FHQ Task Tracker
V4.0

1 Introduction

Tracking tasks and briefing in a mission, especially a multiplayer mission with the potential of team switching and join-in-progress, is a tedious task if done manually. Arma 3 added modules for this, but they have their limitations (like, it's not really possible to re-arrange entries, you have to add them in a certain order).

FHQ TaskTracker was a solution I came up with for this. It started out as a script for Arma 2 and was adapted to Arma 3 later, first as a standalone script, and now finally as a function library.

2 Using FHQ Task Tracker

Since Task Tracker is now a function library, it has to be included into your missions function library in `description.ext` in class `CfgFunctions`. All of the Task Tracker is implemented inside `fhq_tasktracker.hpp`:

```
class CfgFunctions
{
   // … your own declarations go here

   #include "functions\fhq_tasktracker.hpp";
};
```

Initialization is handled automatically before the mission init, so there is no need to add any additional script call to `init.sqf`, and the functions are available immediately, even during mission initialization (in triggers, for example).

All functions names start with `FHQ_fnc_tt*`. For an overview of all available functions, open the function viewer in Eden or during the mission, and select missionConfigFile in the topmost select box, FHQ in the second one, and TaskTracker in the third one. There is an additional section, TaskTrackerInternal, that list internal functions that are used by the

normal interface functions. However, you should not be calling those functions directly, since they are subject to internal changes.

Most of the functions start with a comment block that outlines their use.

With the introduction of Eden, the 3D editor, FHQ Task Tracker has been updated with additional functionality. You can now edit briefings and task without the need for scripting. For more information on how to use FHQ Task Tracker with Eden, refer to chapter 7 of this guide.

3 Filters

Filters determine which units will receive what tasks or briefing entries. The functions `FHQ_fnc_ttAddTasks` and `FHQ_fnc_ttAddBriefing` both accept a list of task or briefing entries. These list entries can either be appropriate entries, or filter entries. A filter entry will determine which units will receive the subsequent entries.

For example, a filter entry might be a group object. If the Task Tracker encounters such a group object in the list, all subsequent tasks and briefing entries will only be given to the members of the group.

Example:

```
[
      _myGroup,
         […], /* 1 */
         […], /* 2 */
      _otherGroup,
         […]  /* 3 */
] FHQ_fnc_ttAddBriefing;
```

The lines `_myGroup` and `_otherGroup` are filter entries. Assuming they are variables holding two different group objects, the entries in lines 1 and 2 are given to members of group `_myGroup`, while entry 3 will only be given to members of `_otherGroup`.

A filter can be one of the following:

- Single object: If a single object is given, only this particular unit will receive the following entries.
- Group: All units in the given group will receive the following entries.
- Side: All members of the given side will receive the following entries.
- Faction (string): All members of the given faction will receive subsequent entries.
- Code: You can also use script commands to select which units will receive the entries. The code is called with all playable units in turn. The unit is passed to the code as the `_this` variable. The following entries are added to all units for which the piece of code returns `true`.

Examples:

```
[
  p1, /* The unit called p1 will receive the following */
      […],
      […],
  group man2, /* All members of man2's group */
      […],
      […],
  west, /* All BLUFOR units */
      […],
  “BLU_F”, /* All NATO units */
      […],
      […],
  {(side _this) != west}, /* All units except BLUFOR */
      […]
] call FHQ_fnc_ttAddBriefing;
```

4 Defining Briefings

A briefing entry always consist of an array of two or three strings. If two strings are given, the first is the title, the second one is the briefing text, and the entry is put in the “Diary” section of the map/briefing screen.

If three strings are given, the first will be the section name, second the title, and third the briefing text. In this form, the entry is put into a new section with the given name. The actual briefing text is structured text and can contain formatting, images and links.

Briefings are added by calling `FHQ_fnc_ttAddBriefing`. Note that this functions MUST be run on the server. Executing it on a client will have no effect. The only input to the function is a filtered array as described in the previous section.

The function can be called multiple times, at the start or during the mission.

Example:

```
[
  west,
    [“Mission”, “Ambush the CSAT supply convoy”],
    [“Weather”, “Overcast skies with 50 % chance of
               + “ rain”],
  “BLU_F”,
    [“Intelligence”, “Here are some details:...”],
  {true}, /* Visible to everybody */
    [“Credits”, “Mission by”, “Some dude”],
    [“Credits”, “Scripting”, “Some more dudes”]
] call FHQ_fnc_ttAddBriefing;
```

The first two lines will be under “Diary” for all BLUFOR units, the third line will be available to NATO only, also under the “Diary”, while the last two lines will be visible under a custom section “Credits” to everyone in the game.

