to 1 (second)

24.4 Demos

- Effects with a flare

25 Fogger

25.1 Summary

A module to give mission designers free control over ground fog

25.2 Dependencies

Fogger requires dcsCommon and cfxZones.

25.3 ME Integration

Name	Description
fog?	This input triggers transition to the current visibility to the fog settings defined below. **MANDATORY**
triggerMethod	DML Method for the input Defaults to “change”
visibility	Number of meters for visibility. The lower the number, the worse visibility. If set to a value below 100, this turns off fog- Value can be a range (e.g. 1000-2000). When specified as a range, a random value between the two values is generated each time that fog? is triggered. Defaults to 0 (no fog)
thickness	Thickness of the fog, as defined by DCS. It is currently insufficiently specified by ED if this means MSL or AGL. Since fog is a global map setting, it seems prudent to assume that the value for thickness is meters above medial sea level. When this value is less than 100, fig is turned off Value can be a range (e.g. 200-600). When specified as a range, a random value between the two values is generated each time that fog? is triggered. Defaults to 0 (no fog)
lcl	When set to true, the altitude at the zone’s center is added to the fog’s thickness, allowing you to create ‘local’ thickness. If, for example, you set thickness to 100, and the zone is positioned at 220 MSL, the fog’s thickness is set to 220 + 100 = 320. This is useful for maps that have many airfields in higher regions (e.g. NTTR) **Note:** currently in DCS, **fog is** a **global** setting, and there is no true local fog. The setting from a fog zone applies to the entire map. Defaults to false (non-local)
duration	Time in seconds that it takes from the moment that the new fog settings are triggered until the fog is fully established. DCS will

Name	Description
	transition to the new settings dynamically over time, so you can use this feature to have fog ‘creep in’ over time Like thickness and visibility, duration supports a range to randomize the duration. Defaults to 1 second

25.4 Demos

- Foggy bottoms

26 Ground Explosion

26.1 Summary

Module to set off explosions on (and above) the map. Supports multiple explosions, randomized locations, time control.

26.2 Dependencies

GroundExplosion requires dcsCommon and cfxZones.

26.3 ME Integration

Name	Description
explosion	Marks this zone as an explosion zone. All explosions triggered will occur within the zone (although their damage/blast may well extend past the zone. The value of this attribute determines the strength of the explosion. A value of 1 may be harmful to individual units, while a value of 3000 is almost sure to level a building (and others close by, depending on their strength) Supports ranges, so if you specify a strength of 50-230, each individual explosion will receive its own randomized strength within that range Defaults to 1 **MANDATORY**
triggerMethod	DML trigger method for inputs. Defaults to “change”
boom?	DML watchflag. When triggered, the module runs through an explosion cycle, setting of as many explosions as specified by the attributes Defaults to <none>
num	Number of explosions to perform every time that the input is triggered. Supports ranges, so setting *num* to “3-9” will cause groundExplosion to set off a random number between 3 and 9 (inclusive) each time that the input is triggered. Defaults to 1 (a single explosion)
rndLoc	Controls where the explosion will be located. If set to false (default) the explosion will happen at the exact center of the trigger zone.

Name	Description
	If *rndLoc* is set to true (or *num* is greater than 1), the positions of all explosions are always randomized to occur within the area of the trigger zone. **NOTE:** If *num* is set to a value greater than one (1), the location of each explosion is always randomized inside the zone. Defaults to false (if a single explosion is chosen, it occurs at center of zone)
agl	Height (in meters) above local ground where the explosion occurs. This allows you to create air bursts, and even Flak effects. **Important:** if your target zone spans large areas with steep inclines, this can result in explosions with great variance in height, as the explosion always occurs *agl* meters above the local ground. Can be set to a random range (e.g. 20-30) that is applied to each explosion individually Defaults to 1 (m above ground)
duration	Only applicable if set to multiple explosions. Sets the time frame (in seconds) for all explosions to occur within, after being triggered. So if you set *num* to 8 and *duration* to 2, all 8 explosions occur within a span of 2 seconds. Defaults to 0 (all explosions happen immediately at time of triggering)
flares	Adds flares that erupt from the explosion. The value that you supply controls the number of flares that are launched with each explosion, and supports ranges (e.g. “1-3”). Note that flares are always launched from the ground, so this effect may look silly when used with *agl*. Defaults to <none> - no flares

26.4 Demos

- Boom boom

27 Group Tracker

27.1 Summary

A module that counts the groups in a set, and changes flags when the number changes. Extremely versatile count based on the tracked groups' survival.

27.2 Dependencies

dcsCommon, cfxZones

27.3 ME Integration

To add all groups that have at least one unit inside the zone to a groupTracker, add the following attribute to the zone:

Name	Description
addToTracker:	List of groupTracker zones. All groups that have at least one unit inside this zone are added to these groupTrackers. This happens only at mission start-up, and therefore only work for non-player-controlled planes (since player-controlled planes do not exist at mission start-up). If your player group contains AI planes, place one of those into the zone, and that group can be added to a tracker. If you have stacked the tracker on the same zone, you can use a single asterisk '*' as zone name. Supports a comma-separated list of trackers if you simultaneously want to pass the groups to multiple trackers, e.g. "GroundTrack, HeloTrack" This is useful if the zone contains more than one group, and your trackers use filtering Add all groups that have at least one unit in this zone to the tracker whose zone name is given in the Value field.

To add a groupTracker to a zone

Name	Description
tracker	Marks this zone as a groupTracker. It can be referenced by the zone's name passed in the trackWith: and addToTracker: attributes. When referenced locally, a single asterisk "*" can be used as wildcard name for easy copy/paste of the entire stack **MANDATORY**
addGroup!	Whenever a group is added to the tracker, the value of this flag is increased. If not changed by other modules, this flag also doubles as a running total of all groups added to the tracker Defaults to <none>
removeGroup!	Whenever a tracked group is destroyed, the value of this flag is increased. If not changed by other modules, this flag also doubles as a running total of all watched groups that have been destroyed while they were tracked

Name	Description
	Defaults to <none>
numGroups!	The value of this flag always represents the number of groups currently watched by this tracker. **This value is updated 1/ups times per second.** Defaults to <none>
numUnits	The value of this flag always represents the total number of units currently watched by this tracker. **This value is updated 1/ups times per second**. Defaults to <none>
groupFilter	Which unit categories to track. If no attribute is given, **all** categories are tracked. When you supply a groupFilter attribute**, only that category is accepted** when attempting to add to a tracker. Currently supported are • 0 (zero) or “aircraft” or “air” • 1 or “helo” or “heli” or “helicopter” • 2 or “ground” • 3 or “ship” • 4 or “train” Defaults to no filtering
triggerMethod trackerTriggerMethod	Watchflag method for inputs Defaults to ‘change’
destroy?	Watchflag that when triggered destroys all groups that are currently being watched. If any groups are destroyed, the removeGroup output is increased by the number of groups that were removed. numGroup is set to 0 Defaults to <none>
method trackerMethod	Method to bang! on output flags Defaults to “inc”
allGone!	Flag to bang! when the number of tracked groups falls to zero. If it was zero before, no output signal is generated. Defaults to <none>

27.4 Demos

- Track This!
- Moving Spawners II
- Impossible Impostors

28 Guardian Angel

28.1 Summary

Provides out-of-the-box protection for aircraft against guided missiles.

28.2 Dependencies

Guardian Angel requires dcsCommon and cfxZones

28.3 ME Integration

Guardian Angel uses a config zone “guardianAngelConfig“ to control all settings

Name	Description
verbose	A value of “true” turns on debugging messages. Default is “false”
autoAddPlayer	When set to true, player planes are automatically added to Guardian Angel’s watchlist. Default is true
launchWarning	If true, Guardian Angel announces a missile launch. Default is true
intervention	If true, Guardian Angel destroys a missile before it destroys a watched aircraft. Default is true
announcer	If set to false, Guardian Angel suppresses all announcements. Defaults to true
msgTime	Number of seconds that a warning from Guardian Angel stays on-screen. Defaults to 30 seconds
private	If set to true, all announcements are only made to the group that a missile was fired at. Set to false (everyone can see)
launchSound	Name of the sound file to be played when a missile is launched. Respects ‘private’ attribute
interventionSound	Name of the sound file to be played when Guadian Angel saves an aircraft. Respects the ‘private’ attribute.
explosion	Guardian Angel can add a mostly harmless explosion when a missile is removed due to an intervention. If this value is smaller than one (e.g., -1) this feature is turned off. If you enter a value > 0 (zero), an explosion with a magnitude of this value is placed in direction of that missile’s last location, 500m from the aircraft. A mostly harmless value is 1.0 (one point zero) **WARNING I** Even though this explosion is usually harmless for the protected plane, it can pose lethal to any other plane (wingmen). **WARNING II**

Name	Description
	The explosive effect is only harmless to the protected plane if the explosion value is small (e.g., 1). If you enter sufficiently larger values, the shock wave can destroy even the protected plane. If you set this value to see explosions, make the value 1.0 Defaults to -1 (off)
fxDistance	When using explosions, this is the distance (in meters) away from the aircraft where the (real) explosion is going to take place. Defaults to 500
active	The state that Guardian Angel starts up in. True means that it is active, false that it is turned off. Defaults to 'false'
activate? on?	Watchflag to turn on (activate) guardian angel. Defaults to <none>
deactivate? off?	Watchflag to turn off (deactivate) guardian angel. Defaults to <none>

You can selectively add and remove aircraft from protection by Guardian Angel by placing them in a zone with the "guardian" attribute:

Name	Description
guardian	**MANDATORY** Tells Guardian Angel how to treat aircraft inside the trigger zone: • *true* All aircraft inside this zone are protected by Guardian Angel • *false* All aircraft inside this zone will not receive protection from Guardian Angel. Defaults to 'true' (all aircraft inside are protected)

28.4 Demos

- Missile Evasion (Guardian Angel)
- Guardian Angel Reloaded

29 Helo Troops

29.1 Summary

This module provides instant out-of-the box ability for troop transport helicopters to pick up and deploy infantry

29.2 Dependencies

Required: dcsCommon, cfxZones, commander, groundTroops

Optional: cfxSpawnZones (for requestable troop spawning)

29.3 ME Integration

HeloTroops supports dropZones to restrict where units can be disembarked and/or where disembarking troops from players generate signals:

Name	Description
dropZone!	Tells DML that this zone is a zone with special relevance for HeloTroops. The value of this lists the flags (commands) to signal if a player-controlled helicopter disembarks troops inside this zone. **MANDATORY**
dropMethod	DML method used to send the signal over outputs Defaults to “inc”
coalition	Colation that can drop here. If set to neutral, both coalition (red and blue) can drop here Defaults to 0 (neutral) – both coalitions can disembark their troops here
autoDespawn	Time interval in seconds after which disembarked troops despawn. Used primarily to clear the drop zone in extraction-based missions that would overcrowd otherwise. Set to a value less than 1 to turn off Defaults to -1 (no auto-despawn)

You mainly control Helo Troops via the configuration zone. All attributes you enter here are global to the module.

To configure the HeloTroops module via a configuration zone,

- Place a Trigger Zone anywhere using ME
- Name it “heloTroopsConfig” (note: name must match exactly)
- Add any of the following attributes to this zone:

Name	Description
legalTroops	Type list (comma separated) that identifies the unit types that helicopters can load. This is compared against any unit on the ground to determine if the helicopter can load the group. All units in the group must be on that list, or the entire group cannot be loaded. For example, if a group consists of four infantry soldiers, the group can be loaded. If the group also contains a vehicle (e.g. “Hummer”), that group cannot be loaded. Defaults to "`Soldier AK, Infantry AK, Infantry AK ver2, Infantry AK ver3, Infantry AK Ins, Soldier M249, Soldier M4 GRG, Soldier M4, Soldier RPG, Paratrooper AKS-74, Paratrooper RPG-16, Stinger comm dsr, Stinger comm, Soldier stinger, SA-18 Igla-S comm, SA-18 Igla-S manpad, Igla manpad INS, SA-18 Igla comm, SA-18 Igla manpad`"
troopCarriers	A list of helicopter types that are allowed to carry troops in this mission. Defaults to DCS Common’s list of troops (which is usually `Mi-8MT, UH-1H, Mi-24P, OH58D, CH-47Fbl1`), but you can provide your own list (for example to add non-official types). Example: “`Mi-8MT, UH-1H, SA342Minigun, C-130J`” removes the Hind and adds the Gazelle in Minigun configuration as well as the Hercules fixed-wing transport to the list of legal troop carriers. Supports wildcard type endings: if a type ends on an asterisk (“*”) all types that match whatever precedes the asterisk are accepted. For example, “`Mi-*`” will match both “`Mi-8T`” and “`Mi-24P`”. You can supply the type ‘`any`’ or ‘`all`’ to allow all player aircraft to carry troops. You can supply the type “`helo`” or “`helos`” to allow all helicopters to carry troops Default <none> (use dcsCommon’s list of troop carriers)
troopWeight	Used to calculate the cargo weight per troop loaded. Currently not used. Defaults to 100 (kg)
autoDrop	Default setting for helicopter when touching down. Players can change this individually. Defaults to true
autoPickup	Default setting for helicopter when touching down. Players can change this individually. Defaults to false
pickupRang	Range in which troops can be picked up, from a helicopter. Defaults to 100 meters
combatDropScore	Score to award when a player unloads troops in a non-aligned owned zone. Requires OwnedZones and PlayerScore to be active. Defaults to 200
actionSound	Sound file to play to the helicopter group whenever troops are loaded or unloaded
loadSound	Name of the sound file to play to the helicopter group whenever troops are loaded into the helicopter. Overrides ‘actionSound’ Defaults to the value set by actionSound (above).
disembarkSound	Name of the sound file to play to the helicopter group whenever troops are unloaded from the helicopter. Override ‘actionSound’.

	Derfaults to the value set by ‘actionSound’ (above).
requestRange	Distance (in meters) that a player helicopter must land within for a spawner or cloner with ‘requestable’ attribute to be eligible to request troops from. Note that this distance is different (usually greater) that the pick-up range (see above), so helicopters can order a spawner/cloner to produce fresh troops, but may not be close enough to pick them up (i.e., the player must move the closer to pick them up). Defaults to 500 (meters)
enforceDropZones	When set to true, a player can only disembark troops when they land inside a *dropZone,* Defaults to false (players can disembark their troops anywhere)

29.4 Demos

- **Helo Trooper**
- CSAR of Georgia
- Send in the Clones
- Inferno at Sea

30 Impostors

30.1 Summary

Impostors allow you to switch AI-controlled units to their static object eqivalents (“impostor”). Impostors will not react to enemies, and consume less CPU. You can use flags to control if a group is in impostor or controlled state.

30.2 Dependencies

Impostors require dcsCommon, cfxMX and cfxZones.

