Documentation

Dynamic Artillery Pieces (DAP)

Last update: v1.0

What is this

DAP is an *Arma 3* script that allows the Mission Editor (you) to create real (or virtual) artillery/mortar fire missions faster and smarter for one or multiple sides, using *Eden* marker's positions and an external fire missions list where you plan the caliber, ammo type, rounds, cycle of repetition and more. *DAP* doesn't change any original *Arma* AI behavior, saving server performance, and preserving the *Arma* integrity and compatibility with any mod.

DAP supports the scheduling of unlimited fire missions per side, each of which can be configured separately:

Per fire-mission:

1. Real or virtual Virtual is WIP
2. The side that owns the fire mission;
3. Potential target sectors (*Eden* markers);
4. How much weaponry you want in the fire mission;
5. What caliber these weaponry will be (Light, Medium, Heavy, Super Heavy);
6. Ammunition type (HE, Cluster, Smoke, Flare etc);
7. Control the volume of rounds;
8. How many cycle repetitions;
9. Triggers that will trigger the fire mission (trigger activation, timer, kill/destruction).

Global settings:

- Custom callsign for artillery side;
- Which pieces can use CommandChat to report (On/Off)
- Infinite ammunition (On/Off);
- Fire mission areas visible on the player map (On/Off) WIP
- Custom cooldown between cycles of fire mission repetition;
- Pre-defined whitelist of weaponry working (Arma, DLCs, RHS, CUP, etc);
- Pre-defined whitelist of ammunition working (Arma, DLCs, RHS, CUP, etc);
- Pre-defined blacklist of currently bugged vehicles;
- Pre-defined blacklist of currently bugged ammunition;
- Debug mode;
- Etc…

Creation concept: make use of artillery pieces practical and fast for multiplayer or single-player missions.

For multiplayer and single-player

DAP works for all game purposes, including multiplayer and single-player.

For Hosted and Dedicated servers

DAP was built for both server types.

Compatible with ACE, RHS, CUP, or any others

Yes! *DAP* has in its *fn_DAP_management.sqf* file a whitelist of vehicles and weapons that are considered Artillery Pieces, and this list contains assets from mods such as RHS and CUP that have already been tested with *DAP* and approved. The same occurs for each ammunition that these pieces employ.

Debug monitor and feedback available

DAP was built to make the Mission Editor life easier, so if you want to get to know the debug monitor now, check this out.

Video Tutorials

Soon. But subscribe to my channel to be notified on the day I release the DAP video tutorials: https://www.youtube.com/@thy1984

Arma 3, CSWR Tutorials
by thy1984
Playlist • 3 videos • 7 views

1 ARMA 3, CSWR: How to install
thy1984 • 92 views • 1 year ago

2 ARMA 3, CSWR: How to set spawn-points
thy1984 • 84 views • 6 months ago

3 ARMA 3, CSWR: How to create an army
thy1984 • 75 views • 2 days ago

On YouTube, a playlist to help you too.

VIDEO TUTORIALS

If you need an SQF editor

Sure, I’m using *Visual Studio Code* with this customs specific for *Arma 3*:
https://forums.bohemia.net/forums/topic/239960-vs-code-tutorial-how-to-config-vs-code-for-arma-3-2023/

If you need something simpler:
https://notepad-plus-plus.org/, install it and, when you open some script file, go to Notepad++ main menu, “Language” and select “C” as file language. That’s it.

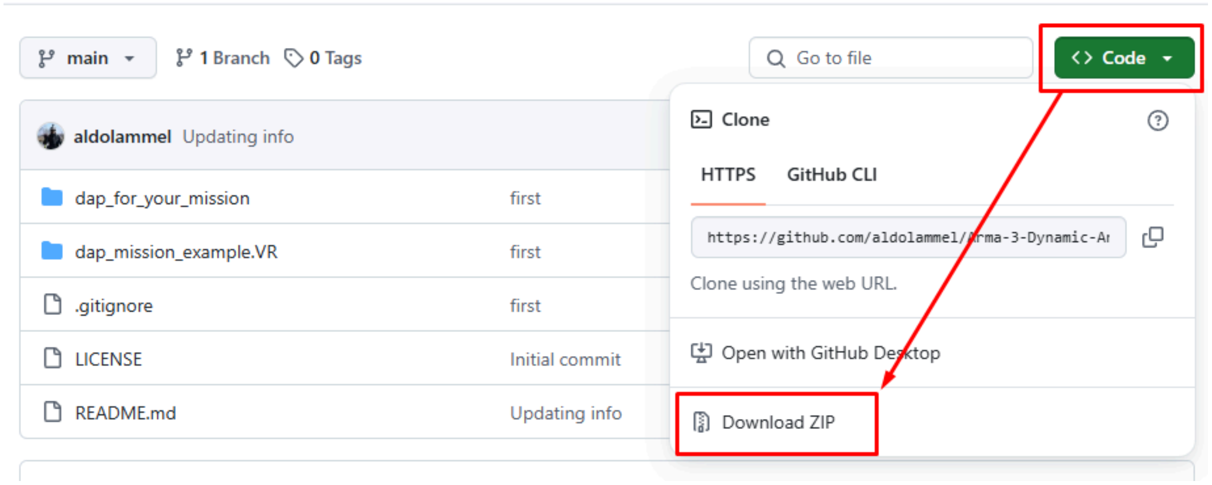
Run the script for a first look

1. Go to https://steamcommunity.com/sharedfiles/filedetails/?id=3371824030
2. Subscribe and wait for Steam to download it;
3. Open *Arma 3*, go to Multiplayer > Server browser > Host server > click Host Server button;
4. Select “Virtual Reality” map and, after that, select “*DAP* (...)”;
5. Play.