5 Tasks

5.1 Anatomy of a task entry

Task entries are much more complex than briefing entries. They can also be “interacted” with. Therefore, every task that you define must have an identifier. This is simply a name for the task, defined as a string. “`taskAssault`” is a valid task name.

Each task entry is also associated with a description. The description is three parts (all of them strings), a title (that is shown in the task list), a long description (the part that will be visible after you clicked on the task in the list), and a short title that is usually shown next to the task marker on the map or in the 3d view (if they are enabled in the 3d view).

Note: It looks as if the short name is no longer used in Arma 3 1.58.

A task can also have a target. This can either be a predefined position (a marker, for example), or an object (for example, a tank the players have to destroy).

Finally, since Arma 3 1.58, a task can also have a type (as defined in `CfgTaskTypes`).

A task can be on it's own, or it can be grouped with other tasks under a common parent task. Parent tasks have to be defined before the child tasks can be defined.

Finally, a task always has a state. This state can be either “created” (the task was just created), “assigned” (the current task assigned to the player), “succeeded” (the task was successfully finished), “failed” (the task was unsuccessful and can no longer be achieved), or “cancelled” (the task was simply invalidated before it could be finished). The default for any task is “created”, meaning the task is unfinished and not the current task

A full task entry is an array of all or some of this information.

The first entry is always the task identifier. This is either a string, or, if the task has a parent, an array of two strings (the first being the new task, and the second the parent). So for example, `[“taskAssault”, …]` or `[[“taskDestroy1”, “taskDestroy”], …]` are correct task identifiers. The second, third and fourth entry are string defining the long description, title and short title. If the task has a target, it is listed as the fifth entry in the array, either as a position (a two or three element array), or an object. The sixth entry is the initial state. This can be any of the task states (in theory, you can start a task as cancelled, if it makes sense for you). Finally, the seventh element is the task type (only in Arma 3 1.58, this parameter will be ignored in previous versions).

A full task entry therefore, looks like this:

```
[_taskId,_description,_title,_shortTitle,_target,_initial,_type]
```

The entries `_target`, `_initial` and `_type` are optional. In fact, if the task does not have a target, `_target` is simply omitted (although `_initial` and `_type` can still be present). If `_initial` is omitted, it will automatically be assumed to be “created”. If `_type` is omitted, it will be assumed to be “default”.

5.2 Task Entry Examples

The following examples are valid task entries. This is a simple example, the smallest possible size:

```
["taskAssault", "Your team is tasked with assaulting the enemy
position at <marker name="markAssault">Kavala </marker>.",
"Assault Kavala", "ASSAULT"]
```

Example for a task with parent:

```
["taskDestroy", "Destroy the two fuel dumps", "Destroy Fuel
Dumps", "DESTROY"],
[["taskDestroy1", "taskDestroy"], "Destroy fuel dump #1",
"Destroy #1", "DESTROY", fuel_dump_1],
[["taskDestroy2", "taskDestroy"], "Destroy fuel dump #2",
"Destroy #2", "DESTROY", fule_dump_2]
```

Finally a full task with all elements present:

```
["taskDefend",
 "Defend the <marker name="markFOB">FOB Clearwater"
+"</marker> from the approaching CSAT troops",
 "Defend Clearwater",
 "DEFEND",
 getMarkerPos "markFOB",
 "assigned",
 "defend"]
```

5.3 Defining tasks

Similar to briefings, tasks are defined by calling `FHQ_fnc_ttAddTasks`. Again, the only input is a filtered array of task entries. Tasks that are parents of other tasks must be given before the child tasks. The function must be run on the server, running it on a client will have no effect.

The function can be called multiple times. The tasks of subsequent calls are added to the current set.

Example:

```
[
  west,
   ["taskAssault", …],
   [["taskAssault1", "taskAssault"], …],
   [["taskAssault2", "taskAssault"], …],
  east,
   ["taskDefend", …]
] call FHQ_fnc_ttAddTasks;
```

5.4 Setting task state

Once tasks are defined, their current state can be updated. At the heart of the task state

are two functions, `FHQ_fnc_ttGetTaskState` and `FHQ_fnc_ttSetTaskState`.

`FHQ_fnc_ttGetTaskState` takes an array with one element, the task identifier, as it's input, and returns the task's current state as a string (created, assigned, succeeded, failed, cancelled).