30.3 ME Integration

Name	Description
impostor?	Marks all groups that have at least one unit inside this zone as potential impostor, giving them the ability to change between static object and AI-controlled unit. The value of this attribute is a Watchflag that triggers transition of all surviving units from AI-controlled to static object. **MANDATORY**
reanimate?	Watchflag that triggers transition of all surviving static units to AI-controlled units. They immediately restart any waypoint actions or route orders (if given). Ground units will start moving their first (not initial) waypoint.
triggerMethod impostorTriggerMethod	Method that triggers inputs (DML Watchflags) Defaults to “change”
onStart	If the value of this attribute is true, all units are turned into imposters at the start of the mission. Default is false
blink	The value of this attribute specifies the brief interval (in seconds) between removing the impostor and spawning of the AI-controlled units. Only required for AI aircraft. A good value is 0.1 – 0.2; a value of zero or negative value means no blinking. Deafults to -1 (no blink interval)
trackWith:	Name of a zone with a groupTracker attached. All units are added to that tracker when they are AI-controlled, and removed when they are turned into impostors. Reanimating them subsequently will again add them to the tracker etc.
allDead!	DML flag to bang! when all groups that are managed by this impostor zone have been destroyed
method impostorMethod	DML method for output Defaults to “inc”

30.4 Demos

- Impossible Impostors

31 Inferno

31.1 Summary

Module that simulates (spreading) fires and that can be extinguished with aircraft using the airtank module

31.2 Dependencies

Inferno requires dcsCommon and cfxZones

31.3 ME Integration

Name	Description
inferno	Tells DML that this zone is an inferno zone. The value of this attribute is ignored. **Mandatory**
cellSize	Size (in meters) per side for the fire cells inside the zone. All fire cells are square. This value overrides the value given in the config zone. Defaults to the config value for cellSize.
fuel	Amount of fuel for each fire cell inside the zone. Once a cell ignites, it spends fuel while burning (except when *eternal* is enabled, see below). When all fuel is spent, the fire cell extinguishes and can't re-ignite until the entire inferno zone resets Defaults to 100
rndLoc	When set to true, the location of the first fire cell to ignite is randomized. This can randomize shape and size of the resulting conflagration. Defaults to true (randomized fire starting cell)
freeBorder	When set to true, the fire cells inside the zone are surrounded by a 'ring' of 'dud' (un-ignitable). Use this to visually better align an inferno with geologic peculiarities of the map. Defaults to true (add an inside border)
eternal	If set to true, fires inside a fire cell do not consume fuel until they receive a signal from an outside source (usually the *airtank* module), making all cells, once ignited, eternal. Defaults to true (eternal burn)
stagger	When set to true, the center of the flame effect is randomized inside the fire cell, making the entire inferno look more natural. Defaults to true (randomized FX location)
canSpread	When set to true, a fire cell can ignite neighboring fire cells.

Name	Description
	Defaults to true (fire can spread to other cells)
maxSpread	The maximum amount of fire cells that a fire can spread to inside this zone. Defaults to infinite
onStart	If set to true, the inferno in this zone is started when the mission starts up Defaults to false (inferno waits for ignite? input)
ignite?	Input to tell this zone to start a fire by igniting a single fire cell inside. If the zone has already started a fire, this signal is ignored. Defaults to <none>
douse?	Input to tell the zone to extinguish all actively burning fire cells inside this zone, and reset it to re-start via *ignite?* Defaults to <none>
extinguished!	Output that bangs on the connected flag when all burning fire cells have been extinguished and the inferno cell is no longer burning. Defaults to <none>

31.4 Demos

32 Limited Airframes

32.1 Summary

Set a maximum per side on the number of player pilots that can be lost.

32.2 Dependencies

dcsCommon, cfxZones, cfxPlayer

32.3 ME Integration

Most features of Limited Airframes are controlled via config zones:

Name	Description
verbose	A value of “true” turns on debugging messages. Default is “false”
enabled	Controls whether or not Limite Airfames is in effect. Defaults to “true”
userCanToggle	Controls whether players can turn Limited Airframes on and off during the mission. Defaults to “true”
maxRed	Maximum (and starting) number of pilots for the red coalition. Set to -1 to make the number unlimited. Defaults to -1 (unlimited)
maxBlue	Maximum (and starting) number of pilots for the red coalition. Set to -1 to make the number unlimited. Defaults to -1 (unlimited)
red# #~~red~~	Flag that continuously is set to the current number of pilots remaining for Red
blue# #~~blue~~	Flag that continuously is set to the current number of pilots remaining for blue
redWins! redWinsFlag!	Flag to bang! when blue has lost all pilots and red wins. Defaults to <none>
blueWins! blueWinsFlag!	Flag to bang! when red has lost all pilots and blue wins. Defaults to <none>
method	DML bang! method. Defaults to ‘inc’
warningSound	Name of sound file to play when limited airframes is displaying a message. Defaults to <none>
winSound	Name of sound file to play for winning side when other side has lost all pilots. Defaults to <none>
loseSound	Name of sound file to play for the side that has lost all pilots and therefore lost the engagement. Defaults to <none>
announcer	When set to false, the are no announcements for change on air frames, Defaults to true (airframe changes are announced)

On the map, you should place safe zones (one per side that has limits on player pilots) using the following attributes

Name	Description
pilotsafe	Marks this zone as safe for pilots to change into other airframes when landed. • If the value to this attribute contains neither the word ‘red’ nor ‘blue’, it is safe for all coalitions

Name	Description
	• If the value of this attribute contains the word ‘red’ this zone is safe for the red coalition • If the value of this attribute contains the word ‘blue’ this zone is safe for the blue coalition Note that safe state may be contingent on ownership of the zone. **MANDATORY**

32.4 Demos

- Pilots at their limit

33 LZ

33.1 Summary

A module that can create trigger events for landings and take-offs inside the zone

33.2 Dependencies

dcsCommon, cfxZones

33.3 ME Integration

Name	Description
LZ	Tells DML that this is an LZ that detects landing and take-off of aircraft inside it. The value of this attribute is ignored **MANDATORY**
landed!	Flags to bang! when a plane matching the criteria lands inside the zone Defaults to <none>
departed!	Flags to bang! when a plane matching the criteria lands inside the zone Defaults to <none>
coalition	When given, the coalition that aircraft must match. Supported are “red”, 1, “blue”, 2, “neutral” 0 When you specify “neutral” or 0, all coalitions match Defaults to 0 (any coalition)
playerOnly	When set to true, all AI planes are ignored Defaults to false (all players and AI are considered)
unit units	A comma-separated list of all unit names (case **in**sensitive) that should be considered. Supports wildcard “*” for the last character, in which case all groups that match that everything before the asterisk are considered: “Hog-*” will match “HOG-”, “hog-1-1” and “HoG-5-2”, but not “Hogger” When matching units, any ‘coalition’ setting is ignored Defaults to <none>
group groups	A comma-separated list of all group names (case **in**sensitive) that should be considered. Supports wildcard “*” for the last character, in which case all groups that match that everything before the asterisk are considered: “He*” will match “He”, “HELLO” and “heinkel-1-1”, but not “Hans Heinkel” Defaults to <none>
type types	List of types, separated by comma, that a unit must match at least one of (e.g. “A-10A”). Types must match exactly Additionally, the following special types are also supported • “ALL” or “ANY” – all aircraft match • “HELO” – all helicopters match • “PLANE” – all fixed-wing planes match

Name	Description
	Defaults to “ALL”
isPaused	When set to true, the LZ is paused at mission start Defaults to false (LZ is active on start)
pause?	DML Watchflag to pause the LZ Defaults to <none>
continue?	DML Watchflag to continue a paused LZ Defaults to <none>
method outputMethod	Method to the outputs Defaults to ‘inc’
triggerMethod lzTriggerMethod	Method that triggers inputs. Defaults to “change”

33.4 Demos

- Departures and Landings

34 Map Markers

34.1 Summary

Allows you to place markers on the F10 map.

34.2 Dependencies

dcsCommon, cfxZones

34.3 ME Integration

Name	Description
mapMarker	Turns on the map marking feature. Simply must be present. Content of this property is displayed as text on the Map. Example “Destroy all vehicles in this area” **MANDATORY**
coalition	Side that sees this marker. Can be “red”, “blue”, “neutral”, or “all”. You can also substitute “1” for red, and “2” for blue. Defaults to “all”

34.4 Demos

35 Messenger

35.1 Summary

A module that generates a text and/or audio output.

35.2 Dependencies

dcsCommon, cfxZones

35.3 ME Integration

Name	Description
messenger?	Watchflag. When triggered, the module will display the message and/or play sound. **MANDATORY**
message	The text of the message to be displayed **FORMATTING WILDCARDS** • <n> creates a new line • <z> is replaced with zone's name • <t> is current time in the format as defined with the *timeFormat* attribute **RESPONSE-SELECTION WILDCARDS** • <rsp: flag name> looks up the value of <flag name> and uses that value and an offset into the responses given with the 'responses' attribute. The first response has an index of 1. If the flag's value is less than 1, the first response is returned, if the value is higher than the number of responses, the last response is returned. • <rrnd> randomly selects one of the possible responses and returns that response • <rhdg: unit/zone> wraps responses around the compass, and then uses the unit/zone's heading as offset. Zones only have a heading if they are linked to a unit. [not yet implemented] • <rbea: unit/zone> selects and returns one of possible responses using the bearing (in degrees) from unit/group to unit/zone as offset into *responses.* [requires group or unit attribute be set] **DATA ACCESS WILDCARDS** • <lat> the latitude of the zone's current position • <lon> the longitude of the zone's current position • <mgrs> the zone's current position in MGRS coordinates • <v: flagName> is replaced with the value currently held by the flag *flagName* • <t: flag name> uses the value from flag *flag name* and interprets it as a time value, formatted according to the timeFormat attribute

	• <lat: unit/zone name> outputs the latitude of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found • <lon: unit/zone> outputs the longitude of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found • <ele: flagName> outputs the elevation of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found. Elevation is calculated in meters (default) or feet if the “imperial” attrigute is true • <latlon: unit/zone> outputs the latitude and longitude of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found • <lle: unit/zone> outputs the longitude, longitude and elevation of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found • <mgrs.: flagName> outputs the mgrs. coordinates of the zone or unit that matches the name *unit/zone name*, or “messageError” (usually an empty string) if neither can be found • <vel : unit/zone> outputs the velocity (in km/h or knots, depending on imperialUnits) of unit. Zones only have a velocity if they are linked to a master unit • <hdg: unit/zone> outputs the direction that the unit/zone referenced is heading. Zones only have a heading if it is linked to a master unit, and the heading then returned is the one of the linked unit. • <alt: unit/zone> outputs the altitude (barometric) of the unit/zone. If the zone is unlinked, it returns the altitude of the land at the zone’s center. If the zone is linked to a unit, it returns the altitude of that unit. Altitude is returned in meters or feet, depending on the imperialUnits attribute. • <type: unit/zone> returns the type of the unit (e.g., “A-10C”). For zones, the type returned is always “Zone” • <player: unit> outputs the player’s log-in name if that unit is controlled by a player (e.g., “New Callsign”), if the unit isn’t controlled by a planer it returns “Unknown” • <coa: flag/unit/zone> outputs the faction (e.g., “RED”) of the flag/unit/zone. It returns the first name match it finds: Unit before zone before flag. • <twn: unit/zone> outputs the name of the nearest named location on the map. Requires the twn module • <loc: unit/zone> outputs position relative to nearest named map location. Requires twn module **“HERE/THERE” RELATIVE (requires group or unit)** These wildcards are only available when you have supplied messenger with information about “here” – with a *group*, *unit* or *linkedUnit* attribute. “There” is the unit that is referenced in the wildcard. • <bea: units/zone> bearing (in degrees) from “here” to “there”

	• <rng: unit/zone> range (distance) from “here” to “there” • <clk: unit/zone> direction as “o’clock” to the unit/zone as seen from the player unit/zone’s heading (12 is straight ahead, 6 is behind) • <hnd: unit/zone> direction “which hand” to the unit/zone: “ahead”, “right”, behind”, “left” as seen from “here” • <sde: unit/zone> direction “side” to the unit/zone as seen from “here”: “ahead”, “starboard”, “aft”, “port” • <asp: unit/zone> aspect of “there” towards “here”. Returns “hot” / “beam” / “drag” • <cls: unit/zone> closing velocity of “here” and “there”. A negative closing velocity means that distance is growing. Closing velocity is given in km/h or knots, depending on imperialUnits • <pcls: unit/zone> precision closing velocity of “here” with “there”. Closing velocity is given in m/s or ft/s with up to one decimal, e.g., “1.3” **Interpreted values** Return one of two possible values, based on the flag’s value or existence of the unit named • <yes: flagOrUnitName> returns “no” if flag’s value is zero and no unit of that name exists, “yes” otherwise • <true: flagOrUnitName> returns “false” if the flag’s value is zero and no such unit exists, “true” otherwise • <in: flagOrUnitName> returns “out” if the flag’s value is zero and no such unit exists, “in” otherwise • <alive: flagOrUnitName> returns “dead” if the flag’s value is zero and no such unit exists, “alive” otherwise • <A/B: flagOrUnitName [zero val | other val]> returns whatever is inside the square brackets “[]” and left of the vertical bar “|” (“zero val” in this case) when the flag’s value is zero and no such unit exists, and whatever is to the right of the vertical bar “|” inside the square brackets “[]” otherwise
responses	A list of comma-separated possible responses that can be accessed by various wildcards. Note that the responses themselves must not contain a comma “,”. Example: “good, better, the best” is treated as three separate possible responses: “good”, “better” and “the best”. Defaults to <none>
triggerMethod msgTriggerMethod	Defines the trigger condition for DML Watchflags. Use only one synonym per zone Defaults to “change”
clearScreen	If true, erase all existing messages. Defaults to false
soundFile	Name of the sound file (including extension like ‘.wav’) that is to be played. Defaults to ‘<none>. Note that the sound file’s name must be specified relative to the mission’s default location for sound files (l10n/DEFAULT/). If you use ME to import the sound files, you do not have to specify the location.

	Remember to import the sound file into the mission else no sound will play.
coalition msgCoalition	The coalition that should receive the message/sound. If no coalition is given, text and sound are played to all. Legal values are “red”, “blue”, “neutral”, 0, 1, 2 Note that if given, the attributes ‘coalition’, ‘group’ and ‘unit’ are mutually exclusive. Defaults to <none>
group msgGroup	The name of the Groups (separated by comma ‘,’) that should receive the message/sound. Adding a group attribute enables unit-relative wildcards. Note that if given, the attributes ‘coalition’, ‘group’ and ‘unit’ are mutually exclusive. Defaults to <none>
unit msgUnit	The name of the Units (separated by comma ‘,’) that should receive the message/sound. Adding a group attribute enables unit-relative wildcards. Note that if given, the attributes ‘coalition’, ‘group’ and ‘unit’ are mutually exclusive. Defaults to <none>
messageOn?	When the value of this flag changes, the messenger is turned on. If it already was on, nothing happens All messengers start in On state and require at least one signal on their messageOff input to disable. Defaults to <none>
messageOff?	When the value of this flag changes, the messenger is turned off. Any further messages are suppressed. If the messenger was already turned off, nothing happens. Defaults to <none>
mute messageMute	If set to true, the messenger starts muted and requires a signal on messageOn? To activate. Defaults to false
duration messageDuration	Time (in seconds) how long a message should stay on-screen Defaults to 30 seconds
timeFormat	Time format for any time values Defaults to “<:h>:<:m>:<:s>” – standard 24 hour time
imperial imperialUnits	When true, elevation is calculated in feet (imperial units) else meters. Defaults to false (meters)
error messageError	The text to substitute for a unit or zone reference if the units or zone cannot be found. Defaults to “” (empty string)

35.4 Demos

- Bottled Messages
- The Zonal Countdown
- Frog Men Training
- Follow Me!
- CSAR of Georgia
- Track This!
- Watchflag Demo
- Radio Go-Go
- xFlags Field Day

- Count Base’s Blues
- Gate and Switch
- Moving Spawners II
- Reinforcements A La Carte
- **Formation Trainer**

36 (Simple) Mission Restart

36.1 Summary

Restarts the mission when the flag named ‘simpleMissionRestart’ changes to any value other than 0 (zero)

36.2 Dependencies

Mission must be run in multiplayer.

No other dependencies

36.3 ME Integration

n/a

36.4 Demos

(none)

37 NDB

37.1 Summary

Allows you to place an NDB in any zone. This includes linked zones so you can place NDBs that follow ships.

37.2 Dependencies

dcsCommon, cfxZones.