Important: if you want to apply this script in your own missions, check this out.

Install the script in my mission ★★★

1. Go to: https://github.com/aldolammel/Arma-3-Dynamic-Artillery-Pieces-Script
2. Download the zip and open it;



3. In zip, get in the "dap_for_your_mission" folder;
4. Copy all "dap_for_your_mission" content to your mission folder root: DRIVE:\Users\yourName\Documents\Arma 3\yourProfile\missions\yourMission\
5. WARNING: in the mission folder root, if you already have a "description.ext" file, don't replace the current file with the new one. Just add the code below in the current file:

```
class cfgFunctions
{
        // DAP: DYNAMIC ARTILLERY PIECES
        #include "DynamicArtilleryPieces\THY_DAP_functions.hpp"
};
```

6. WARNING: still in the mission folder, if you already have an "init.sqf" file, don't replace the current file with the new one. Just add the code below in the current file:

```
// DAP > HIDE THE SCRIPT MARKERS:
if ( !DAP_isOn || !DAP_debug_isOn ) then {{private _mkr = toUpper _x; private
_mkrChecking = _mkr splitString DAP_spacer; if (_mkrChecking find DAP_prefix
isNotEqualTo -1) then {_x setMarkerAlpha 0}} forEach allMapMarkers};
```

7. WARNING: finally, in the mission folder, if you already have an “initPlayerLocal.sqf” file, don’t replace the current file with the new one. Just add the code below in the current file:

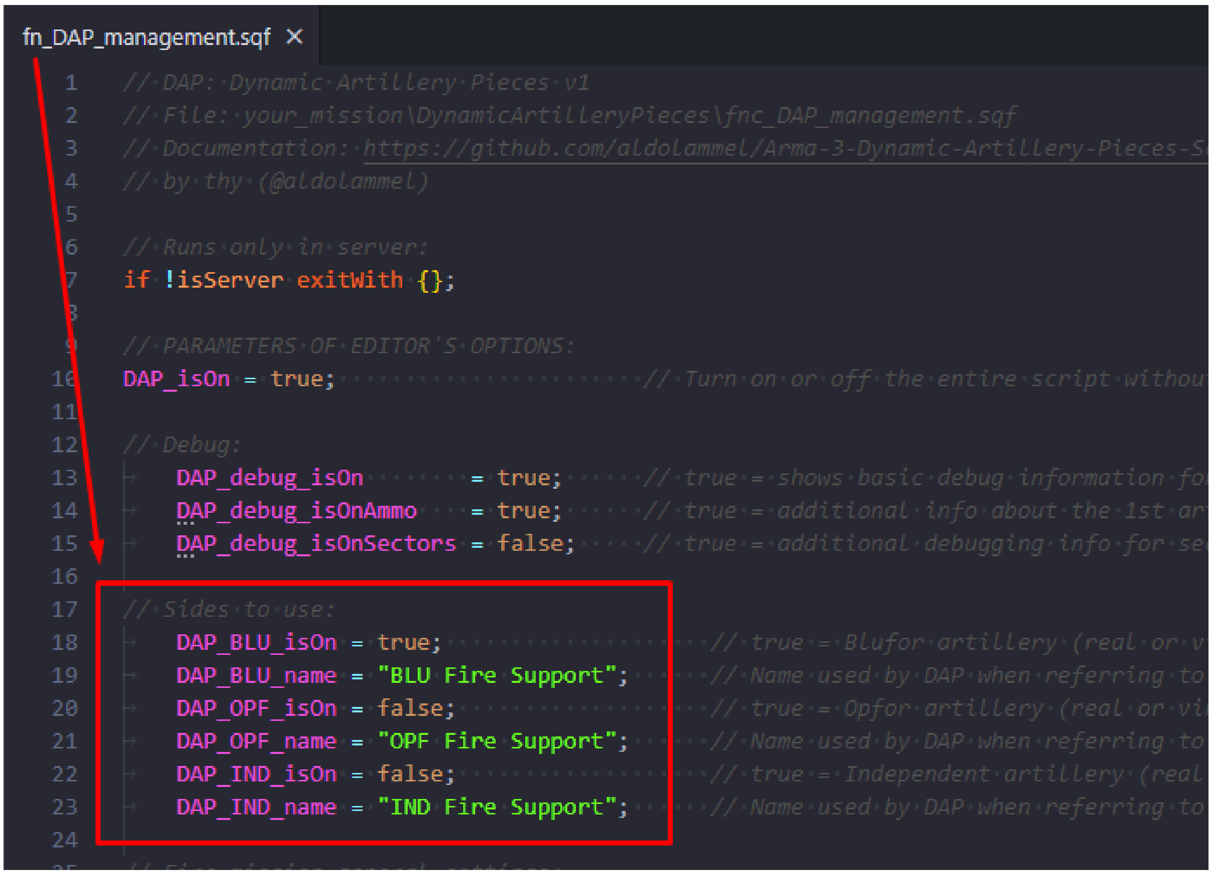
```
// DAP: DYNAMIC ARTILLERY PIECES
[player] execVM "DynamicArtilleryPieces\fn_DAP_playerLocal.sqf";
```