`FHQ_fnc_ttSetTaskState` takes an array with two elements as input, the task identifier, and the new state. It will set the task's state to the new one, regardless of it's previous state. That means, it is possible to, for example, set a task state from failed to succeeded (if that makes sense for your mission is yours to decide). If the previous and new state are different, it will also bring up a notification for the user.

A few additional functions exist for convenience. For their full usage, see the function browser. We'll list the functions here with a short description of what they do, to give you an idea about what is available to you.

- `FHQ_fnc_ttIsTaskCompleted`: This checks whether the given task is completed or not, i.e. it has a state of succeeded, failed, or cancelled.
- `FHQ_fnc_ttAreTasksCompleted`: Like above, but checks an array of tasks.
- `FHQ_fnc_ttIsTaskSuccessful`: This checks whether the given task is successfully completed, i.e. is has a state of succeeded.
- `FHQ_fnc_ttAreTasksSucecssful`: Like above, but checks an array of tasks.
- `FHQ_fnc_ttGetAllTasksWithState`: For the given state, return an array with all tasks that have that state.
- `FHQ_fnc_ttSetTaskStateAndNext`: Set a task to the given state, then selects the first task from the rest of the input array that is not yet completed and make it assigned. This function is pretty useful since it will allow you to automatically select a new task.

Examples:

```
// Check if "taskAssault" is complete
if (["taskAssault"] call FHQ_fnc_ttIsTaskCompleted) then {
        hint "yes";
};

// Check if "taskAssault" and "taskDefend" are successful
_res = ["taskAssault", "taskDefend"] call
                        FHQ_fnc_ttAreTasksSuccessful;

// Mark "taskAssault" as failed and select a followup
["taskAssault", "failed", "taskDefend", "taskExfiltrate"]
                  call FHQ_fnc_ttSetTaskStateAndNext;
```

6 Customizing notifications

By default, FHQ Task Tracker uses notify templates “TaskCreated”, “TaskAssigned”,

“TaskSucceeded”, “TaskCanceled”, and “TaskFailed” for showing task states of new, assigned, successful, cancelled or failed tasks, respectively.

For new briefing entries, it uses TaskAssigned by default, unless your description.ext contains an entry NewBriefing in CfgNotifications. An example for such an entry is given below:

```
class CfgNotifications{
 class NewBriefing {
 title = "BRIEFING UPDATED";
 iconPicture = "\a3\ui_f\data\IGUI\Cfg\Actions\talk_ca.paa";
 description = "%2";
 priority = 7;
 };
};
```

7 Eden Integration

7.1 Introduction

FHQ EdenTT is an addon for Arma 3 that allows mission makers to edit their briefings directly inside Eden. No scripting is required for this (although the actual usage of tasks in game still requires it), so this method is very well suited for both beginners and people that prefer comfort over minute control. Not every feature of FHQ Task Tracker can be used from EdenTT, but for almost all applications the amount of control is sufficient.

EdenTT introduces two new concepts - briefings and task groups. These are a replacement for filters used in the scripted TT.

EdenTT is an editor-only addon. This means that it is ONLY required during editing the mission. When the mission is played, EdenTT need not be loaded. You need to have added the task tracker to your mission, however, as was described in section 2 of the manual.

7.2 Briefings and Task Groups

A *briefing* in EdenTT is a collection of briefing headings and associated content, to be displayed under the "Diary->Briefing" section. Briefings have an associated name to be able to reference them. Each playable unit can have one briefing assigned to him, and the same briefing can be assigned to any or all units. If a unit has a briefing assigned to it, then it will see this briefing (i.e. the associated briefing headers) in game.