37.3 ME Integration

Name	Description
NDB	Creates an NDB at the zone's center. If the zone is linked to a unit, this NDB will automatically update to the unit's location. **T**he value of this attribute is the frequency (**in MHz**) at which the NDB transmits (e.g. 121.5 for 121.5 MHz, 0.42 for 420 kHz) The NDB starts transmitting at mission start unless there is also a paused=true attribute present (see below) **MANDATORY**
fm	If true, the transmission is in FM, else in AM Defaults to false (AM)
soundFile	Name of the sound file with extension that is to be transmitted. Defaults to '<none>. Note that the sound file's name must be specified relative to the missions default location for sound files (l10n/DEFAULT/). If you use ME to import the sound files, you do not have to specify the location. Remember to import the sound file into the mission else no sound will play.
watts	Transmission power (in watts) for the NDB. 100 Watts usually has a range of some 150 km. Defaults to 100 Watts
paused	If set to true, on mission start the NDB will not start up. Use the "on?" watch flag attribute or API to turn it on. Defaults to false
on?	Watchflag. **Each time the flag triggers, the NDB is started (will also cause the transmission sound to rewind).** The current paused value is ignored, and then set to false after the NDB has started. Defaults to no flag to watch
off?	Watchflag. **Each time the flag triggers, the NDB is stopped.** paused value is set to true after the NDB has stopped. Defaults to no flag to watch
triggerMethod ndbTriggerMethod	Defines the trigger condition for DML Watchflags. Use only one synonym per zone Defaults to "change"

37.4 Demos

- ADF and NDB Fun

38 NoGap / NoGapGUI

38.1 Summary

Fill empty player ground slots with static aircraft of the same type and livery until players slot into them. Makes airfields look much more alive. This is ‘stopGaps’ sibling that better resolves multi-unit player groups, but does not work with many popular scripts like SSB (simple slot block)

For multiplayer servers, install noGapGUI to avoid synchronization issues (server only)

38.2 Dependencies

dcsCommon, cfxZones, cfxMX

For Multiplayer, you must install the noGapGUI server extension on the server.

38.3 ME Integration

To *exclude* aircraft from noGap, use trigger zones and the following attributes:

Name	Description
noGap	When the value of this attribute is **false**, any player aircraft inside this zone will not receive a static stand-in, their slot remains empty. Defaults to true (unit receives a static stand-in) **MANDATORY**

Alternatively, you can disable noGap when the mission is run in single-player mode (this is useful when, due to some map properties, the aircraft ‘falls out of the sky’ in single-player, but works well in multiplayer due to noGapGUI’s superior synching.

Name	Description
noGapSP	When the value of this attribute is **false**, any player aircraft inside this zone will not receive a static stand-in **in single-player mode**, their slot remains empty. In multiplayer mode, the slot is filled with a static stand-in. Requires noGapGui on the server. Defaults to true (unit receives a static stand-in) **MANDATORY**

38.4 Demos

- No Gap, No Problem

39 Object Destruct Detector

39.1 Summary

Generates a signal (flag change) when the map object that is referenced by the zone is destroyed.

39.2 Dependencies

dcsCommon, cfxZones

39.3 ME Integration

Name	Description
OBJECT ID	THIS ATTRIBUTE IS FILLED BY ME AND **MUST NOT** BE CHANGED **MANDATORY**
NAME	THIS ATTRIBUTE IS FILLED BY ME AND **MUST NOT** BE CHANGED **MANDATORY**
method oddMethod	DML Method to bang! the flags connected to output! when the map object is destroyed. Defaults to “inc”
f! destroyed! objectDestroyed!	The flag to bang! when the map object is destroyed. Use only one synonym per zone. Defaults to “*none”

39.4 Demos

- Object Destruct Detection

40 Object Spawn Zones

40.1 Summary

Allows static objects and cargo to spawn according to an attribute (type)

40.2 Dependencies

dcsCommon, cfxZones, (cargoManager)

40.3 ME Integration

Name	Description
objectSpawner	Marks this ME Zone as a spawn zone. **Value of this attribute is ignored**, use it to describe what it spawns to make mission editor easier for you **MANDATORY**
f? spawn? spawnObjects?	An ME-compatible flag (e.g. 100) that this object spawner monitors for change. Whenever the value of the monitored flag changes, a new set of objects is spawned immediately, ignoring all maxSpawn and cooldown rules.
pause?	Flag to observe. Each time the flag's value changes, the spawner's 'paused' setting is forced to 'true. Used to 'pause a spawner
activate?	Flag to observe. Each time the flag's value changes, the spawner's 'paused' setting is forced to 'false'. Used to 'activate' a paused spawne
types	Type string array for the STATIC OBJECTS that are spawned. Example "White_Tyre, Red_Flag". These objects may look like units (if you use the type string for a ground unit or aircraft), but they are static. **WARNING**: Blanks are part of the type, and blanks directly before and after the last character are automatically stripped. All static objects given here are stacked on top of each other, and count as one instance (the example creates a tire with a red flag in the middle) **MANDATORY**
count	The number of times that the combined object in types is to be repeated. If count equals one (or is omitted), the objects defined in types are assembled in the center of the zone. Otherwise, the objects are distributed over the zone's circumference count times. Defaults to one
autoTurn	When *count* (see above) is greater than 1, the objects are distributed over the zone's circumference. If autoTurn is on, all objects are also turned by the same amount that they are rotated around the center. The result is that, e.g. a soldier that faces inward always faces inward. Defaults to false (all objects face the same heading)
country	The country for which the static objects are spawned. Examples: 0 = Russia, 1 = Ukraine, 2 = USA etc. Defaults to 2 (USA)
baseName	A designation (e.g. "Hill Marines") that is used to name units and groups from during unit spawning.

Name	Description
	If provided, you must ensure that resulting unit and group names are UNIQUE. If you do not assign a base name, a safe baseName that is guaranteed to prevent possible name conflicts is generated. A **convenience shortcut "*"** replaces baseName with the name of the zone. This is a safe baseName and can be used for all spawns. If two spawners have the same baseName, spawned units from one spawner may lead to existing (previously spawned) units being removed from the mission if they have the same name. So if for some reason a spawner appears not spawn units, make it a habit to check for potential name conflicts first. Default: <auto-generated safe baseName>.
cooldown	Number of seconds after the last spawn was removed before new objects are spawned. Default is 60 seconds
autoRemove	Wait for the spawned objects to be removed or destroyed, immediately start cooldown, then re-spawn according to rules. Default is false
autoLink	Only used when the spawner is linked to a unit: should the spawned objects move with the unit that the zone is linked to (usually ships, but can also be other objects). Defaults to true. Set to false if the spawner should 'drop' the objects to the ground.
heading	Orientation of the objects when they are spawned. Default is 0 (North)
weight	Used with cargo objects: the weight of this object in kg. Defaults to zero.
isCargo	Are these objects to be picked up by helicopters? Defaults to false.
managed	Used only if the objects spawned are cargo. If true, cargo objects are automatically registered with cfxCargoManager when they are spawned and cfxCargoManager is loaded). Defaults to true
maxSpawns	Number of times that the spawner spawns the objects. Defaults to 1 (one)
paused	A paused spawner will not spawn automatically (but can be forced to spawn via API or query flag f?). Set to true to pause spawning. Defaults to false.
requestable	This spawner should only spawn on request (i.e. via API or from other zones). Forces paused to true. Default value is false
useDelicates	Name of a Delicates Zone that is used to assign *delicate* (brittle, explodes when receiving minimal damage) status when this spawner spawns units You can use an asterisk ("*") as wildcard to refer to this zone Defaults to <none>

40.4 Demos

- ME Triggered Spawns
- Spawn Zones (training and lasing)
- Random Glory
- Helo Cargo

41 OwnAll

41.1 Summary

ownAll is a coordination/helper module that pulls together, and consolidates, information from multiple modules that deal with zone ownership

41.2 Dependencies

dcsCommon, cfxZones

41.3 ME Integration

Name	Description
ownAll	List of the names of zones, separated by comma "," that must be owned by the same faction and are monitored throughout the mission. Example: `Zone A, Another Zone, abc` You should include at least two (2) zones to the list, else you will receive a warning Supports all trigger zones, including (especially, really) DML zones with managed ownership like Airfield Zones, Owned Zones and FARP Zones. Since ownAll zones magaes its own ownership, you can also include ownAll zones to this value. Do this to create a cascade (or hierarchy) of ownership. **Mandatory**
red#	A direct output that represents the number of zones from the list that are currently held by RED faction Defaults to <none>
blue#	A direct output that represents the number of zones from the list that are currently held by BLUE faction Defaults to <none>
total#	A direct output (set only at the beginning of the mission) that represents the number of zones that are monitored

red!	Output to send a signal on when RED possess **ALL** zones that are listed Defaults to <none>
blue!	Output to send a signal on when BLUE possess **ALL** zones that are listed Defaults to <none>
method	DML method for outputs (excluding direct outputs) Defaults to “inc”
redLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the RED faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *redline*
redFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the RED faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *redFill*
blueLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the BLUE faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *blueLine*
blueFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the BLUE faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *blueFill*
neutralLine	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the NEUTRAL faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *neutralLine*
neutralFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the NEUTRAL faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *neutralFill*

41.4 Demos

- All is what I own

42 Owned Zones

42.1 Summary

Zones that can be conquered and generate signals when conquered. They can be marked on the map with color and text, and you have control over which color is displayed for which owner

42.2 Dependencies

dcsCommon, cfxZones

42.3 ME Integration

Name	Description
owner	Coalition that owns the zone at beginning of Mission. Can be 0, 1, 2 or “red”, “blue”, “neutral”. If nothing or some illegal value give, this defaults to neutral (0) **MANDATORY** **Note:** “owner” is a zone attribute that is implicitly available for **all** zones in DML. By default, all zones are owned by the neutral faction. All zones support the ‘owner’ attribute, as it is a core zone ability. By adding the “OwnedZones” module to your mission, you merely change the static ownership of a zone to a dynamic property managed by the OwnedZones module. **Note II:** Originally (pre version 2.x of ownedZones), this module combined the functionality of ownedZones and factoryZones. All production ability has been moved to factoryZone.
conquered!	Flag to bang! when this zone changes hands Defaults to <none>
redCap!	Flag to bang! when red captures this zone Defaults to <none>
redLost!	Flag to bang! when red loses control over this zone. This includes the zone turning neutral/becoming contested Defaults to <none>
blueCap!	Flag to bang! when blue captures this zone Defaults to <none>
blueLost!	Flag to bang! when blue loses control over this zone. This includes the zone turning neutral/becoming contested Defaults to <none>
neutral!	Flag to bang! when this zone becomes neutral or contested Defaults to <none>
ownedBy#	When present, the flag specified here is always set to the currently owning faction: 0 (neutral), 1 (red) or 2 (blue). This can be used to create win conditions, and changes on ownership will trigger standard ‘change’ inputs

Name	Description
	Defaults to <none>
unbeatable	“true” or “yes” makes it unbeatable/unconquerable. This zone can’t be conquered by the other side, but may still be targeted. Defaults to “no”
untargetable	“true” or “yes” makes it untargetable. Zone will not be targeted by troops with ‘attackOwnedZones’. The zone may still be conquered by the other side. Defaults to “no”
hidden	“true” or “yes” hides it. Zone is not shown on F10 Map. Defaults to “no”
title	If present, the value (text) is drawn starting at the center of the zone, in the lineColor of the current owner. The special wildcard “*” can be used to denote that the title is the same as the zone’s name. Defaults to <none> - no title is drawn.
redLine	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the RED faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *redline*
redFill	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the RED faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *redFill*
blueLine	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the BLUE faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *blueLine*
blueFill	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the BLUE faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *blueFill*
neutralLine	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the zone’s outline when it belongs to the NEUTRAL faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *neutralLine*
neutralFill	Four numeric values, separated by comma (e.g. 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the zone when it belongs to the NEUTRAL faction. RGBA values each range from 0.0 to 1.0 Defaults to config zone’s setting of *neutralFill*
method	DML output method for all outputs (“!”) that aren’t direct (“#”) Defaults to “inc”
numCap	Local override for the global numCap setting in ownedZonesConfig. Minimum number of ground units required to capture this zone. No units of the opposing faction must be inside the zone, neutral units are not counted. Defaults to ownedZonesConfig’s settings

Name	Description
numKeep	Local override for the global numKeep setting in ownedZonesConfig. Minimum number of ground units required inside the zone to keep a zone *after* capture. If less units than this number are inside the zone, it becomes neutral. Note that the mere presence of enemy units inside an owned zone usually (see exception below) does not make it contested/neutral. Defaults to ownedZonesConfig's settings
announcer announce	Announce capture by text and sound if this zone is captured. Overrides and defaults to config's *announcer* attribute setting.

42.4 Demos

- Owned Zones ME Integration
- My first factory

43 Persistence

43.1 Summary

This module provides persistence (load and save) functionality to modules. Must load before any mode that is to use persistence. Requires that the DCS instance that is running the mission be de-sanitized for lfs and io.

43.2 Dependencies

Persistence requires dcsCommon and cfxZones.

43.3 ME Integration

You configure the persistence module with a Trigger Zone named ‘persistenceConfig’

Name	Description
verbose	A value of “true” turns on debugging messages. Default is “false”
versionID	If present, this turns on version matching. When a mission starts up, persistence checks the value provided via the Zone with the one saved. If they do not match, the entire save data is discarded, and the mission starts fresh Defaults to <none>
root	Path to the DCS standard directory (usually “C:\userName\saved games\DCS.openbeta\” or “C:\userName\saved games\DCS\”). This value is passed from DCS to persistence. You can change this to adapt your missions to conform with more elaborate server setups. If you change this. Be sure that you know what you are doing, and initially have verbosity set to true, so you can see which directory your mission will save to. Defaults to your currently running DCS instance’s write dir.
serverDir	Path from the root directory (see above) to the Missions directory. Use this if you set up your DCS different (usually important for dedicated servers). Defaults to “Missions\”
saveDir	Name for the mission’s data directory. Defaults to “<mission name> (data)”. This directory is created in the serverDir automatically if it does not exist If you set saveDir to “”, the mission saves its data directly into serverDir Defaults to “<mission name> (data)” if a configuration zone is present, none without configuration zone (i.e. the data is written into serverDir)
saveFileName	Name for the file that persistence uses to write mission data. Defaults to “<mission name> Data.txt”
saveInterval	Controls auto-save. Any value larger than zero will turn on auto save. The value you give here is the number of minutes between auto saves. Auto-saves co-operate with manual saves, so you can use both methods in your mission

Name	Description
	Defaults to -1 (auto-save off)
cleanRestart?	DML Watchflag. A change signal on this input triggers a “fresh start” request: next time the mission starts up, it won’t load mission data. Defaults to <none>
saveMission?	DML Watchflag. A change signal on this input triggers a ‘manual’ save. Defaults to <none>

You control where persistence saves your mission’s data with the “root”, “serverDir”, “saveDir” and “saveFileName” attributes. Assuming a mission named “coolMission.miz” running from a standard DCS Install, persistence saves data as follows:

In a fully defaulted configuration (i.e. out-of-the-box), persistence uses the following defaults:

attribute	Default
root	The directory that you configured DCS to be the ‘home’ directory. In a freshly DCS install that would usually be C:\Users\(user name)\Saved Games\DCS\ Defaults to what DCS tells persistence is the current home directory
serverDir	The directory name inside “root” that contains all missions. It’s usually called “Missions\” and that is what persistence defaults to
saveDir	A folder (allocated if it doesn’t exist) inside serverDir where persistence saves the mission data as a separate file. You can use this to pool multiple missions’ data into the same folder. Defaults to “<mission name> (Data)” (see note below) **IMPORTANT NOTE** If you completely omit persistence’s config zone, it reverts to simplified save mode, and defaults saveDir to “” (empty string), saving the mission’s data file directly into the serverDir.
saveFileName	The name for the data (plain text in JSON format, can be edited with any text editor) file inside saveDir. Defaults to “<mission name> Data.txt”

You can use Trigger Zones with the ‘saveFlags’ attribute to list the flags that persistence should save.

Name	Description
saveFlags	A list of flags that you want to be saved with the mission. Supports local flags (e.g., “*go”) and numbered ranges (e.g. “3-17”). **MANDATORY**

43.4 Demos

- Being persistent

44 Player Score

44.1 Summary

A module to keep award and tabulate kills and feats for players

44.2 Dependencies

Player Score requires the modules dcsCommon and cfxZones.