8. Now, let’s tell DAP what you want from it!

Choose which side(s) will use DAP

In *Arma 3* there are 4 sides available: Blufor, Opfor, Independent, and Civilian. Civilian is not an option for obvious reasons. That said, open the *DAP Management*:
\DynamicArtilleryPieces\fn_DAP_management.sqf

Set as *true* the side you want to build fire missions:



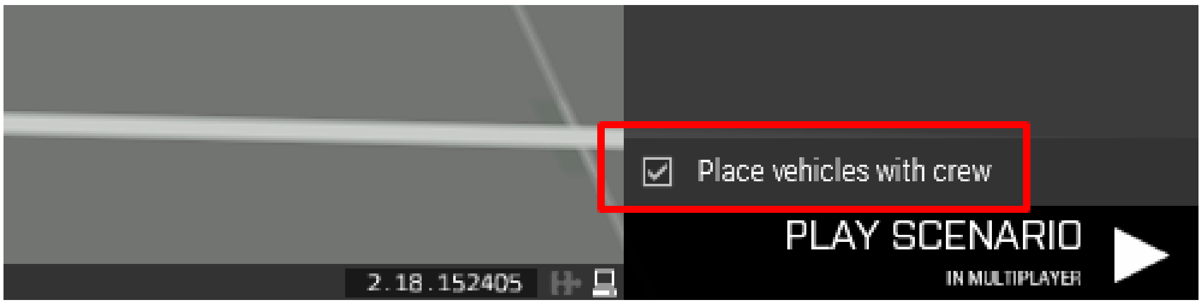
fn_DAP_management.sqf

```
// DAP: Dynamic Artillery Pieces v1
// File: your_mission\DynamicArtilleryPieces\fnc_DAP_management.sqf
// Documentation: https://github.com/aldolammel/Arma-3-Dynamic-Artillery-Pieces-S
// by thy (@aldolammel)

// Runs only in server:
if !isServer exitWith {};

// PARAMETERS OF EDITOR'S OPTIONS:
DAP_isOn = true;                        // Turn on or off the entire script withou

// Debug:
    DAP_debug_isOn        = true;      // true = shows basic debug information fo
    DAP_debug_isOnAmmo    = true;      // true = additional info about the 1st ar
    DAP_debug_isOnSectors = false;     // true = additional debugging info for se

// Sides to use:
    DAP_BLU_isOn = true;                       // true = Blufor artillery (real or v
    DAP_BLU_name = "BLU Fire Support";         // Name used by DAP when referring to
    DAP_OPF_isOn = false;                      // true = Opfor artillery (real or vi
    DAP_OPF_name = "OPF Fire Support";         // Name used by DAP when referring to
    DAP_IND_isOn = false;                      // true = Independent artillery (real
    DAP_IND_name = "IND Fire Support";         // Name used by DAP when referring to