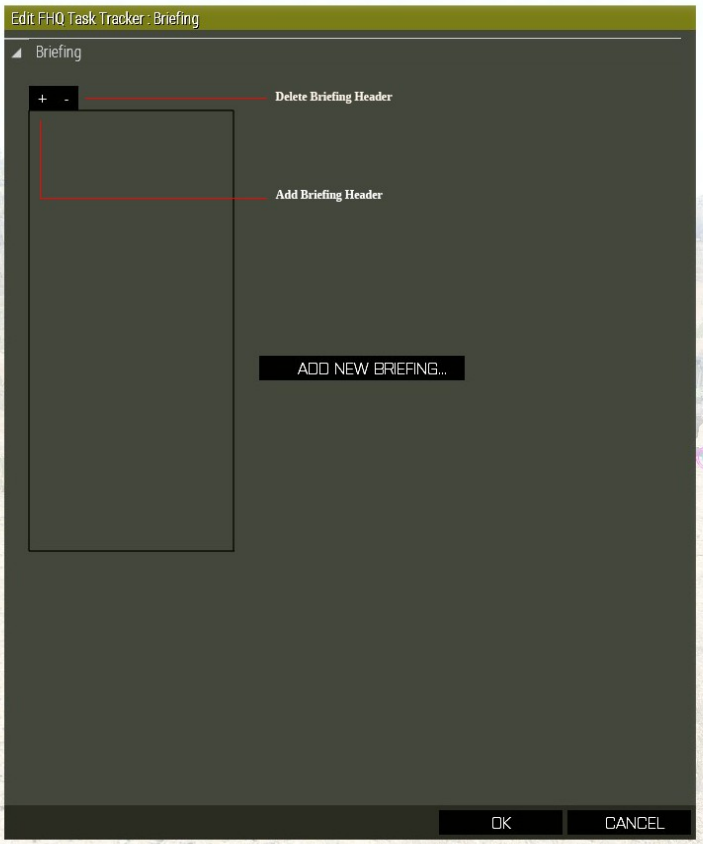
Consequently, a *task group* is a collection of tasks combined under a name. Like with briefings, task groups can be assigned to units, but unlike briefings, a unit can have any number of task groups assigned to it. The unit will see tasks from all task groups that it has been assigned. Currently, it is not possible to assign a task group dynamically; this is planned for a future update.

7.3 Using EdenTT

Once in Eden, the EdenTT dialogs are located unter the Attribute menu:

7.3.1 Editing Briefings

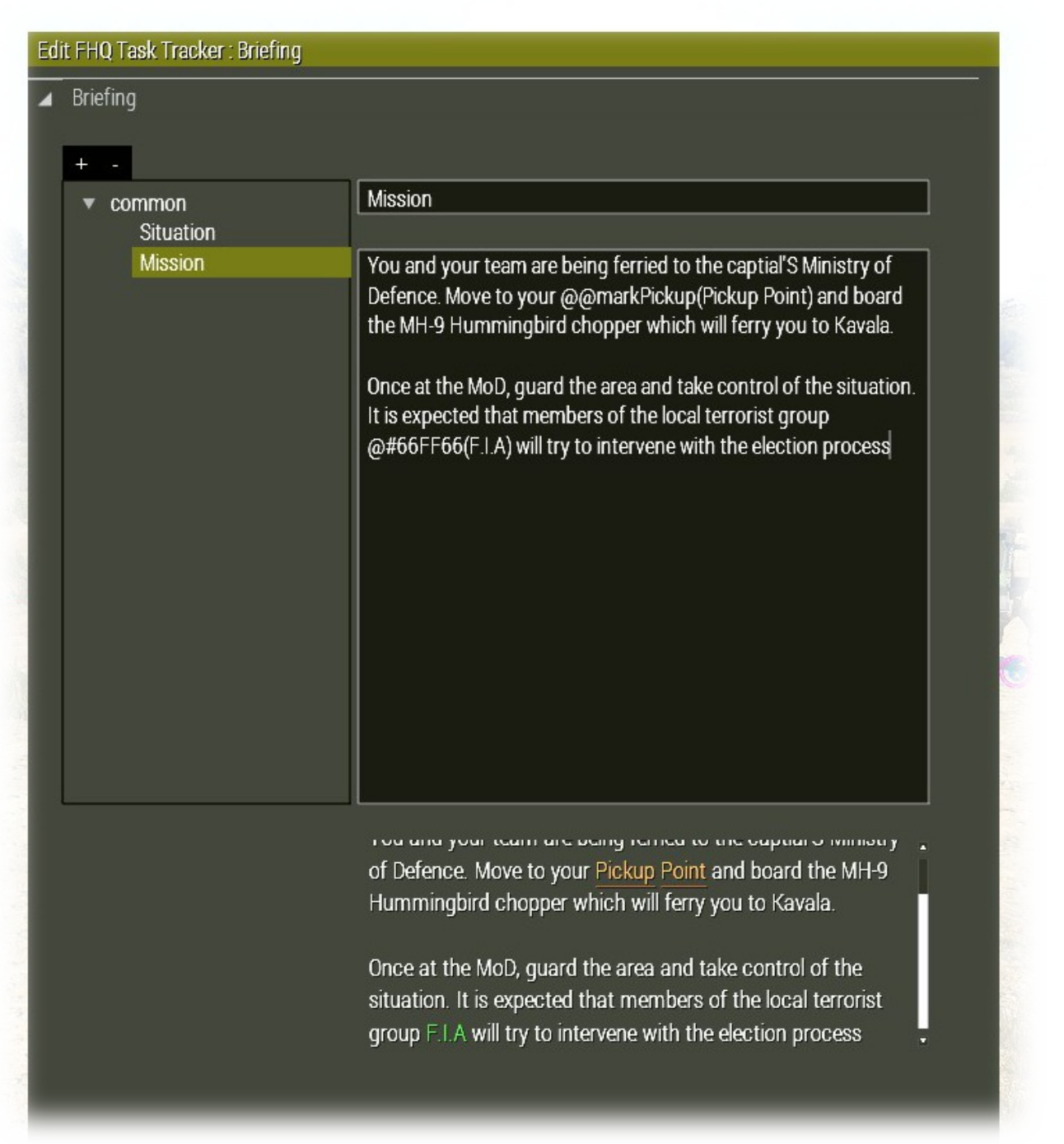
Once you select "Briefing..." the briefing editor will open. There are two buttons labeled "+" and "-" at the top, and a big button "Add New Briefing" in the center. The center will change according to what is selected in the tree view on the left side of the window (this is initially empty). Clicking the "Add New Briefing" button will add a new briefing with a default name, you can edit the text box to rename it to something meaningful. Also, a new button "Delete Briefing" will appear that does the obvious.