If you want to save the score table to storage, you must also include the module *persistence*.

If you want to give players access to their score via Communications, add the cfxPlayerScoreUI module

44.3 ME Integration

Most of PlayerScore's aspects are controlled via a configuration zone named "playerScoreConfig" (exact match required) with the following attributes

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
aircraft	The fallback score to award for killing an aircraft if that unit wasn't found on the score table (name or type). Defaults to 50
helo	The fallback score to award for killing a helicopter if that unit wasn't found on the score table (name or type). Defaults to 40
ground	The fallback score to award for killing ground unit if that unit wasn't found on the score table (name or type). Defaults to 10
ship	The fallback score to award for killing a ship if that unit wasn't found on the score table (name or type). Defaults to 80
train	The fallback score to award for killing a train if that unit wasn't found on the score table (name or type). Defaults to 5
pkMod	Factor to multiply score for a player-kill (PvP). Defaults to 1 (same score)
ffMod	Factor to multiply score with for a friendly kill. Set to 0 to award no points for friendly kills. Defaults to -2 (negative double score for friendly fire kills)
planeLoss	Score to award when a player loses their airframe. This score is warded immediately. Set i to a negative value to deduct score from a player. Defaults to 0 (zero, plane loss has no consequence)
landing	Score for a successful landing, will also award a landing feat. Is only active if the value for landing is greater than zero. Also applies to helicopters. Default: 0 (no score, no feat awarded for landings).
announcer	If false, no kills are announced. Score is still kept. Defaults to true
scoreSound	Name of the sound file to play when a score is announced
badSound	Name of the sound file to play when killing own troops or being killed in PvP
saveScore?	Watchflag. When triggered, PlayerScrore saves the current score table to a plain text file.

Name	Description
	Requires the 'persistence' module be active in the mission, and the hosting DCS installation be de-sanitized. Defaults to <none>
incremental	When set to true, score save to plain text (via saveScore?) are appended to the save file if it already exists. Defaults to false (overwrite existing player score file)
rankPlayers	When set to true, players are ranked by their achieved score when saving player score to storage Defaults to false
scoreOnly	When set to true, a player's score omits all feat details on export to plain text file Defaults to true
deferred	When set to true, scores and feats are only awarded after the player successfully lands and remains for a certain time in a designated 'safeScore' zone. Defaults to false (immediately awards score)
delayAfterLanding	Number of seconds a player must remain in a safeScore zone after a successful landing before they are rewarded their accumulated score. Should they die in the interim, all accumulated scores are discarded. Only applicable if deferred is true. Default: 10 (seconds)
scoreFileName	Name of the plain text file (without extension) that scores are exported to when saveScore? is triggered. Defaults to "Player Scores"
reportScore	When queried to report the player's statistics, should the module include score? Set to false to suppress the player's score. Defaults to true (score is reported)
reportFeats	When set to true, the module also reports feats when queried for the player's statistics. Deafults to true (feats are reported)
reportCoalition	When set to true, the module reports the coalition's total with a player's score. Defaults to false (no coalition score reported)
noGrief	Controls if negative player scores also affect the player's coalition score. By default, to prevent 'griefing' (i.e. childish behavior), when a player is awarded demerits (negative score), that is only subtracted from the player's total, but not the coalition. That way, a griefing player can't intentionally sabotage a coalition's score by committing fratricide or crashing planes. When set to false, this protection is disabled, and griefing would be possible. Defaults to true (griefing protection is on, only positive player scores affect coalition score)
redScore#	Immediate Output flag that always contains the current total score for RED faction Defaults to <none>
blueScore#	Immediate Output flag that always contains the current total score for BLUE faction Defaults to <none>

You can also set up a mission-specific score table via a zone called “playerScoreTable” (exact match required) that can include named units and unit types as attributes as follows:

Name	Description
<tape or name> Type Exampe: BTR-80 Name Example: Big Kahuna	<Score as number> Example: 15 Example: 130 Note: The name is first checked again a unit’s name, and then against the unit’s group.

If you enable deferred scoring in the config zone, you designate zones in which scores are awarded after landing by adding the following attributes:

Name	Description
scoreSafe	Designates this entire zone as a zone where players can land inside and, after successfully landing and remaining the required amount of time (see *delayAfterLanding* config attribute) inside the zone, have their accumulated score and feats awarded. The value of this attribute determines which faction can have their scores awarded after landing as follows: • 0 or ’neutral’: all factions • 1 or ‘red’: only red faction • 2 or ‘blue’: only blue faction Defaults to ‘neutral’ (both red and blue can land here to have their scores awarded) **Mandatory**

You can place feats in ME with trigger zones that have the following attributes:

Name	Description
feat	Designates this zone as a zone in which the completion of certain actions leads to a feat being awarded to the player who completes it. The value of this attribute can restrict the feat to certain factions: • 0 or `neutral` – all players can achieve this feat • 1 or `red` – only red players can achieve this feat • 2 or `blue` – only blue players can achieve this feat Defaults to 0 (all players can achive this feat) **Mandatory**
featType	What a player needs to accomplish in order to have this feat awarded. Currently supports the following • `land` or `landing` – player lands their unit inside this zone • `kill` – players kills a unit that is currently inside this zone • `pvp` – player kills a unit that is currently inside this zone and is controlled by another player.

	Defaults to `kill`
description	Text of the feature that is to be awarded to the player. The description supports **wildcards** as follows. Feat-specific Wildcards: • `<player>` - callsign of the player • `<kplayer>` - callsign of other player when that plane was controlled by a player, 'unknown AI' else. • `<punit>` - name of the unit that the player is controlling • `<unit>` - name of unit that was killed • `<ptype>` - type (e.g. "A-10A") of the unit that the player is controlling • `<type>` - type (e.g. "A-10A") of the unit that was killed • `<pgroup>` - name of the group that the player's unit is with • `<group>` - name of the group that the unit that was killed belonged to General Wildcards • `<n>` creates a line feed • `<z>` - feat zone's name as set with ME • `<t>` - current mission time in the format "HH:MM:SS", e.g. "08:42:11". • `<lat>` - latitude of the feat zone's current position • `<lon>` - longitude of the feat zone's current position • `<ele>` - elevation of the feat zone's current position. • `<mgrs>` - feat zone's current position in MGRS coordinates • `<v: flagName>` - value of flag <flagName> • `<t: flagName>` - value of flag <flagName> interpreted as time. Format be further configured with the *timeFormat* attribute. • `<lat: unit/zone>` - latitude of unit/zone with that name • `<lon: unit/zone>` - longitude of unit/zone with that name • `<ele: unit/zone>` - elevation of unit/zone with that name. • `<mgrs: unit/zone>` - mgrs of unit/zone with that name • `<vel: unit/zone>` - velocity of unit/zone with that name. • `<alt: unit/zone>` - altitude of unit/zone with that name. • `<hdg: unit/zone>` - heading of unit/zone with that name • `<type: unit>` - type (e.g. "A-10A") of unit with that name • `<player: unit>` - player callsign who controls a unit with that name Defaults to `(some feat)`
awardLimit	Total number of times that this feat can be awarded during the life time of a mission. A negative value means 'infinite number of times' Defaults to -1 (feat can be accomplished an infinite number of times)
awardOnce	If set to true, this feat can only be awarded once per player. Note that awardLimit and awardOnce work cumulative: if you set awardOnce to true, and awardLimit to 2, the feat can be accomplished twice, but only by different players. Defaults to 'false', a player can achieve this feat multiple times during the same mission

You can restrict all scorable kills to kill zones (units can still be killed outside kill zones, but killing them will not award score to players)

Name	Description
killZone	Designates this zone as a zone in which killing other units can earn score or feats. All kills outside this or other kill zones do not award a score or feat. To count, at least the target must be inside a kill zone, optionally (see *duet* attribute, below) you can also enforce that the victorious player must be inside the same kill zone. The value of this attribute is ignored **Mandatory**
duet	If set to true, *both* units (player and target) must be inside the *same* kill zone. Defaults to 'false' (only killed unit must be inside kill zone)

Flags to trigger coalition score
You can use specially named zones "blueScoreFlags" (for blue coalition) and "redScoreFlags" (red coalition) to award score whenever one of the flags listed as attribute name changes its value. The spelling of the zone's names must match exactly.

Name	Description
<any flag name>	Whenever the value of the flag listed here changes, PlayerScore adds the amount given in the Value field to the relevant coalition. Negative values are valid. If you give a value of 0 (zero) or non-numeric value, PlayerScore will give you a warning when it starts the mission. The value of this attribute is ignored

44.4 Demos

- PlayerScore (Keeping The score)
- More Score
- Later Score
- Flag Score
- Player Score to Win

45 Player Zone

45.1 Summary

Counts the number of players currently in a zone, and generates signals when players enter or leave the zone.

45.2 Dependencies

Player Zone requires the modules dcsCommon and cfxZones.

45.3 ME Integration

Name	Description
playerZone	Tells DML that this trigger zone is a player zone. The value of this attribute specifies which faction's players are counted. Possible values • "0" or "neutral" – red and blue are counted • "1" or "red" – only red players are counted • "2" or blue – only blue players are counted Defaults to "neutral" (red and blue players are counted) **MANDATORY**
method pwMethod	DML output method. Defaults to "inc"
pNum#	Direct output: the number of qualifying player units that are inside the zone at this moment. Defaults to <none>
added!	Output to bang! when players units have entered the zone since the last time the zone checked. Note that there are edge cases where added! and removed! can signal a change even though the flag pNum remains the same: for example if at the same time one player unit leaves the zone (or is killed), and another player unit enters the zone. Defaults to <none>
removed!	Output to bang! when plyer units have left the zone since the last time that the zone has cheked. Note that there are edge cases where added! and removed! can signal a change even though the flag pNum remains the same: for example if at the same time one player unit leaves the zone (or is killed), and another player unit enters the zone. Defaults to <none>

45.4 Demos

- Players in the Zone

46 Pulse Flags

46.1 Summary

Flag “Heartbeat” – (somewhat) regularly sets/changes a flag

46.2 Dependencies

dcsCommon, cfxZones

46.3 ME Integration

Name	Description
pulse!	Marks this as a pulser. The value describes the flags to change on each pulse. The flags are changed according to the method attribute **MANDATORY**
method pulseMethod outputMethod	DML Flag output method Defaults to “flip”
done! pulsesDone!	This flag’s value is changed when the pulser completes a fully run of pulses. Can only happen when the *pulses* attribute supplies a positive number. Use only one synonym per zone. Defaults to <none>
triggerMethod pulseTriggerMethod	Watchflag condition for input flags. Use only one synonym per zone
activate? startPulse?	Watchflag. When triggered, a paused pulser is reset and then restarted. Use only one synonym per zone. Defaults to <none>
pause? pausePulse?	Watchflag. When triggered, a pulser is paused. Use only one synonym per zone. Defaults to <none>
onStart	When set to false, a pulser starts paused, else active. **Defaults to true** (pulser starts automatically)
pulses	The number of pulses to complete. • If set to -1, the pulser runs until the mission ends or the pause?-flag changes • If set to a number, the pulser will generate that many pulses. • If set to a range (e.g. “3-5”) the pulser will generate a random number of pulses within that range. Defaults to -1 (endless)
time pulseInterval	Seconds between pulses. You can supply a range (two numbers separated by a hyphen, e.g. “4-19”), the time between pulses is randomized after each pulse to a number in that range. Defaults to 1
zeroPulse	Usually, a pulser starts with an initial pulse (“pulse zero”). This initial pulse can be delayed by *time* by setting zeroPulse to false The effect is that the initial pulse happens after the first delay

Name	Description
	Default is true (initial pulse immediately)

46.4 Demos

- Pulsing Fun
- Frog Men Training
- The Zonal Countdown
- Watchflag Demo
- Forever-looping Spawners

47 Radio Menu

47.1 Summary

Adds a configurable player menu to the Communications→F10 Other menu, with up to four menu items (commands). Supports cooldown.

You can use the *attachTo:* attribute in a *radioMenu* to attach it to a *radioMainMenu* zone instead to group/cascade menus.

47.2 Dependencies

Radio Menu requires dcsCommon and cfxZones.

47.3 ME Integration

Name	Description
radioMenu	Name of the menu to install in the Communications→F10 Other menu or the *radioMainMenu* that you refer to in *attachTo:* (see below). **NOTE:** The same trigger zone **must not** have both a radioMenu and radioMainMenu (see below) attribute. **MANDATORY**
attachTo:	By default, a radioMenu is added to a mission's 'F10: Other' menu. You can use radioMainMenu zones (see below) to group radioMenus. If you use "attachTo:" (mind the colon ":"), this radio menu attaches to the radioMainMenu that is described as value for this attribute. The trigger zone that you refer to as value must exist, and have a radioMainMenu attribute. Defaults to <none> (radioMenu is installed into F10-Other)
coalition	The coalition that has access to this menu. If omitted or set to 'neutral', *all* coalitions have access. 'blue' or 'red' restricts access to that coalition. Defaults to <no coalition>
group groups	Restricts this menu only to groups with that name. Supports comma separated groups names, e.g. "Eagles 5, Uzi One, Aleph" will make this menu available to all members of those groups. **Supports wildcard** names: If you supply a group name "Cat*", all groups **that start with** "Cat" (WARNING: CASE SENSITIVE!) get access to this menu. For example, groups name "Cat Balou" and "Catobar Hornets" qualify, while the group named "catastrophy" does not (case mismatch for "C").