```

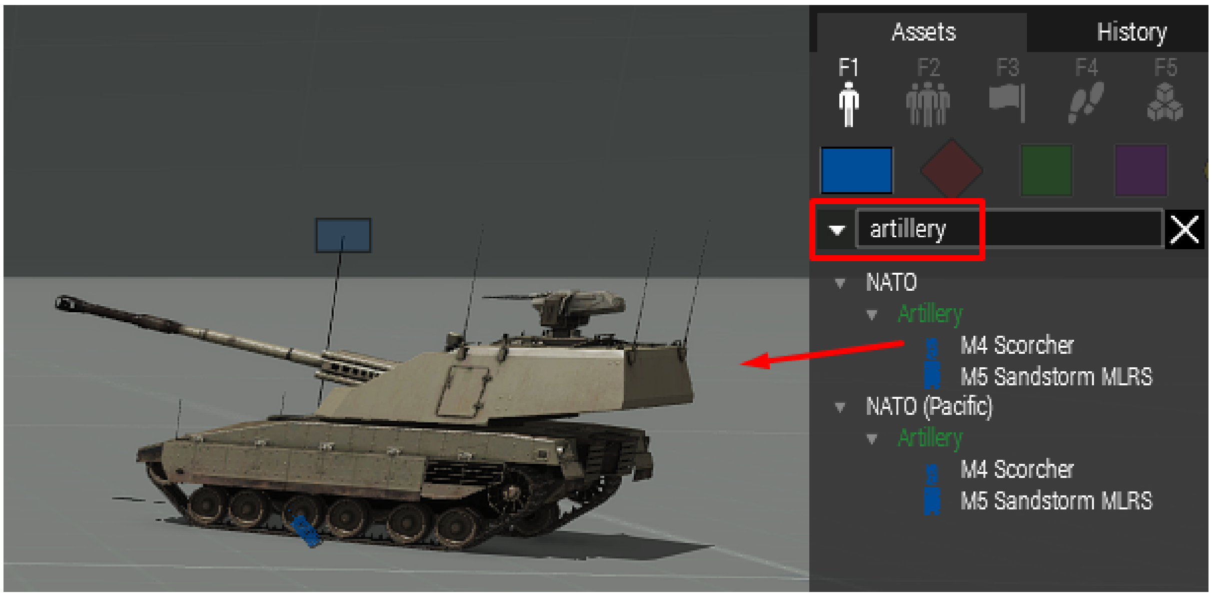
Select all artillery pieces you want

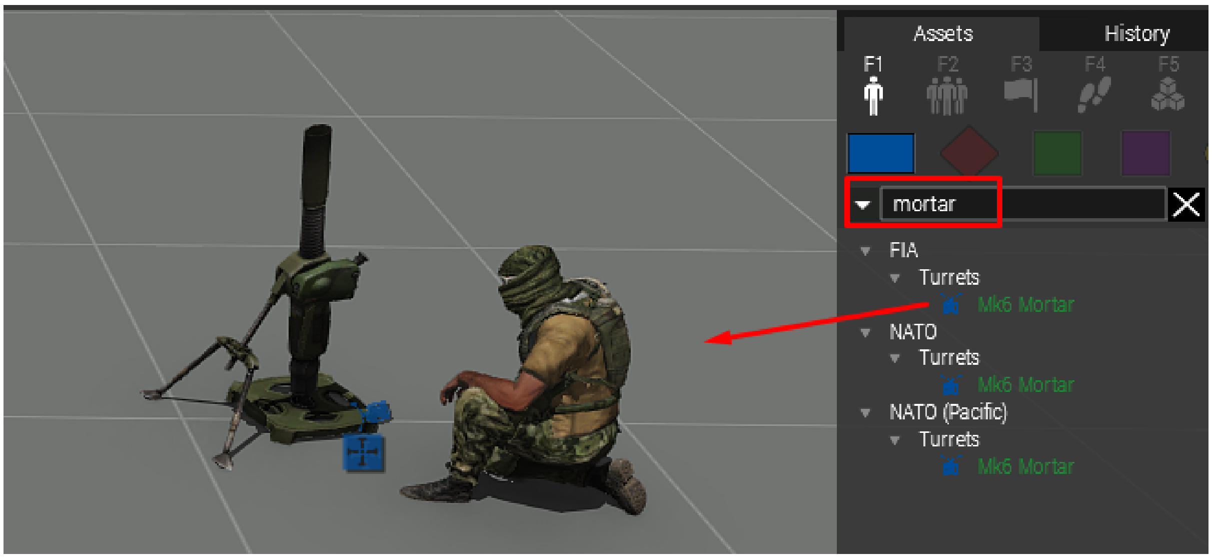
The next step is to set in your mission those vehicles and static/turret weapons (both are considered here “artillery pieces” or just “pieces”) with artillery or mortar capacities. But let’s start with just one to make the process easier.

Press “F” on Eden and keep the “Place vehicles with crew” checked:



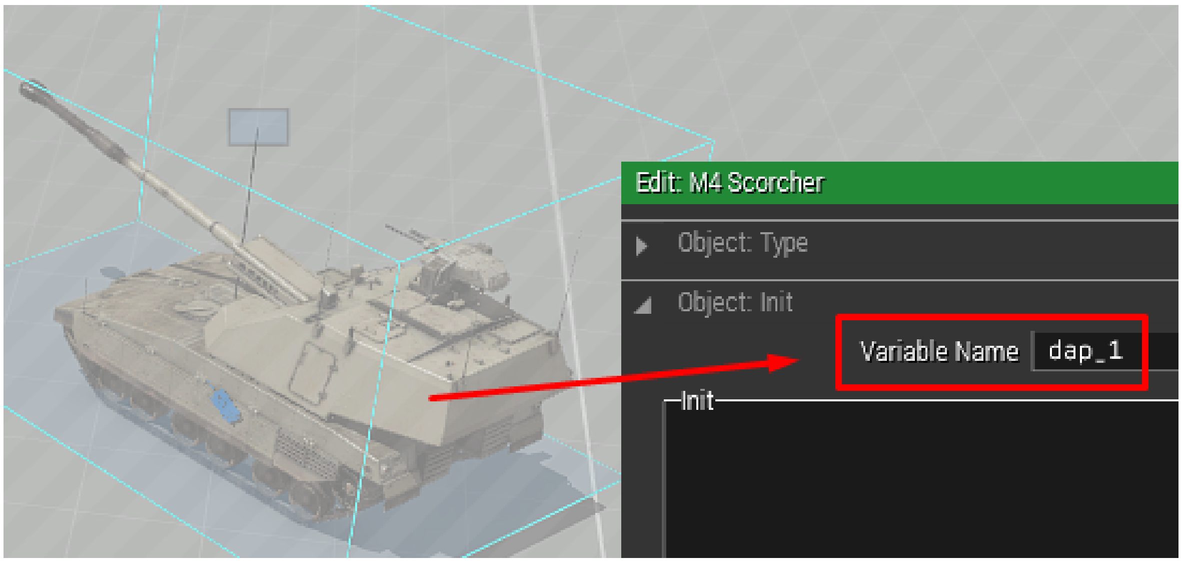
DAP doesn’t work with empty pieces.