Once you have a briefing, the "+" and "-" buttons on top can be used to add briefing headers. Headers will appear under the currently highlighted briefing or in the same briefing as the currently highlighted header.

Adding a new briefing header will create one with a default name that should be changed in order to have some meaning. The headers are shown (along with the content) to the player that has received this briefing.

Note that once you have selected a header, the editor window will change to something that looks like this:



On the left is the tree view with the briefings and defined briefing headers. On the right, the top text box is the name of the header, while the bottom text box is the text of the briefing. At the very bottom, a preview is displayed how the briefing will approximately look in game.

Note that (as you might see in the picture) certain sequences can be used for extra functionality. First of all, any time you press CTRL-RETURN, a line feed is inserted (Arma 3 patch 1.58 and later only). This will be converted into the appropriate "
" sequence in game. A word of warning, though: Pressing only Return will close the dialog and save

the briefing. There is no way to override this behavior, it is defined by Arma itself.

Secondly, a couple of "commands" are available that start with an '@' sign. Currently, there are two such commands defined:

```
@#RRGGBB(Text to color)
```

And '@' sign followed by a '#' and six hexadecimal numbers defines a colored text. The bracketed text is displayed in-game with the specified color. Note that if the text should contain brackets itself, it must be set in quotes.

The color value can be obtained from multiple sources. Most paint programs like GIMP or Photoshop will display the six figure color code in their color mixer. Alternatively, there are web pages that contain predefined codes as well as live color mixers for the purpose.