Name	Description
	Overrides any 'coalition' and 'type' attribute; when you add a 'group' attribute, those other attributes are ignored. Defaults to <no group restriction> **NOTE:** requires module cfxMX to load before radioMenus
type types	Restricts access to this menu to player units who control a unit that matches one of the listed types. You can list multiple types, separated by a comma (e.g., "F-15C, A-10A") Supports the class-wildcards 'plane' (all fixed-wing players) and 'helo' (all rotor-wings controlled by players). When you also supply a 'coalition' attribute, access to this menu is restricted to those players who match both. Defaults to <no type restriction> **NOTE:** requires module cfxMX to load before radioMenus
itemA	Name of itemA in this menu. If this attribute is omitted, no menu item appears
itemB	Name of itemB in this menu. If this attribute is omitted, no menu item appears
itemC	Name of itemC in this menu. If this attribute is omitted, no menu item appears
itemD	Name of itemD in this menu. If this attribute is omitted, no menu item appears
A!	DML flag to bang when itemA is chosen. Defaults to <none>
B!	DML flag to bang when itemB is chosen. Defaults to <none>
C!	DML flag to bang when itemC is chosen. Defaults to <none>
D!	DML flag to bang when itemD is chosen. Defaults to <none>
cooldownA	Cooldown (in seconds) after itemA is chosen before it becomes available again. Defaults to 0 (immediately available again)
cooldownB	Cooldown (in seconds) after itemA is chosen before it becomes available again. Defaults to 0 (immediately available again)
cooldownC	Cooldown (in seconds) after itemA is chosen before it becomes available again. Defaults to 0 (immediately available again)
cooldownD	Cooldown (in seconds) after itemA is chosen before it becomes available again. Defaults to 0 (immediately available again)
busyA	Message to display when itemA is chosen while cooldown is still active. Defaults to "Please stand by (<s> seconds)". Supports Time Wildcards <s>, <m>, <h>, <:s>, <:m> and <:h>
busyB	Message to display when itemB is chosen while cooldown is still active. Defaults to "Please stand by (<s> seconds)". Supports Time Wildcards <s>, <m>, <h>, <:s>, <:m> and <:h>
busyC	Message to display when itemC is chosen while cooldown is still active. Defaults to "Please stand by (<s> seconds)". Supports Time Wildcards <s>, <m>, <h>, <:s>, <:m> and <:h>
busyD	Message to display when itemD is chosen while cooldown is still active. Defaults to "Please stand by (<s> seconds)". Supports Time Wildcards <s>, <m>, <h>, <:s>, <:m> and <:h>

Name	Description
valA	Overrides radioMethod (see below) when this item is chosen to set flag A! according to this method. Example: #3 Defaults to <none> (no override)
valB	Overrides radioMethod (see below) when this item is chosen to set flag B! according to this method. Example: #3 Defaults to <none> (no override)
valC	Overrides radioMethod (see below) when this item is chosen to set flag C! according to this method. Example: #3 Defaults to <none> (no override)
valD	Overrides radioMethod (see below) when this item is chosen to set flag D! according to this method. Example: #3 Defaults to <none> (no override)
ackA	Acknowledge message when itemA was selected. Broadcast to all/coalition when itemA was chosen and not on cooldown. Supports wildcards. Defaults to <none> - no acknowledging message
ackB	Acknowledge message when itemB was selected. Broadcast to all/coalition when itemB was chosen and not on cooldown. Supports wildcards. Defaults to <none> - no acknowledging message
ackC	Acknowledge message when itemC was selected. Broadcast to all/coalition when itemC was chosen and not on cooldown. Supports wildcards. Defaults to <none> - no acknowledging message
ackD	Acknowledge message when itemD was selected. Broadcast to all/coalition when itemD was chosen and not on cooldown. Supports wildcards. Defaults to <none> - no acknowledging message
ackASnd	Name of sound file to play as a response to choosing itemA. Broadcast to all members of that group Default to <none>
ackBSnd	Name of sound file to play as a response to choosing itemB. Broadcast to all members of that group Default to <none>
ackCSnd	Name of sound file to play as a response to choosing itemC. Broadcast to all members of that group Default to <none>
ackDSnd	Name of sound file to play as a response to choosing itemD. Broadcast to all members of that group Default to <none>
method radioMethod	DML method to bang flags. Defaults to ‘inc’, meaning that each time that a menu item is chosen, the flag’s number is increased, generating a signal. To emulate ME’s native menu method that sets a flag to a value <v>, use that number <v> as method
radioTriggerMethod	Watchflag method for inputs Defaults to “change”
removeMenu?	Watchflag that triggers removal of entire menu
addMenu?	Watchflag that triggers adding the menu if it wasn’t shown or removed previously

Name	Description
menuVisible	When set (as per default) the menu is shown at the start of the mission. When set to false, the mission starts up with the menu hidden and requires a signal on addMenu? to appear Default to true (menu is visible on mission start)

Use the following wildcards (note that they are identical to messenger's 'display as time wildcards) in busyX:

- <s>
  remaining cooldown in seconds. E.g., if remaining cooldown is 254, <s> is replaced with '254'

- <m>
  remaining cooldown as whole minutes. E.g., if remaining cooldown is 254, <m> is replaced with '4', while 3891 returns "64"

- <h>
  remaining cooldown as whole hours. E.g., if remaining cooldown is 3891, <m> is replaced with '1'

- <:s>
  remaining cooldown converted to a seconds time value (0-60) and formatted with leading zero. E.g., if remaining cooldown is 64, <:s> is replaced with '04'

- <:m>
  remaining cooldown converted to a minutes time value (0-60) and formatted with leading zero. E.g., remaining cooldown is 64, <:m> is replaced with '01', and 803 returns "13"

- <:h>
  remaining cooldown converted to an hours time value and formatted with leading zero. E.g., if remaining cooldown is 3764, <:h> is replaced with '01'

Additionally, busyX and ackX also support the full gamut of wildcards that valet and messenger support.

Name	Description
radioMainMenu	Name of the menu to install in the Communications→F10 Other menu or the *radioMainMenu* that you refer to in *attachTo:* (see below). **MANDATORY**
attachTo:	By default, a radioMainMenu is added to a mission's 'F10: Other' menu. You can use radioMainMenu zones to cascade radioMainMenu or group radioMenus. If you use "attachTo:" (mind

Name	Description
	the colon ":"), this radio main menu attaches to the radioMainMenu that is described as value for this attribute. The trigger zone that you refer to as value must exist, and have a *radioMainMenu* attribute. Defaults to <none> (this radioMainMenu is installed into F10-Other)
coalition	The coalition that has access to this main menu. Applies to all attached menus. If omitted or set to 'neutral', *all* coalitions have access. 'blue' or 'red' restricts access to that coalition. Defaults to <no coalition>
group groups	Restricts this menu and all attached menus to groups with that name. Supports comma separated groups names, e.g. "Eagles 5, Uzi One, Aleph" will make this menu available to all members of those groups. **Supports wildcard** names: If you supply a group name "Cat*", all groups **that start with** "Cat" (WARNING: CASE SENSITIVE!) get access to this menu. For example, groups name "Cat Balou" and "Catobar Hornets" qualify, while the group named "catastrophy" does not (case mismatch for "C"). **Overrides any 'coalition' and 'type' attribute**; when you add a 'group' attribute, those other attributes are ignored. Defaults to <no group restriction> **NOTE:** requires module cfxMX to load before radioMenus
type types	Restricts access to this main menu (and all menus attached to this main menu) to player units who control a unit that matches one of the listed types. You can list multiple types, separated by a comma (e.g., "F-15C, A-10A") Supports class-wildcards '*plane*' (all fixed-wing players) and '*helo*' (all rotor-wings controlled by players). When you also supply a 'coalition' attribute, access to this menu is restricted to those players who match *both*. Defaults to <no type restriction> **NOTE:** requires module cfxMX to load before radioMenus

47.4 Demos

- Reinforcements A La Carte
- Recon Mode reloaded
- Guardian Angel reloaded
- Sequencing Fun
- Slot Blocking and You
- BFM Combat Trainer
- What’s on the cascading menu
- Foggy bottoms

48 Radio Trigger

48.1 Summary

Provides an interface for ME-based Communication→Other Radio Items, allowing multiple uses by re-setting the flag after it has been triggered.

48.2 Dependencies

dcsCommon, cfxZones

48.3 ME Integration

Name	Description
radio?	Watchflag. Triggers a radio cycle, then resets this flag to its pervious value **MANDATORY**
triggerMethod radioTriggerMethod	Method that triggers the Watchflag Defaults to 'change'
method rtMethod	Method how the output flag should be triggered. Defaults to 'inc'
out! rtOut!	DML Flag to set when the module triggers Defaults to <none>

48.4 Demos

- Radio go go

49 Raise Flag

49.1 Summary

A simple, DML way to set flags to values. Supports randomization and delayed setting of flags.

49.2 Dependencies

dcsCommon, cfxZones

49.3 ME Integration

Name	Description
raiseFlag!	Marks this as a flag raiser. The value of this attribute is the flag that is to be raised after a delay. **MANDATORY**
method	DML method to set the flag to **Examples:** • inc - increment the flag specified in raiseFlag! • #5 – set the flag specified in raiseFlag! to the number 5 Defaults to 'inc' – the flag's value will be incremented by one
afterTime	Amount of time (in seconds) after mission start to set the flag. Can be a range. If a range is given, the time is a random number from this range. Defaults to 0.5 seconds after mission start
stopFlag?	Watchflag, only useful in conjunction with afterTime. When triggered and raiseFlag is still waiting for afterTime, raiseFlag is 'disarmed' and no flag will be raised in the future. Once stopped, raiseFlag cannot be re-started.
triggerMethod raiseTriggerMethod	Watchflag condition for stopFlag?

49.4 Demos

- Flag Fun
- Attack of the CloneZ
- Send in the Clones

50 Reaper

50.1 Summary
Places laser-targeting drones on your map

50.2 Dependencies
dcsCommon, cfxZones

Optionally requires radioMenus (for attachtTo:) or bank (for priced launches)

50.3 ME Integration

Name	Description
reaper	Tells DML that this zone is a reaper zone that is patrolled by a drone. The value of this attribute defines the type of drone. Possible values are • Reaper – the drone is a type “MQ-9 Reaper”, max altitude 9500m (28000ft), speed 200 km/h (110kts) • Predator – the drone is a type “RQ-1A Predator”, max altitude 7500m (22000ft), speed 120 km/h (65 kts) Defaults to “Reaper” **Mandatory**
alt	Altitude (in meters) for the drone’s racetrack pattern. Note that the altitude also defines the drone’s visual range, which is roughly equal to the altitude. Predator and Reapers have different ceiling altitudes.
Coalition	Faction that owns the drone. Possible values are • 1 or “red” • 2 or “blue” • Note that neutral drones are automatically switched to BLUE Defaults to BLUE
onStart	If set to true, the drone is launched on mission start, at no cost. Defaults to true (drone launches at mission start)
code	Laser code for lasing. Defaults to 1688
doSmoke	Visual aid for targeting. When a drone starts targeting a unit, it can also place smoke at the location of the target. The smoke marker lasts circa 5 minutes and does not move with the unit. Defaults to false (no smoke when marking units)
smokeColor	Color of smoke to place at the location of the target. Valid values are • 0 or “green” • 1 or “red” • 2 or “white” • 3 of “orange”

Name	Description
	• 4 or “blue” Defaults to “blue”
cost	Only used when config’s ‘useCost’ is enabled and the ‘bank’ module is loaded. Cost in ‘fund units’ to launch a drone. Does not affect initial launches if the onStart attribute is true When actice, the cost is included in reaper’s launch UI Defaults to 700
autoRespawn	If a drone is disappeared (for any reason), it is instantly replaced with a new one (for free) if this attribute is set to true Defaults to false
launchUI	If set to true, this reaper zone is included with the reaper UI Defaults to true (zone is included with launch UI)
statusUI	If set to true, this zone’s drone is included in the status reports that can be accessed with reaper’s UI. Defaults to true (drone is included in reaper status reports)
launch?	Input that when triggered launches this zone’s drone. When cost enabled and the bank module present, a funds check is performed and the drone is only launched if sufficient funds are present. Defaults to <none>
status?	Input that when triggered outputs this zone’s drone status to the owning side. Defaults to <none>

50.4 Demos

- Reaper, man

51 Recon Mode

51.1 Summary

Provides out-of-the-box advanced recon abilities for aircraft.

51.2 Dependencies

Tbc

51.3 ME Integration

Zone to make aircraft scouts or remove scout abilities:

Name	Description
scout	Marks all aircraft (fixed- and rotor-wing) inside the zone. If the attribute's value is 'true', the aircraft have recon ability. If the value is false, they are 'blind', i.e. they have no recon abilities. Defaults to 'true' **MANDATORY**
dynamic	Controls if all units that start with the same name are automatically included. This is helpful for clone zones that base all names for clones on the name of the unit in the template. Defaults to false

Zone to mark priority and blacklisted ground forces:

Name	Description
recon	Marks all ground groups inside this zone as recon relevant. If the value is "black" all groups that have at least one inside this zone are added to the blacklist, otherwise they are added to the priority target list Defaults to "prio" **MANDATORY**
dynamic	Controls if all groups that start with the same name are automatically included. This is helpful for clone zones that base all names for clones on the name of the unit in the template. Defaults to true
prioMessage	A message that is displayed to the coalition that the scout belongs to. Can contain most of the text wildcards that the messenger module provides: • <n> creates a new line • <z> is replaced with zone's name • <t> is current time in HH:MM:SS format • <lat> the latitude of the discovered group's current position • <lon> the longitude of the discovered group's current position • <ele> the elevation (in feet or meters, as determined by the imperialUnits attribute in reconModeConfig) • <mgrs> the discovered group's current position in MGRS coordinates

Name	Description
	Only applicable for priority targets (i.e. recon's value is something other than "black"), ignored otherwise Default is <none>, i.e.., no message is displayed
spotted!	DML output flags to bang when a group is spotted. Flag is banged *in addition* to the module's global "prio!" flag. Uses the config zone's method. *Only applies to prio zones* (ignored when blacklist zone) Defaults to <none>
silent	Only applicable to priority groups. When present and set to true, the recon report and map mark are suppressed for any groups defined with this zone. Note that this does NOT apply to any prioMessage you have defined for this zone, as that will still be displayed. Defaults to false

Config Zone Settings

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
autoRecon	If true, all planes are automatically treated as actively reconnoitering. **NOTE** This is on by default. To avoid excessive scouting activity, you should reduce the number of active scout planes with enabling or disabling one of the following attributes: redScouts (off), blueScouts (off), greyScouts (off), playerOnlyRecon (on) Default: true
redScouts	If true, all red planes are included as scouts when autoRRecon is true. Default is false
blueScouts	If true, all blue planes are included as scouts when autoRecon is true. Default is **true**
greyScouts	If true, all neutral planes are included as scouts when autoRecon is true. Default is false
playerOnlyRecon	If true, only player aircraft are included as scouts when autoRecon is true. AI planes will not be automatically included as scouts. **IMPORTANT** This condition is applied **in addition** to blueScouts and redScouts. If you disallow red scouts, red players will not automatically be added to the list of scouts. Defaults to false
reportNumbers	If true, the F10 map markings include a unit count of the group at the time the group was discovered. Default is true
applyMarks	If true, discovered groups are marked on the F10 map. Default is true
marksFadeAfter	Time (in seconds) after which a mark on the F10 map is automatically removed. Set to a negative value (e.g., -1) for 'eternity' (marks never time out) Defaults to 30*60 = 1800 seconds (= 30 minutes)
marksLocked	When set to true, marks cannot be removed by players, and will disappear automatically after they time out (see marksFadeAfter, above) or the group is destroyed (when autoRemove is true)

Name	Description
	Defaults to false (player can remove marks)
announcer	If true, discovered groups are announced via text. Default is true
detectionMinRange	The detection range of a recon plane under worst conditions (low-level flying). Default is 3000 (3 km)
detectionMaxRange	The detection range of a recon plane under best conditions (high-altitude). Default is 12000 (12 km)
maxAlt	The altitude at which a plane achieves maxDetectionRange. Default is 9000 (9 km, 27'000 ft)
prio+	A flag in ME that is increased every time that a priority unit is detected
detect+	A flag in ME that is increased every time that a normal (non-priority) is detected
reconSound	The name of the sound file to play when e recon event occurs. Defaults to <nosound>, which will not play a sound
autoRemove	When a detected group is destroyed, that group's mark is immediately removed from the map if this attribute is set to true Defaults to true
mgrs	Defines if the location of the group that is detected is given in Lat/Lon (default) or MGRS. Set to true to enable MGRS. Default is false (coordinates are displayed in Lat/Lon)
imperial imperialUnits	When set to true, the value given in <ele> (elevation) is in feet, otherwise in meters Defaults to false (ele is returned in meters)
activate? on?	Watchflag to monitor for a change. If a change is detected, Recon Mode goes into active state. Defaults to <none>
deactivate? off?	Watchflag to monitor for a change. If a change is detected, Recon Mode turns off Defaults to <none>
active	Set to false to start Recon Mode in disabled (off) state. Defaults to true (Recon Mode enabled)
reportTime	Number of seconds that a recon message stays on the screen. Defaults to 30.

51.4 Demos

- Recon Mode
- Recon Mode - Reloaded

52 Rnd Flags

52.1 Summary

Can randomly set flags – from a pool of flags, in many different methods

52.2 Dependencies

dcsCommon, cfxZones

52.3 ME Integration

Name	Description
RND!	Marks this as a randomizer. Set of flags, as a comma (‘,’) separated list of the flag names that can be chosen from. The flag names can appear in any sequence. Supports ranges like “2-7” (ME numbers only). Flag names can be included multiple times, including the same flag name multiple times simply increases the likelihood that this number is chosen. Examples: “2, 4, A, A, F, 6” “A9, 3-18, C33, samAttack, 11-11” **MANDATORY**
method rndMethod	DML Output flag method. **Defaults to “inc”**
pollSize	Number of items to choose from the set of flags during a cycle. Can be a range: two numbers separated by a hyphen, e.g. “2-5”. When a range is given, pollSize is randomized each cycle to a number between the lower and upper bounds, inclusive. Defaults to 1.
remove	When set to true, the flags that were chosen during a cycle are removed from the set of flags. Defaults to false.
reshuffle	When set, the original full set of flags is restored when all flags have been removed Defaults to false
f? in? rndPoll?	DML Watchflag to start a random cycle. Defaults to <none set> You can use any synonym, but only one per Zone
triggerMethod rndTriggerMethod	Watchflag condition when to trigger. Defaults to “change”
onStart	If true, a cycle is run for this randomizer 0.25 seconds after the mission starts. Defaults to false **NOTE**: if no rndPoll? (nor synonym) is specified, and onStart is false, the randomizer will never activate. You’ll receive a warning.