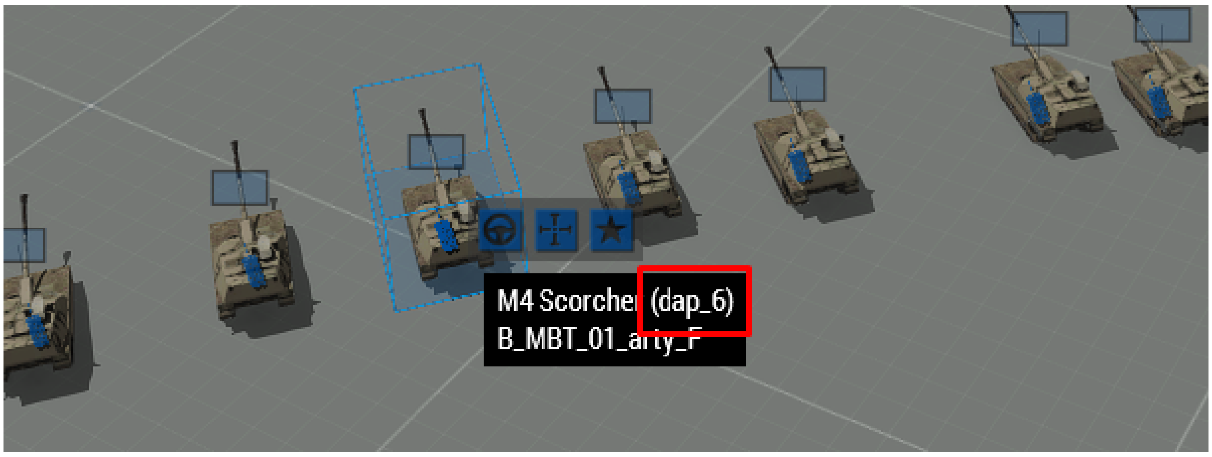
Open the piece attributes and add this name or whatever you wish since it brings “dap” in its variable name, for example:

- dap_1 (Recommended)
- an_dap_example
- an_example_dap

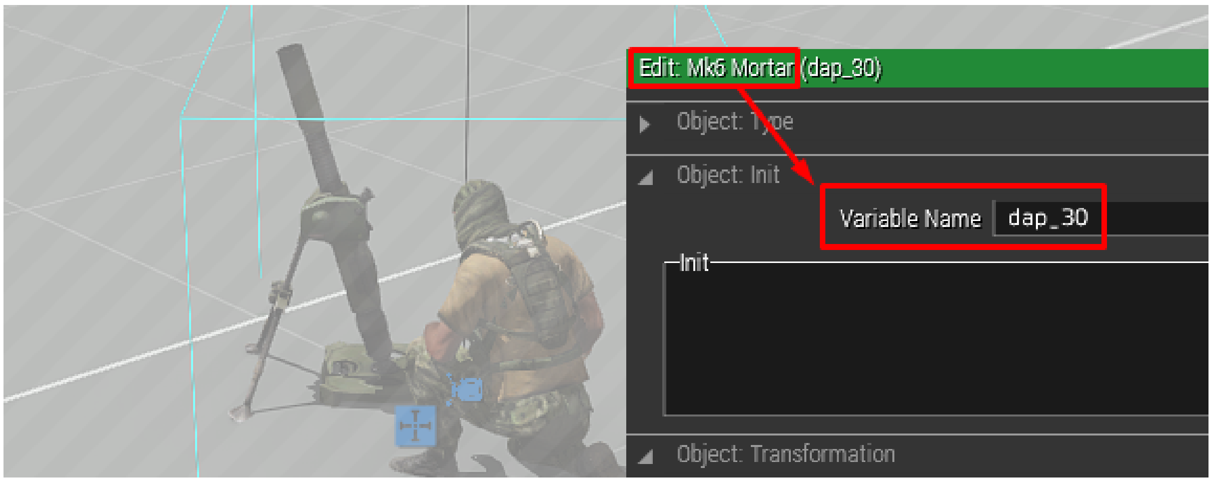


Notice the variable “dap_…” is always in the piece and NOT in crew members. Eden will create some for them.

Once you named one piece, just *Copy and Paste* how much you want because *Eden* will follow the *DAP* naming logic, adding a different number for each new copied piece (below).



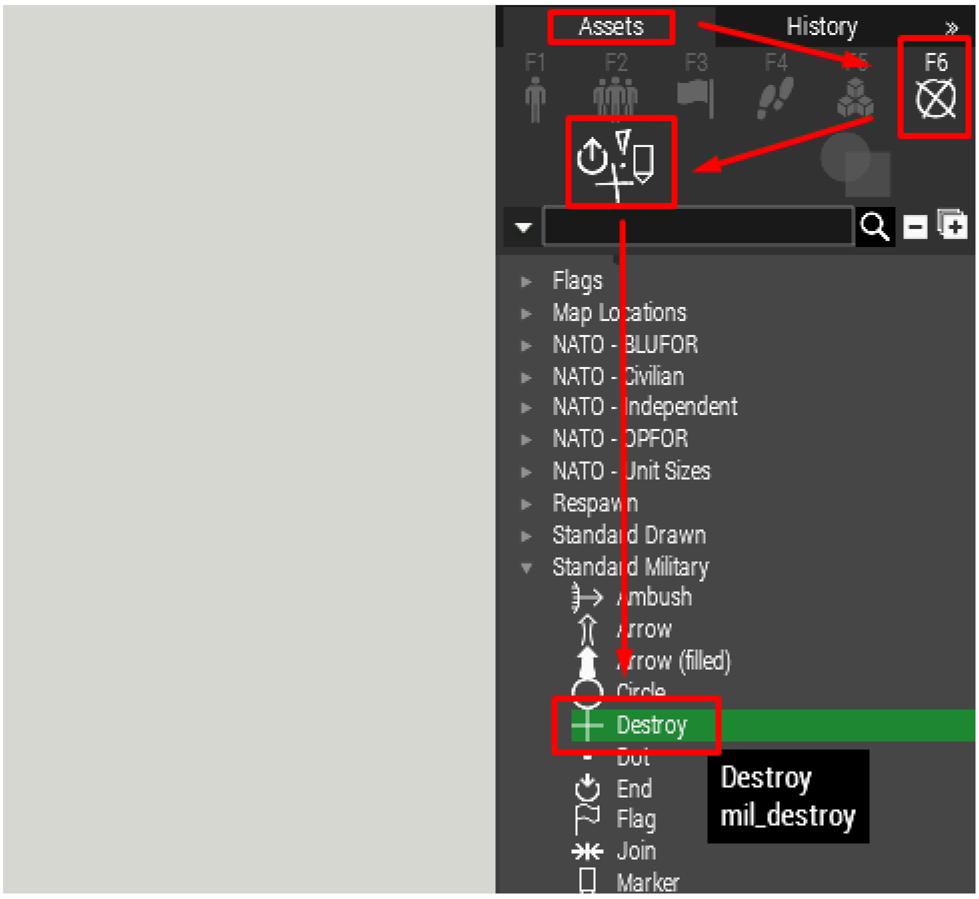
For another type of piece, again, create the first one and name it using a higher number to avoid duplicate (don't worry, *Eden* tells you if it happens):



Awesome, you got your fire support team! Let's set some targets now.

Defining targets

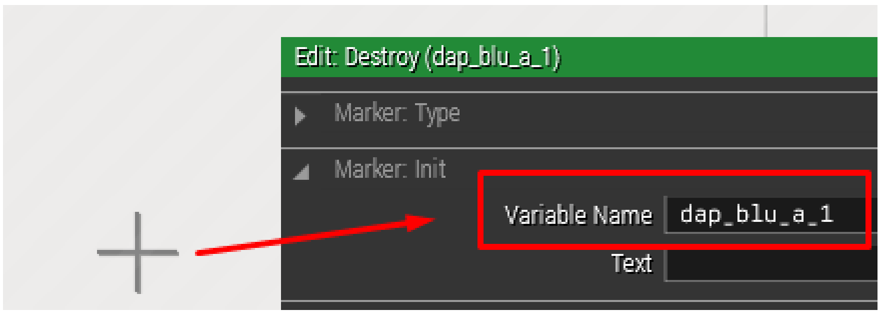
Targets are *Eden* markers you will *Drag and Drop* in your mission. You will always use the "*Destroy*" marker for *DAP*, never another type.



Above, the Asset-Browser on *Eden Editor*.

Unlike piece naming, targets need to receive the side tag and a sector letter. Sounds weird but it's easy. Check these examples:

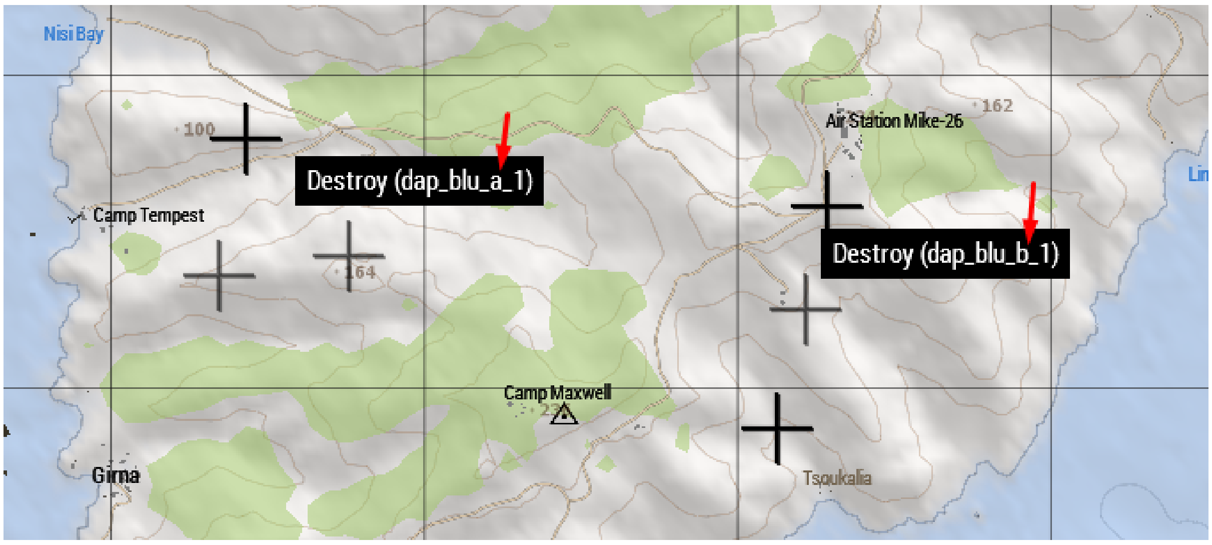
- dap_blu_a_1
- dap_blu_a_20
- dap_blu_z_1
- dap_blu_z_999



Variable Name: dap_blu_a_1

Above, editing a target-marker attributes on *Eden*. The name is telling DAP this target belongs to the BluFor fire teams and they calling this target sector "A".