Name	Description
	If you specify neither onStart nor an input rndPoll? (or synonym), onStart is automatically set to true, so that the randomizer runs exactly once, 0.25 seconds after mission start.
done!	The flag number to bang! when the randomizer has run out of flags to change, and reshuffle is false (randomizer did nothing) Is banged! every time that the randomizer runs a cycle on an empty flag set

52.4 Demos

- Random Glory
- Random Death
- Pulsing Fun
- Attack of the CloneZ

53 Scribe

53.1 Summary

This module **records all players accomplishments** (engine start-ups, time in a type, landings, take-offs, crashes) and can – per player – give them a summary. Supports persistence so a player can see their accumulated stats **across multiple replays**.

Scribe can share mission data with other scribe modules (using persistence)

53.2 Dependencies

Scribe requires dcsCommon, cfxZones and cfxMX

Scribe requires persistence to persist data

Scribe records all successful rescues when csarManager is active.

53.3 ME Integration

Scribe currently uses no map-specific zone, all features are controlled through the config zone

53.4 Demos

- On the Record
- rvb RED,rvb BLUE

54 Sequencer

54.1 Summary

Creates signals in a pre-determined sequence of flags

54.2 Dependencies

Sequencer requires dcsCommon, cfxZones.

54.3 ME Integration

Name	Description
sequence!	A comma-separated list of flags that should be banged! in exactly that order. Example: “startGround, startSAM, startSirens” Value of this attribute is ignored **MANDATORY**
interval invervals	A comma-separated list of durations (in seconds) between the flag banging. The number of intervals does not have to match the number of flags given under sequence!, the values simply repeat. A single value means that the same interval is applied between all stages. You can supply value ranges instead of a single value, and the sequencer picks a random value from that range. Example: “10, 12-17, 33” Defaults to 86400 (24 hours)
next?	A signal on this input causes a running sequencer to end the current count-down to the next stage, and immediately start the next stage. Defaults to <none>
onStart	If set to true, the sequencer starts the first sequence 0.25 seconds into the mission (this delay allows all other modules to load and initialize before the first signal is sent from the sequence) Defaults to false (sequencer requires a startSeq? signal.
zeroSequence	Controls if the sequencer operates in “signal-wait” or “wait-signal” mode: if it starts with a signal (zeroSequence is true) or a wait (zeroSequence is false) Defaults to true (start with signal, “signal-wait” mode)
loop	If true, the sequence repeats endlessly or until stopped with ‘stopSeq?’ or ‘reset’. If false, the sequence ends after the last stage, and a signal is sent on done! Defaults to false
done! seqDone!	The signal to send when the sequencer has ended the sequence Defaults to <none>
startSeq?	A signal on this input starts a paused or un-started (when onStart is false) sequencer. If the sequence was paused, the count-down resumes at the moment it was paused.

Name	Description
	If the sequence is already running it has no effect. Note that a sequence that has ended (run through all stages) cannot be restarted with startSeq? but must be reset first. Defaults to <none>
stopSeq?	A signal on this input pauses a running sequence. The current state of the stage's timer is preserved. If the sequencer is already paused, this has no effect. Defaults to <none>
resetSeq?	A signal on this input resets the sequencer to the initial state: paused (if onStart is false), sequence stage one, duration one. A sequence that has ended (run out of stages) can be reset and then started again. Defaults to <none>
method seqMethod	The method to use for all outputs. Defaults to 'inc'
triggerMethod seqTriggerMethod	The trigger method for all inputs. Defaults to "change"

54.4 Demos

- Sequencing Fun

55 SittingDucks

55.1 Summary

A module that allows ‘destructible player slots’ for multiplayer.

55.2 Dependencies

SittingDucks requires dcsCommon, cfxZones. And stopGap

To run, the mission must as multiplayer, and the server must run the following add-ons:

- SSB (simple slot block)
- stopGapGUI

55.3 ME Integration

SittingDucks requires no ME integration past the config zone

55.4 Demos

- Sitting Ducks in a Barrel

56 Slotty

56.1 Summary

Slotty provides **single-player slot blocking** and a fallback for **multiplayer slot-blockin**g if the server that is running your mission does not have SSB installed.

Slotty conforms to the **SSB ("Simply slot block") protocol** and kicks players that are trying to 'slot' into an aircraft that is marked as not available. Modules like *ssbClient*, for example, use this protocol to prevent players from slotting into aircraft that are located on airfields or FARPs that belong to the enemy.

56.2 Dependencies

None.

56.3 ME Integration

Add the *Slotty* module to your mission in a DOSCRIPT action during MISSION START

56.4 Demos

- non SSB & SP slot blocking

57 Smoke Zones

57.1 Summary

Places a colored permanently refreshing smoke at the center of the zone

57.2 Dependencies

dcsCommon, cfxZones

57.3 ME Integration

Name	Description
smoke	Adds a permanent smoke effect to the center of the zone. Possible values for the smoke effect are: • "green" or "0" • "red" or "1" • "white" or "2" • "orange" or "3" • "blue" or "4" • "random", "?" or "rnd" (random color from above) **MANDATORY**
paused	When true, this trigger zone will not start smoke at mission start, but wait for a signal on the *startSmoke?* flag. Defaults to false (smoke starts at mission beginning). Note that if you set *paused* to true and omit a *startSmoke?* attribute, you cannot start this zone's smoke in the mission.
startSmoke? ~~f?~~	DML input to listen for a command/signal. When this input hears/receives the specified command/signal, smoke starts to emit from the zone (if it wasn't already emitting before, no change otherwise). Defaults to <none>
stopSmoke?	DML input to listen for a command/signal. When this input hears/receives the specified command/signal, smoke is scheduled to stop emitting from this zone. Note that **it can take up to 5 minutes** for the colored smoke to stop (unlike *fireFX*, there are no provisions in DCS for colored smoke to be extinguished immediately, so we have to wait for it to time out). Defaults to <none>
agl alt altitude	Altitude (in meters) above ground that the smoke should be created. Defaults to 1 meter
triggerMethod smokeTriggerMethod	Conditions when the inputs should trigger. Used with the "Flag Model" Defaults to 'change'

57.4 Demos

- Smoke’em DML Intro
- Random Glory
- Once, twice, three times a maybe

58 Smoking

58.1 Summary
Gives colored ‘smoke trail’ ability to player aircraft

58.2 Dependencies
smoking requires dcsCommon and cfxZones.

58.3 ME Integration

Name	Description
smoking	Marks this zone as a smoke trail designator that controls the behavior of all player aircraft that spawn inside this zone. Note that **the presence of a single zone with this attribute switches all aircraft in the mission to require individual ‘smoking’ designations.** A player aircraft that does not spawn inside such a zone no longer has access to the smoke trail function The value of this attribute defines the color of the smoke trail. Valid colors are (without quotes “”) • “green” or “0” • “red” or “1” • “white” or “2” • “orange” or “3” • “blue” or “4” • “rnd” or “random” or “?” Defaults to “white” **MANDATORY**
~~alt~~	For later expansion, currently not expected to function

58.4 Demos
- The smoke is ON

59 Spawn Zones

59.1 Summary

Allows spawning of ground units based on the types attribute (text)

59.2 Dependencies

dcsCommon, cfxZones, commander, groundTroops, (Helo Troops)

59.3 ME Integration

Name	Description
spawner	Marks this ME Zone as a spawn zone. The value for this attribute is a "type string" list that describes the ground units that are to be spawned. Example "Roland ADS, Roland Radar, Roland ADS" or "Soldier M4" – **WARNING**: Blanks are part of the type, and blanks before and after the last character are automatically stripped. For a full reference of objects and their types, see here https://github.com/mrSkortch/DCS-miscScripts/tree/master/ObjectDB and use whatever is given as value for the "typeName" attribute, e.g. "Soldier M249" for the "INF Soldier M249.lua" **MANDATORY**
typeMult	A number. This is a multiplicator for the number of troops created. This will simply take above type string, and add n copies of the type string. Allows you to quickly create large groups with little effort Defaults to <none>
~~f?~~ spawn? spawnUnits?	Input (watchflag) to tell the spawner when to spawn. Each time that a signal/command is received on this input, a new spawn cycle is forced, ignoring all other settings like maxSpawn, cooldown, paused, etc. Defaults to <none> (no flag to observe) Use only one synonym per zone
pause?	Input. Each time that a signal/command is received/heard, the spawner's 'paused' setting is forced to 'true. Used to 'pause a spawner Defaults to <none>
activate?	Input. Each time that a signal/command is received/heard, the spawner's 'paused' setting is forced to 'false'. Used to 'activate' a paused spawner Defaults to <none>
country	The country (a number) the units that spawn belong to, e.g. "22" for Switzerland (**Warning**: unlike many other zone extensions, we use a ***Country***, not a Coalition for this attribute. The coalition is determined

Name	Description
	by which Faction the country belongs to as is defined when you create the mission, or by using the faction editor. Common Countries are Russia = 0, Ukraine = 1, USA = 2, UN Peace Keepers = 82 You can find a reference of all country codes here: https://wiki.hoggitworld.com/view/DCS_enum_country). Defaults to 0
masterOwner	A string that references another ME Zone by name. It must match that Zone's name exactly, and that zone must have an owner (e.g. defined as an cfxOwnedZone or FARPZone). **A spawner only spawns *automatically* when the masterOwner's owning faction is the same as the spawner's country affiliation.** On the map, the spawner does not have to be inside the masterOwner's zone, it can be hundreds of miles away. You can use this to start spawning reinforcements in a completely unrelated part of the map when units conquer the masterOwner zone. If no masterOwner is specified, the Spawner spawns as directed and disregards any surrounding zones that happen to be owned Defaults to <none>, spawn units as directed by "county"
baseName	A designation (e.g. "Hill Marines") that is used to name units and groups during unit spawning (advanced uses only, usually safe to ignore). **If provided, you must ensure that resulting unit and group names are UNIQUE.** If you do not assign a base name, a safe baseName that is guaranteed to prevent possible name conflicts is generated. A **convenience shortcut "*"** replaces baseName with the name of the zone. This is a safe baseName and can be used for all spawns. If two spawners have the same baseName, spawned units from one spawner may lead to existing (previously spawned) units being removed from the mission if they have the same name. So if for some reason a spawner appears not to spawn units, make it a habit to check for potential name conflicts first. Default: <auto-generated safe baseName, based on zone name>.
cooldown	Time interval (in seconds) from when a new group can be produced after the last group was removed from the spawner. Defaults to 60 (seconds)
autoRemove	Usually, a spawner retains ownership of a group that is produced, and will re-start the spawning cycle only after it was removed. If you add the autoRemove attribute with a "yes" or "true" value, the Spawner will automatically re-start the spawning cycle (cooldown, produce) as soon as the new group has spawned. You can use this to automatically give orders and have units move out after they have spawned (similar to how OwnedZones spawn attackers). Be advised that you can create a lot of vehicles on your map in a very short time, so be careful when using autoRemove.

Name	Description
	Defaults to 'false'
heading	The direction the spawned group it oriented to, from the center of the spawn zone. Defaults to 0
formation	The "formation" of the spawned group *when in assembles on the spawn location*. See dcsCommon for supported group formations. Note: this is not the formation that units assume to move to their objective. Defaults to 'circle_out'.
moveFormation	The formation that the units assume when moving to a destination. Legal values are (warning: case sensitive!) • `Cone` • `Rank` • `Diamond` • `Vee` • `EchelonR` • `EchelonL` • `Custom` Note that when units have attackOwnedZones oders, they ignore moveFormation and try to move over roads. Defaults to "Custom"
drivable	Controls whether a spawned unit can be controlled via from a player with the "Combined Arms" module. Note that some infantry units (e.g. Igla, Stinger) can also be 'drivable' although they are infantry, not vehicles. Defaults to false (unit is not player-controllable)
paused	When paused, a spawner only spawns when other scripts tell it to (e.g. your own scripts, cfxHeloTroops, triggers). Defaults to "no"
orders	This interfaces to groundTroops. See the main manual's *Orders* section for possible values. Spawners support the "wait" prefix (e.g. "wait-attackOwnedZone") to tell spawned troops that they should wait until picked up by a player-controlled helicopter (→heloTroops) before starting on their orders. Use "lase" if you want your spawned troops to laser-designate all targets within their range (see range, below) Defaults to "guard"
range	An attribute used to pass a range (in meters) value to some orders (e.g. JTAC lase range, detection/engage range). The range attribute has no effect on attackOwnedZone or captureHandHold. Defaults to 300 (meters)
target	An attribute used to pass a target (trigger) zone when used in conjunction with the 'attackZone' orders Defaults to <none>

Name	Description
maxSpawns	The maximum number of times that this spawner spawns groups. Set it to a positive number (e.g. 3) to spawn that many time. Set it to a negative number for an unlimited number of spawns (default is -1). Set it to zero (0) and the spawner will never spawn. Defaults to -1 (unlimited)
requestable	Interfaces with other modules (e.g. HeloTroops). If you set this value to true, troops will only spawn on request via `cfxSpawnZones.spawnWithSpawner()`. See the API section on how to get a list of eligible spawners. Automatically interfaces with HeloTroops and other enhancements. Defaults to false (cannot be requested by heloTroops)
trackWith:	List of groupTracker zones. All spawned groups are added to these groupTrackers. If you have stacked the tracker on the same zone as the spawner, you can use a single asterisk '*' as zone name. Supports a comma-separated list of trackers if you simultaneously want to pass the spawned groups to multiple trackers Defaults to <None>
useDelicates	Name of a Delicates Zone that is used to assign *delicate* (brittle, explodes when receiving minimal damage) status when this spawner spawns units You can use an asterisk ("*") as wildcard to refer to this zone Defaults to <none>

59.4 Demos

- Random Death
- Moving Spawners
- Moving Spawners II
- Helo Troops

60 ssbClient

REQUIRES THAT SSB INBSTALLED ON HOSTING MP SERVER *AND* MISSION TO RUN AS MP

60.1 Summary

This module allows intelligent slot-blocking of aircraft based on airfield/FARP ownership, and single-use aircraft ("crash them and lose them").