Now, *Copy and Paste* how many targets "A" you want to give for BluFor attacks. After that, if you want, create some targets for sector "B" and so on, if needed.



Important: if you set a fire mission to hammer sector "A", *DAP* will look for artillery pieces you placed on the mission that better fit that request. Once *DAP* has the team to execute the fire mission, *DAP* will select randomly just one of those target markers of the sector "A". Use how many target markers you want. It doesn't affect any server performance.

Perfect! Before we jump into the coolest part, you need to set a trigger for each fire mission you have in mind.

Defining triggers

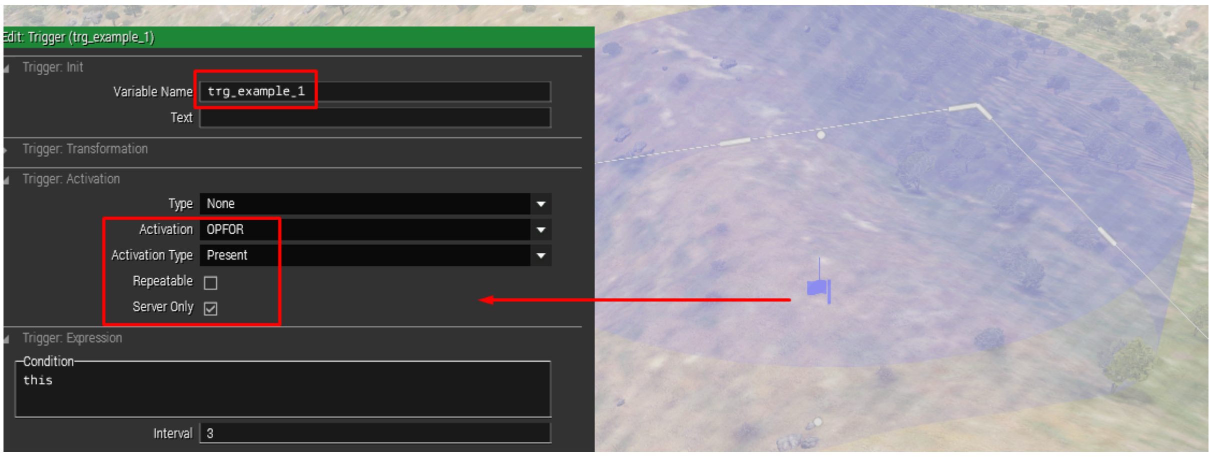
A fire mission doesn't take place out of the blue. You always need to tell *DAP* (NO EXCEPTION) what should initiate bombs being dropped from the sky.

There are 3 methods:

- by *Eden* trigger;
- by killing or destroying a target (unit, vehicle, building);
- by timer;
- and also you can combine them.

Fire mission released by Trigger

As any other trigger. You define what conditions will activate it. You only need to set a varname on it to use later.



Edit: Trigger (trg_example_1)

Trigger: Init

Variable Name	trg_example_1
Text	

Trigger: Transformation

Trigger: Activation

Type	None
Activation	OPFOR
Activation Type	Present
Repeatable	☐
Server Only	☑

Trigger: Expression

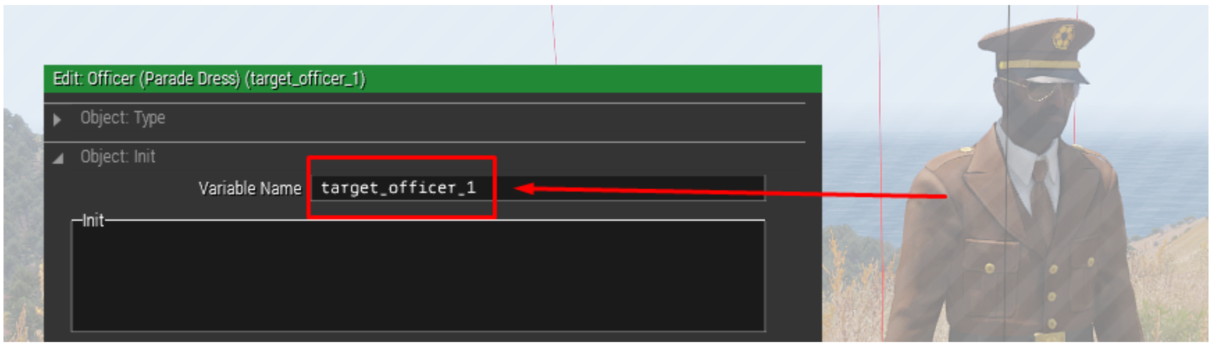
Condition

```
this
```

Interval: 3

Fire mission released by Target

You could start a fire mission if a unit dies or, for example, a vehicle blows up. Even a breakable structure collapse would release a fire mission with this method.