60.2 Dependencies

ssbClient requires dcsCommon, cfxGroups and cfxZones

60.3 ME Integration

Most of ssbClient is controlled via the config zone that must be named "cfxSSBClientConfig"

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
singleUse	A value of "true" turns on single-use: an airframe is blocked after crashing it. Note that this requires that the server's SSB setup sets kickReset to false in SSB. Defaults to false
reUseAfter	If singleUse is **en**abled, this optional attribute controls after how long a delay (in seconds) the slot may be re-used. This can simulate replacements arriving after some time. Setting this value to -1 blocks the slot for the remainder of the mission. Defaults to -1 (remain blocked)
allowNeutralFields	If set to "true", aircraft can spawn on neutral airfields (otherwise they are blocked). Defaults to false (neutral fields do not allow blue nor red aircraft to spawn
maxAirfieldRange	Maximum range in meters to find an airfield/FARP for a 'from ground' start. If no airfield is found that slot will be permanently open. Defaults to 3000 meters
keepInAirGroups	For performance reasons, ssbClient strips all slots for air-starting aircraft from its observation list. In some cases (e.g. when you want to bind the availability of an air-starting aircraft slot to the ownership of an airfield) ssbClient must also manage air-starts. Set this value to true to also retain air-starting slots. Defaults to false
enabledFlagValue	This reflects SSB's flag value of that same name. **DO NOT CHANGE THIS UNLESS YOU ARE ABSOLUTERLY SURE YOU KNOW WHAT YOU ARE DOING**. Defaults to 0

Name	Description
enabledFlagValue	This reflects SSB's flag value of that same name. **DO NOT CHANGE THIS UNLESS YOU ARE ABSOLUTERLY SURE YOU KNOW WHAT YOU ARE DOING**. Defaults to enabledFlagValue + 100

Additionally, ssbClient supports ssbClient control zones that attach themselves to the closest airfield/FARP and can be used to open and close them with DML flags:

Name	Description
ssbClient	Marks this zone as a control zone **for the airfield/FARP that is closest** to this zone. **MANDATORY**
open?	DML Input Watchflag. When triggered, this airfield "opens", allowing planes spawn from this airfield. Note that ownership rules still apply. Defaults to <none>
close?	EML input Watchflag. When triggered, this airfield/FARP "closes". Aircraft can still land and depart from a closed airfield, but player aircraft that originate from there will no longer be able to spawn, their slots are blocked. Defaults to <none>
openOnStart	When set to false, the airfield is closed on mission start, all slots for aircraft spawning here are blocked Defaults to true (airfield is open, allowing slots to spawn according to faction ownership of airfield/FARP)
ssbTriggerMethod	Method for Watchflags. Defaults to "Change"

60.4 Demos

- Slot-Blocking and You
- non SSB & SP slot blocking

61 StopGap / StopGapGUI

61.1 Summary

Fill empty player ground slots with static aircraft of the same type and livery until players slot into them. Makes airfields look much more alive.

For multiplayer servers, install stopGapGUI to avoid synchronization issues (server only)

61.2 Dependencies

dcsCommon, cfxZones, cfxMX

For Multiplayer, you must install the stopGapGUI server extension on the server.

61.3 ME Integration

To *exclude* aircraft from stopGap, use trigger zones and the following attributes:

Name	Description
stopGap	When the value of this attribute is **false**, any player aircraft inside this zone will not receive a static stand-in, their slot remains empty. Defaults to true (unit receives a static stand-in) **MANDATORY**

Alternatively, you can disable StopGap only when the mission is run in single-player mode (this is useful when, due to some map properties, the aircraft 'falls out of the sky' in single-player, but works well in multiplayer due to StopGapGUI's superior synching.

Name	Description
stopGapSP	When the value of this attribute is **false**, any player aircraft inside this zone will not receive a static stand-in **in single-player mode**, their slot remains empty. In multiplayer mode, the slot is filled with a static stand-in. Requires stopGapGui on the server. Defaults to true (unit receives a static stand-in) **MANDATORY**

61.4 Demos

- No Gap, No Glory
- Caucasus Hangar
- My Immortal

62 Sweeper

62.1 Summary

Sweeper zones remove objects that have remained inside their trigger zone for longer than a definable time interval (more precisely: are present inside the same trigger zone on two consecutive checks.)

62.2 Dependencies

Sweeper requires dcsCommon and cfxZones

62.3 ME Integration

Name	Description
sweeper	Marks this zone as a sweeper zone. No unit may remain longer than the interval (defined in the config zone) inside the zone or it risks being removed from the game **MANDATORY**
aircraft	If set to true, aircraft are included in the sweep and removed after being found inside the zone on two consecutive checks Defaults to true (aircraft are included in sweeps)
helos	If set to true, helicopters are included in the sweep and removed after being found inside the zone on two consecutive checks Defaults to true (helicopters are included in sweeps)

62.4 Demos

63 TACAN

63.1 Summary

Places TACAN nav aids inside trigger zones.

63.2 Dependencies

Tacan requires the modules dcsCommon and cfxZones.

Optionally: groupTracker

63.3 ME Integration

Name	Description
tacan	Tells DML that this trigger zone is a tacan zone. The value of this attribute specifies which faction owns the TACAN. Possible values • "0" or "neutral" • "1" or "red" • "2" or blue Defaults to "neutral" (TACAN is owned by neutral faction, listed for red and blue factions when TACAN GUI is enabled) **MANDATORY**
channel	The channel to use for the TACAN, e.g. "72". For randomization, supports lists and ranges, e.g. "12, 17, 42-47" Defaults to "1" (The channel 1)
mode	Either "X" or "Y" Mode "Y" usually is reserved for airborne TACAN (currently not supported with tacan zones) Defaults to "X"
callsign	Three-letter callsign for this TACAN station, e.g. "TCN". For randomization, supports lists, e.g. "TXN", "ABC", "XYZ" Defaults to "TXN"
onStart	When set to true (default) the TACAN is created on mission start. If set to false, you must provide a "deploy?" attribute to be able to deploy the TACAN while the mission is running. Defaults to true (deploy on mission start)
heading	Orientation (in degrees) of the physical unit that is placed in game when the TACAN is deployed. Has no influence on the TACAN's function. Defaults to 0 (oriented North)
rndLoc	When set to true, the TACAN is placed in a random location inside the trigger zone. Supports polygonal trigger zones. Otherwise, the TACAN is placed at the exact center of the trigger zone. Defaults to false (place at center of zone)
triggerMethod	DML method that tells when an input should trigger.

Name	Description
	Defaults to “change”
deploy?	Watchflag that triggers on triggerMethod and then initiates a deployment cycle. If preWipe is set to false, the previously deployed TACAN remains, otherwise it is removed before the new TACAN deploys. Defaults to <none>
preWipe	If set to true (default), the last deployed TACAN from this zone is removed (if it exists) before a new TACAN is deployed. Defaults to true (remove last spawn before deploy)
destroy?	Watchflag that triggers on triggerMethod and then removes the last previously deployed TACAN if it still exists. Defauts to <none>
c#	Output (direct). Is always set to the currently selected channel of the last spawned TACAN. When that TACAN is destroyed via destroy? Signal, the cannel is set to 0 (zero) Defaults to <none>
announcer	When set to true, any TACAN created with this tacan zone is announced with callsign, channel and mode to all players (when owning faction of the TACAN is neutral) or all players of the same faction that also owns the TACAN. Note that TACANS that are created at mission startup (onStart = true) are not announced. Defaults to false (no announcement)
trackWith:	When a TACAN is created, it is tracked with the groupTracker modules that are listed as value. Supports lists. Note that the groupTracker module must load before the tacan module in order to work correctly.

63.4 Demos

- Take on TACAN

64 Taxi Police

64.1 Summary

This module enforces a speed limit for player-controlled planes on tarmac, ramp and taxiways. Repeat offenders are actioned against.

64.2 Dependencies

TaxiPolice requires dcsCommon and cfxZones

64.3 ME Integration

TaxiPolice uses mainly a config zone named "*taxiPoliceConfig*" for most settings:

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
speedLimit	The speed limit for aircraft on all taxiways and tarmacs, set in m/s (meter per second, a.k.a. 'civilized units'). Recommended values are 10 (36 km/h) and 14 (50 km/h) Defaults to 14 (14m/s = 50 km/h = 27 knots)
triggerTime	Grace period (in second) after which a transgression is registered. If a pilot goes above the speed limit for less than that time, it is forgotten. Defaults to 3 (seconds)
leeway	The width in meters of the 'corridor' along a runway to which the runway 'no limit' bounds extend to each side. Thus, exceeding the speed limit just off the runway is allowed – in case the pilots veers off the runway slightly. To see each airfield's runway corridors, turn on verbosity, and switch to the F10 Map view. All runway corridors are marked on the map with a dotted black line Defaults to 5 (meters)
extend	The length in meters of the 'corridor' along the runway to with it extends in front of and behind the runway. That way, pilots who touch down too early or too late (and survive) don't get in more trouble than they already are. To see each airfield's runway corridors, turn on verbosity, and switch to the F10 Map view. All runway corridors are marked on the map with a dotted black line. Defaults to 500 (meters)
radius	The radius (in meters) around an airfield's location as defined in DCS's airfield data base. Aircraft inside this radius come under scrutiny if they are on the ground. To see each airfield's center, turn on verbosity, and switch to the F10 Map view. Each airfield's center is marked on the map with a dotted red line. Note that in many maps (e.g., Caucasus), that airfield's location coincides with the runup area of a particular

Name	Description
	runway, not the center of an airfield as one would have naively assumed. Defaults to 3000 (meters)
maxTickets	Number of tickets a pilot (not unit) may receive before TaxiPolice starts taking active retribution for offenses. Defaults to 3 (i.e. the fourth offense will be painful)
active	Initial state of taxiPolice when the mission starts up. When set to false, TaxiPolice is off duty until enabled. Defaults to true (taxiPolice is active at mission start)
greetings	When set to true, TaxiPolice will greet every pilot upon slotting or landing on an airfield, unless they are off duty. Pilots are given the current speed limit in km/h and knots. If an airfield is exempt from policing (as set with a 'taxiPolice' attribute at an airfield), that fact is also mentioned. Defaults to true – pilots are updated of the current speed limit and enforcement policies of their current airfield
onPatrol	Watchflag. Whenever the value of this flag changes, TaxiPolice becomes active on all policed airfields. A NOTAM is sent to all current players. Player's transgressions are recorded and sanctioned when they exceed the maxTicket limit. Note that turning taxiPolice on or off does not reset a player's violation count, only restarting the mission will do that. Defaults to <none>
offDuty	Watchflag. Whenever the value of this flag changes, TaxiPolice steps off the plate and ingnores any speed limit violations on the map. A NOTAM explaining that fact is sent to all current players. Note that turning taxiPolice on or off does not reset a player's violation count, only restarting the mission will do that. Defaults to <none>

Also, you can selectively exclude airfields from police oversight as follows:

Name	Description
TaxiPolice	When present, **the airfield closest to this trigger zone** is removed from TaxiPolice's list of airfields to monitor. Speeding on taxiways, ramps or tarmac areas is ignored. **MANDATORY**

64.4 Demos

- Taxi Police

65 TDZ (Touch-Down Zone)

65.1 Summary

A module to rate player's landings with regards to accuracy (touching down inside the TDZ) and giving statistics on landing run etc.

65.2 Dependencies

TDZ requires dcsCommon and cfxZones

65.3 ME Integration

Name	Description
TDZ	Marks this trigger zone as TDZ and use the center of this zone to determine which runway on the map to attach itself to. **MANDATORY**
verbose	When set to true, TDZ reports debugging information for this zone. This is particularly useful to discover the "Left" direction of an airfield's runways Defaults to false (verbosity off)
helos	When set to true, the Zone also reacts to helicopter landings. Defaults to false
extend	Extend the length of the runway (as reported internally by the map's DB) by this amount of meters. Note that the extension is added to both ends. Negative numbers shorten the runway on both end by this amount. Defaults to 0 (meters)
expand	Expands the width (i.e. widens) of the runway (as reported internally by the map's DB) by this amount of meters. Note that the width is added to both sides. Negative numbers make the runway narrower on both sides by this amount. Defaults to 0 (meters)
starts	Offset (in meters) from the runway threshold (as defined by DCS's airfield DB) where the TDZ starts Defaults to 0 (meters, directly at threshold)
ends	Offset (in meters) from the runway threshold (as defined by DCS's airfield DB) where the TDZ ends. Defaults to 610 (meters = 2000 ft)
landed!	Output to send a signal when an aircraft has successfully landed (touched down, did not leave the runway, and come to a complete stop on the runway (if 'opposite' is true – as per default – the direction of the

Name	Description
	landing is ignored) Defaults to <none>
touchdown!	Output to send a signal when an aircraft has touched down on the runway Defaults to <none>
failed!	Output to send a signal when an aircraft has touched down, and subsequently left the runway before coming to a complete stop (for whatever reasons, be they intentionally or an unscheduled rapid disassembly) Defaults to <none>
rwFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the runway. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to “0.0, 0.0, 0.0, 0.0” or “#00000000” transparent black
rwFrame	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the runway’s outline. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to “0.0, 0.0, 0.0, 1.0” or “#000000FF” opaque black
tdzFill	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to fill the TDZ. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to “0.0, 1.0, 0.0, 0.25” or “#00FF0040” transparent green
tdzFrame	Four numeric values, separated by comma (e.g., 1.0, 0, 0, 1.0) that define the RGBA (red, green, blue, alpha = opacity) values for the color used to draw the runway’s outline. RGBA values each range from 0.0 to 1.0 Alternatively, you can use the #RRGGBBAA format, where RR, GG, BB and AA are each hexadecimals (as is used in HTML/CSS to denote colors) Defaults to “0.0, 1.0, 0.0, 1.0” or “#00FF00FF” opaque black
visible	When set to true, TDZ draws a frame and fills the runway with the selected colors and then draws the TDZ(s) on top. Defaults to true (draw runway and TDZ)

Name	Description
left	When set to true, the ‘left’ direction (runway’s “main” landing direction, as reported by the map’s DB) of this runway is active, landings in this direction are reported, and the TDZ in this direction is active. Defaults to true
right	When set to true, the ‘right direction (opposite to the airfield’s reported ‘main’ landing direction) of this runway is active, landings in this direction are reported, and the TDZ in this direction is active. Defaults to true

65.4 Demos

- Landing Lessons

66 UnGrief

66.1 Summary

Module to deter griefers and to enforce PVE-only rules

66.2 Dependencies

dcsCommon, cfxZones

(SSB on server when using SSB as retaliation)

66.3 ME Integration

unGrief uses a config zone to control the module's main functionality

Name	Description
verbose	A value of "true" turns on debugging messages. Default is "false"
graceKills	Number of own faction ("friendly") kills that are permissible before unGrief retaliates. Set to 0 to disallow (and immediately punish) any friendly fire kills. Default is 1
retaliation	How unGrief retaliates towards the player when their graceKills are exceeded. The following options are supported: • 'boom' Place a small explosive inside the plane, grin, ignite. • 'ssb' Use server-side SSB to kick the player from the plane Default is 'boom'
wrathful	Sometimes griefers are slow learners. When set to true, unGrief disallows a repeat offender to re-slot after their second transgression. Their plane is destroyed every time they try to slot until the mission ends.
pve pveOnly	When set to true, PVE rules are in force for this mission. Player versus Player kills count like friendly kills Defaults to FALSE (PVP killing is allowed)
ignoreAI	Ignores friendly kills if the killed unit was AI-controlled (i.e. not a player unit). Useful in conjunction with PVE rules. Defaults to false.
warnings	When set to true, players entering and leaving a PvP zone receive a notification. Only works when PvE is set to true Defaults to true (notifications when entering/leaving a PVP zone)

You can designate areas as PvP combat zones (requires that pve is enabled)

Name	Description
pvp	The mere presence of this attribute marks this zone as a PVP combat zone. Killing players of another faction in this zone is legal. Same-faction kills are still illegal and will be punished.
strict	When set to true enforces ‘strict’ pvp rules, requiring both planes be inside the PvP zone at the time of the kill. When false, only the killed player plane must be inside the PvP zone. Defaults to false

66.4 Demos

- Good Grief
- The Danger Zone

67 Unit Persistence

67.1 Summary

Module that provides persistence (load & save) ability for ground units and static objects that are placed with ME (i.e. it does not supports units that are spawned dynamically)

67.2 Dependencies

Requires dcsCommon, cfxZones, persistence and cfxMX.

67.3 ME Integration

67.4 Demos

- Being Persistent

68 Unit Zone

68.1 Summary

Tests if a player unit or any member of an AI group is inside/outside a zone and can change flags when the status changes.