Edit: Officer (Parade Dress) (target_officer_1)

Object: Type

Object: Init

Variable Name: target_officer_1

Init

Fire mission released by Timer

This method you do directly in the fire missions file you will see next.

Combining all of them

And the combination you do also directly in the fire missions file.

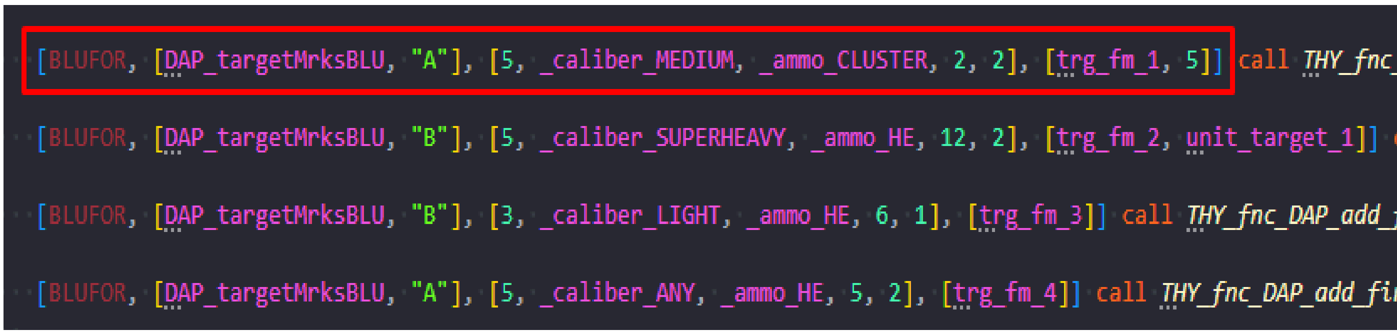
Okay, so let’s do it!

Defining fire missions

And then the fire missions, finally. Open your Fire Missions file:

\DynamicArtilleryPieces\fn_DAP_fireMissions.sqf

This row is literally a fire mission. You can create fire missions as much as you want or the server supports with no performance drops.



```
[BLUFOR, [DAP_targetMrksBLU, "A"], [5, _caliber_MEDIUM, _ammo_CLUSTER, 2, 2], [trg_fm_1, 5]] call THY_fnc_
[BLUFOR, [DAP_targetMrksBLU, "B"], [5, _caliber_SUPERHEAVY, _ammo_HE, 12, 2], [trg_fm_2, unit_target_1]]
[BLUFOR, [DAP_targetMrksBLU, "B"], [3, _caliber_LIGHT, _ammo_HE, 6, 1], [trg_fm_3]] call THY_fnc_DAP_add_
[BLUFOR, [DAP_targetMrksBLU, "A"], [5, _caliber_ANY, _ammo_HE, 5, 2], [trg_fm_4]] call THY_fnc_DAP_add_fi
```

Let's understand what each column of a fire mission row:

Side	Target markers and sector	Num of pieces involved, caliber of them, ammo type, rounds per piece, and repetition cycle	Triggers
That side owns it.	Each side has just one target marker label, e.g. "DAP_targetMrksBLU". Target segmentation is done through the use of a letter which is called a 'sector letter'. It means this fire mission will only consider targets from sector "A".	**Number of pieces involved:** You can request more than one artillery piece, but if you don't have enough in-game, *DAP* will use just those available that fit all fire mission requirements. **Caliber:** In *DAP* the pieces are chosen by their calibers. If you ask for heavy caliber, *DAP* will investigate what howitzers, MRLs, and/or mortars fit with heavy calibers and are/still available in the mission. **Ammo type:** If you request High Explosive ammo, *DAP* will filter the options, looking for just those pieces with this ammunition type. **Rounds per piece:** You got it. **Repetition cycle:** Hope you got it too.	Ways to release a fire mission. You must use at least one method for each fire mission: **Trigger** **Target** **Timer** (in minutes) If you combine methods, to release the fire mission just one of them needs to be reached, not all.

Testing your fire missions

But before that, I do advise you to check these pocket guides right below to make your fire mission planning faster.

Pocket guide: Artillery Piece Calibers

Soon.

<table>

Pocket guide: Ammunition Magazines

Soon.

<table>

Fixing: a piece or ammunition is not recognized by DAP

Soon.

Fixing: some artillery pieces are frozen

Soon..

Fixing: investigate further what is happening (Debugging)

Soon.

Contribute to the *DAP* script

Discussion on Bohemia Forums

https://forums.bohemia.net/forums/topic/290962-release-dynamic-artillery-pieces-dap/

Changelog on GitHub

https://github.com/aldolammel/Arma-3-Dynamic-Artillery-Pieces-Script?tab=readme-ov-file#changelog

Author

Based in Porto Alegre, Brazil

thy @aldolammel

If you care

Just give a like for *DAP* on Workshop to spread the word :)

https://steamcommunity.com/sharedfiles/filedetails/?id=3371824030