68.2 Dependencies

dcsCommon, cfxZones

68.3 ME Integration

Name	Description
unitZone	Marks this ME Zone as an anchor for unitZone. The value of this attribute defines which coalition groups/players are checked. Legal values are “0”, “1”, “2”, “red”, “blue”, “neutral”. Note that “**0” (zero) and ‘neutral’ means ‘both’:** unitZones never considers groups belonging to the neutral coalition. Defaults to 0 (both coalitions red and blue) **MANDATORY**
lookFor	Name for the group or (player) unit to check zone status. If the last character in the name is an asterisk “*”, exact matches and all group/unit names that start with that string (minus asterisk) are accepted, e.g. if you supply ”Hel*” all of the following would be accepted: • Hel • Hello World • Helo Rescue-1 • Hellfire If you want to match all groups or players, simply supply “*” (default) as this will match all names. Use this feature to your advantage in conjunction with cloners or spawners, as these all produce groups with a known base name. If you only supplied “Hel”,only (without the asterisk “*”) only the group whose name exactly matches “Hel” is checked. Defauls to “*” – meaning all names are matched.
matching	What type of units to match. Currently supported are • group (default): look for group names • player – look only at player units and match their unit’s (not group’s) name against lookFor Default: group
filterFor filter	Which unit categories to look for. If no attribute is given, **all** categories are checked against the zone (when their name pattern matches).

Name	Description
	When you supply a filterFor attribute, only that category is considered. Currently supported are • 0 (zero) or “aircraft” or “air” • 1 or “helo” or “heli” or “helicopter” • 2 or “ground” • 3 or “ship” • 4 or “train” Defaults to no filtering
enterZone!	Change this flag when the first unit (player) or part of all groups that match the criteria enters the zone Defaults to <none>
exitZone!	Change this flag when the last unit (player) of all groups that match the criteria have exited the zone (being destroyed counts as leaving) Defaults to <none>
changeZone!	Changes this flag whenever enterZone! or exitZone! are triggered Defaults to <none>
method uzMethod	DML Flag method for output. Use only one synonym per zone Defaults to “inc”
uzOff?	Watchflag. When triggered, this zone will no longer perform checks. When already off, nothing happens Defaults to <none>
uzOn?	Watchflag. When triggered, this zone will resume checks. When already on, nothing happens Defaults to <none>
triggerMethod uzTriggerMethod	Method that determines when the watchflags should trigger. Default is “change”
uzDirect uzDirect# direct#	Used mainly to control (directly enable/disable modules and open/close gates (changer modules)) When present, this flag (or flags) is always set to the current state of the unit zone: • 1 when one or more units in the zone • 0 when none of the indicated units in the zone. Default is <none>

68.4 Demos

- Follow Me!
- xFlags – Field Day
- Gate and Switch
- Forever-looping Spawners

69 Usher

69.1 Summary
This module creates events around players joining and/or leaving units

69.2 Dependencies
Usher requires dcsCommon and cfxZones

69.3 ME Integration
Usher is entiorely driven via its config zone

69.4 Demos
- Usher me to my plane

70 Valet

70.1 Summary

Valet is a module that helps you to easily generate events, text message or sound effects for players who enter or leave a zone like airfields, Hospital zones, FARPs, naval groups, destinations etc. Valet has strong built-in abilities to assemble complex text messages and can send these or sound to that player unit.

70.2 Dependencies

dcsCommon, cfxZones

70.3 ME Integration

Name	Description
valet	Tells DML that this trigger zone is a valet **MANDATORY**
greeting	Message to be displayed if a player unit triggers the valet. The message can be heavily customized with dynamic wildcards. Valet supports the following wildcards in the greeting, which are replaced with current values as the mission runs: • `<player>` - callsign of the player • `<unit>` - name of the unit that the player is controlling • `<type>` - type (e.g. “A-10A”) of the unit that the player is controlling • `<group>` - name of the group that the player’s unit is in • `<in>` - number of greetings that the player has received from this valet • `<out>` - number of goodbyes that the player has received from this valet • `<n>` creates a line feed • `<z>` - valet zone’s name as set with ME • `<t>` - current mission time in the format “HH:MM:SS”, e.g. “08:42:11”. Format be further configured with the *timeFormat* attribute. • `<lat>` - latitude of the valet zone’s current position • `<lon>` - longitude of the valet zone’s current position • `<ele>` - elevation of the valet zone’s current position. Format be further configured with the *imperialUnits* attribute. • `<mgrs>` - valet zone’s current position in MGRS coordinates • `<v: flagName>` - value of flag <flagName> • `<t: flagName>` - value of flag <flagName> interpreted as time. Format be further configured with the *timeFormat* attribute.

Name	Description
	• <lat: unit/zone> - latitude of unit/zone with that name. • <lon: unit/zone> - longitude of unit/zone with that name. • <ele: unit/zone> - elevation of unit/zone with that name. Format be further configured with the *imperialUnits* attribute. • <mgrs: unit/zone> - mgrs of unit/zone with that name • <coa: flag/unit/zone> - faction (e.g., "RED") of zone/unit, or the flag value interpreted as such. • <vel: unit/zone> - velocity of unit/zone with that name. Format be further configured with the *imperialUnits* attribute. • <alt: unit/zone> - altitude of unit/zone with that name. Format be further configured with the *imperialUnits* attribute. • <hdg: unit/zone> - heading of unit/zone with that name. • <type: unit> - type (e.g. "A-10A") of unit with that name • <player: unit> - player callsign who controls a unit with that name. • <twn: unit/zone> outputs the name of the nearest named location on the map. Requires the twn module • <loc: unit/zone> outputs position relative to nearest named map location. Requires twn module If the message text is empty, no message is displayed. Defaults to <empty message>, i.e., no greeting diosplayed
firstGreeting	When present, the text of this message replaces the message that is displayed when a player enters the valet for the first time since they (re)spawned. If the attribute is present but the text is empty, no message is displayed (i.e., this can be used to suppress the first greeting to a player) Supports all wildcards that *greeting* does. Defaults to <none, no separate first greeting)
goodbye	Message to display when a player leaves the valet zone Supports all wildcards that *greeting* does. Defaults to <empty>, i.e. no text message.
imperial imperialUnits	When set to true, wildcards return units for velocities, length etc. in "imperial" funny units (knots, feet, tablespoon etc.) Defaults to false (SI units are used: m, km/h)
timeFormat	The time format in which time values are displayed. Defaults to <:h>:<:m>:<:s>
duration	Time duration (in seconds) that a text message is displayed for Defaults to 30
inSoundFile	Name of the sound file that is to be played when a player enters the valet zone Defaults to <none>
firstInSoundFile	When present, this audio is played when a player enters the valet for the first time since they (re)spawned. Defaults to <none>, i.e., insoundFile is played

Name	Description
greetSpawns	Controls if a unit that spawns inside a valet zone should be greeted. If set to false, a unit that spawns within a valet zone will not receive a greeting, and must exit and then re-enter the valet zone for its first greeting. Default is false (spawning units inside a valet zone will not be greeted but have to leave zone first)
method valetMethod	DML method for outputs Defaults to 'inc'
hi!	Output flag to set when an eligible player unit enters the valet zone
bye!	Output flag to set when an eligible player unit leaves the valet zone
coalition valetCoalition	When given, restricts the valet to players of that coalition (e.g., 'red'). Players of other coalition are ignored. Defaults to <none> - all players are eligible
types valetTypes	A list of unit types (e.g., "A-10A"), separated by comma. The valet zone is restricted to players who control units of that type. List supports end-wildcards, i.e., when the type ends on an asterisk "*", all types that match everything before the asterisk are matches. Example: "A-10*" matches "A-10A", "A-10C" and "A-10C_2", but not "AV8BNA". Types are not case sensitive. Defaults to <none>, all types permitted
groups valetGroups	A list of group names, separated by comma. The valet zone is restricted to players who control units who are members of the groups on the list. List supports end-wildcards, i.e., when the type ends on an asterisk "*", all group names that match everything before the asterisk are matches. Example: "atta*" matches "Attackers-1" and "atta99" but not "avenger-1" Group names are NOT case sensitive. Defaults to <none> - no group restrictions
units valetUnits	A list of unit names, separated by comma. The valet zone is restricted to players who control units with names on the list. List supports end-wildcards, i.e., when the type ends on an asterisk "*", all unit names that match everything before the asterisk are matches. Example: "atta*" matches "Attackers-1" and "atta99" but not "avenger-1" Unit names are NOT case sensitive. Defaults to <none> - no unit restrictions

70.4 Demos

- I say hello goodbye

71 Willie Pete

71.1 Summary

A module to create artillery fire on locations marked with smoke munitions (WP – white phosphorous)

71.2 Dependencies

WilliePete requires dcsCommon, cfxZones and cfxMX.

Optional: PlayerScore (no load time limitation)

71.3 ME Integration

Name	Description
wpTarget	Marks this zone as a WP Zone. The value of this attribute defines which coalition uses this zone. Valid values are • RED or 1 • BLUE or 2 **MANDATORY**
shellStrength	Explosive power of shell hitting the ground. Defaults to 500
shellNum	Number of shells per artillery salvo Defaults to 17
transitionTime	The time (in seconds) it takes for shells to arrive after the 'fire' command is given Defaults to 20
coolDown	Number of seconds after which the artillery can fire again. Defaults to 180 (3 minutes)
baseAccuracy	Radius (in m) around the wp where the shells hit in. Defaults to 50
method wpMethod	DML method for the outputs
wpFired!	Output that is signaled when a successfully Fire command was received by the WP zone Defaults to <none>
checkInRange	Distance (from the zone's border) from which a plane can check into the zone. Example: if a WP zone has a radius of 3 km, and a checkInRange of 2 km, a plane can check into the zone when it's 5km (= 3 + 2) or less away from the zone's center. Be careful with check-in ranges when they overlap, as this may lead to confusing situations for pilots who may inadvertently sign into the wrong zone.
ackSound	Sound file to play after a player has successfully transmitted target coordinates. Overrides any setting of the wpConfig Defaults to what you defined in wpConfig

Name	Description
guiSound	Sound file to play whenever a player uses a menu item from the WP "FAC" menu. Overrides any setting of the wpConfig Defaults to what you defined in wpConfig

71.4 Demos

- Willie Nillie

72 Wiper

72.1 Summary

Removes objects inside a zone whenever you trigger it. Can be used to remove some detonation artifacts like debris, wrecks and craters.

72.2 Dependencies

dcsCommon, cfxZones

72.3 ME Integration

Name	Description
wipe?	Watchflag. Triggers a wipe cycle **MANDATORY**
triggerMethod triggerWiperMethod	Method that triggers the Watchflag
category wipeCat wipeCategory	List, separated by comma "," of object categories that are affected by the wipe (i.e. if they belong to the category they may be wiped). Possible values are • "unit" or 1 • "weapon" or 2 • "static" or 3 • "base" or 4 • "scenery" or 5 • "cargo" or 6 If a category is not recognized, it defaults to "3" (static) A special value 'none' can be used to skip object removal entirely. This can be useful if you are only interested in the 'declutter' (wreck and debris removal) ability. *Examples* • "`none`" (skips entire object removal step) • "`unit, 3`" (wipes units and static objects) • "`cargo`" (wipes cargo type objects) Defaults to 'none'
wipeNamed	*Optional* comma-separated name list that an object's name must match in order to be wiped. Supports an asterisk ("*") as wildcard to match anything. For example, "Ba*" would match "Base", "Ba", "Babushka", and "Bathyscape" **Examples:** • "Ba*" – all objects inside the zone whose name starts with "Ba"

Name	Description
	• "Grou*, Commander Kirk, He*" – all objects whose name starts with "Grou" or "He", and the object whose name exactly matches "Commander Kirk" Defaults to <option off>, no name filtering
declutter	If set to true will remove all debris, wrecks, craters and similar detonation artifacts from inside the zone. *Caution:* This option invokes DCS's `world.removeJunk()` method which is currently suspected of doing some nefarious stuff, including mission crashes in multiplayer. Suspected, not proven. Defaults to false (no debris removal)
wipeInventory	A Boolean that turns on the wiper's inventory function. Whenever triggered, the zone lists all objects, it finds inside the zone, sorted by category. Note that there may be objects inside a zone that wiper cannot find, and that it may return objects that are not really inside the zone. Both are a DCS limitation, not a bug in wiper.

72.4 Demos

- Viper with a double youu

73 XFlags (Flag Testing)

73.1 Summary

This requires that multiple input flags meet certain conditions to trigger the output flag. xFlags first applies the trigger method to evaluate input flags individually, and then requries that the individual results meet a condition to arrive at a final results. For example, it can be used to determine if ALL flags (requirement) have individual values of ">2" (individual trigger)

73.2 Dependencies

dcsCommon, cfxZones

73.3 ME Integration

Name	Description
xFlags?	A list (comma-separated) of input flags whose values should be evaluated to form the output signal **MANDATORY**
require	Condition/Operation that should apply to the input flags to form the output value. Currently supports the following conditions: • 'or', 'any', or 'some' triggers if at least one of the input flags has triggered • 'and' or 'all' triggers if all the input flags have triggered • 'more than' triggers if more than the value given in '#hits' of input flags have triggered • 'at least' triggers if #hits or more of the input flags have triggered • 'exactly' if triggers if the number of input flags that have triggered is equal to #hits • 'none' triggers if none of the input flags have triggered. Requires that you turn off one-shot mode • 'not all' or 'nand' triggers when not all input flags have triggered. Requires that you turn off one-shot mode

Name	Description
	• 'most' triggers when more than half of the input flags have triggered. Will not trigger is exactly have have triggered, so be careful if the number of input flags is even. • 'half or more' triggers when half or more of the input flags have triggered. Will also trigger if exactly half of all flags have triggered • 'never' used when you are using xFlag's "direct outputs" (xDirect, xCount, xChange) and want it to operate during the entire mission. xFlags will never trigger xSuccess, and keep evaluating, setting xDirect, xCount and xChange accordingly Defaults to 'some'
#hits	Value used for only some of the require attribute. Can be a value or flag name (in which case the value will be loaded from that flag). Numbered flags must be enclosed in double quotes, e.g. "22" to access flag 22. Defaults to 1 (one)
xFlagMethod	Condition that must be met for individual input flags. Is identical to trigger method for Watchflags except it is applied to each input flag individually. Defaults to "change"
xSuccess! out!	Flag to bang! when xFlags when the evaluation of input flags succeeds (all conditions are met). Once xSuccess is triggered, and unless oneShot is set to false, this zone's xFlag pauses until xReset is triggered. Defaults to <none>
xChange!	Flag to bang! when xFlags detects a change in the input configuration. Merely detects a change in the input configuration, has no relation with xSuccess!, except that xSuccess will also be accompanied by a bang on xChange! Defaults to <none>
xDirect	Each time xFlags evaluates the input flags, it directly sets the xDirect flag to the evaluation result (0 or 1) – this is different from what xSuccess may output, since that flag's value is dependent on the xMethod attribute. **This flags value is set directly, not via DML method.** Defaults to <none>
xCount	Each time xFlags evaluates the input flags, it directly sets the xDirect flag to the number of hits (positive test results from the individual flags tests). For example, if three tests of the input flags are successful, xFlags sets the value of this output to the number 3 **This flags value is set directly, not via DML method.** Defaults to <none>
xReset?	When the value of this input changes, the zone's xFlag module is reset, and evaluation starts afresh.

Name	Description
	Note: this input always reacts to a change in the flag's value Defaults to <none>
method xMethod	DML method for output flags Defaults to "inc"
oneShot	When the value if this attribute is false, that zone's xFlag module will not stop evaluating after it triggers xSuccess.
xOff?	Flag to turn the xFlag off, suspending it. When turned off, no processing of input flag occurs. The xFlag will still respond to xReset by loading a new zero state. **Note: this input always reacts to a change in the flag's value** Defaults to <none>
xOn?	Turns a suspended xFlag back on to resume processing. It resumes processing where it left off **Note: this input always reacts to a change in the flag's value** Defaults to <none>
xSuspended	Sets the initial state of xFlag. Setting it to true suspends the xFlag at mission start Defaults to false

73.4 Demos

- xFlags – Field Day
- Count Base's Blues

74 Module Name

74.1 Summary

74.2 Dependencies

74.3 ME Integration

74.4 Demos