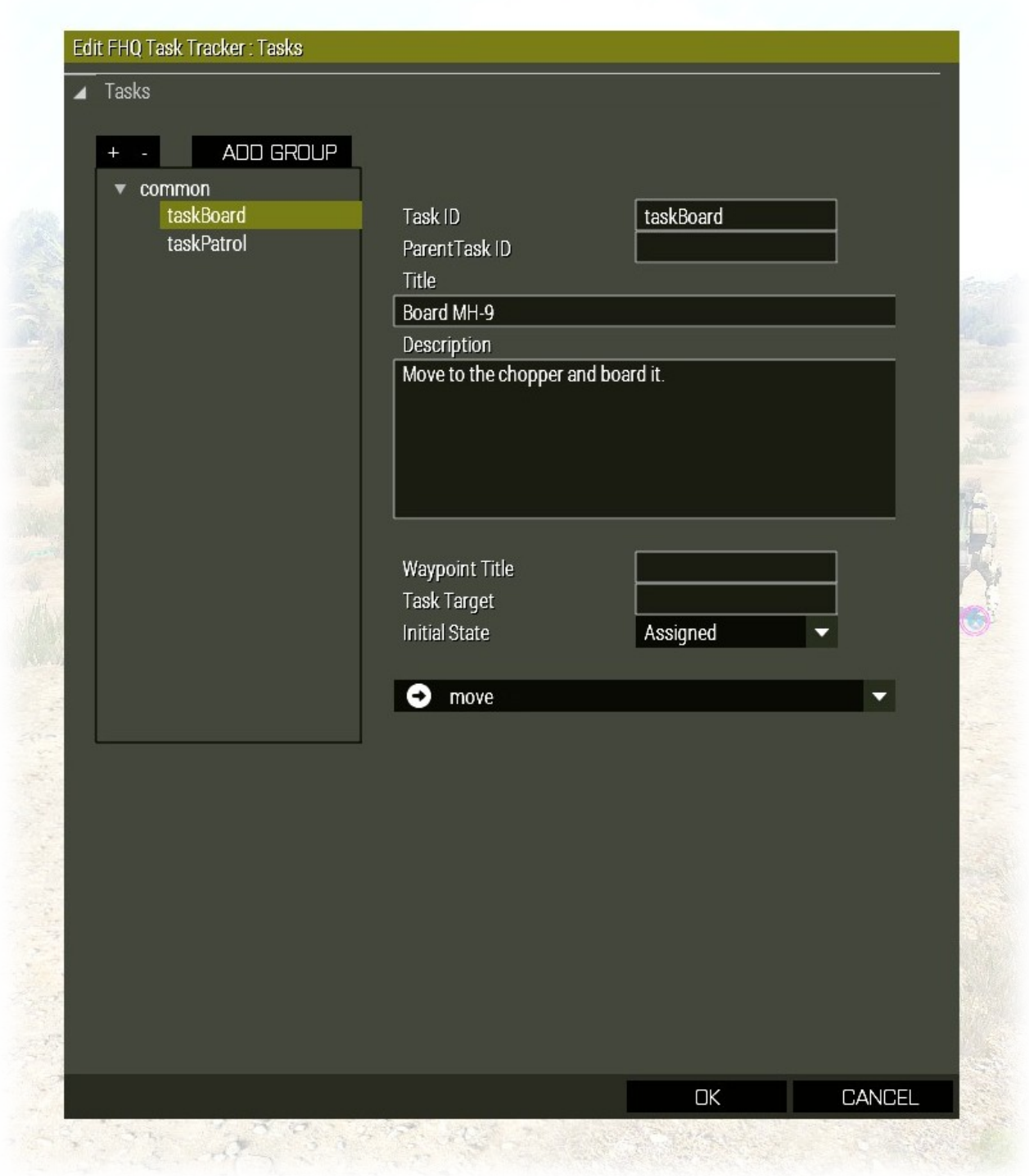
Note that there musn't be any space between the '@' sign and the opening bracket, but there might be spaces inside the brackets. For an example, refer to the picture above (the green FIA text).

```
@@marker(link text)
```

Two '@' signs followed by the name of a marker and bracketed text define a hyperlink in the briefing. The 'marker' must be the name of an existing marker on the map. The 'link text' will appear underlined in the briefing and will be clickable. The same rules apply as with the color command above. For an example, see the "Pickup Point" text in the image above.

7.3.2 Editing Tasks

Editing tasks works very similar to the briefings. Once the Task editor opens, the left side is empty and there are three buttons over the (initially empty) tree view. The '+' and '-' buttons add individual tasks to an existing task group. The "Add Group" button will add a task group.



New tasks are automatically named. These names do not appear to the user in game, so you can go with the defaults if you want, but it is much more sensible to name them in a meaningful way.

The right side of the window shows a couple of fields that need to be filled in. We'll go over all of them in sequence.

- Task ID: A short identifier for the task, as used in e.g. FHQ_fnc_ttSetTaskState and similar function. Must be unique.
- Parent Task ID: The ID of the parent task, if this is a complex task. If left empty, the task will be a normal task. Otherwise it will appear below the parent task as a subtask in game.
- Title: The name of the task that is displayed in the task menu in the Map screen. This should be a short, expressive name like "Assault Position" or "Defend Kavala".
- Description: A descriptive text telling the player what this task is about. Currently the briefing commands ('@@', '@#' etc) do not work here, you need to enter them manually. In the future, this will be added.
- Waypoint Title: A short string that appears next to the waypoint on the 3D view. As of 1.58, this is no longer used, but if you want your missions to be compatible with earlier versions of the game for Linux and Mac users, the waypoint tile could be set. Usually, it is short and written in all caps.
- Task Target: If empty, the task will not have a "target", that is, there will be no location on the map automatically associated with this task. There are, however, two shortcuts you can use to assign the task a target:
    - An '@' sign followed by a name indicates that the marker of that name should be used for the task target. In the example mentioned above, '@markPickup" would connect this task to the markPickup marker on the map.
    - A '#' sign followed by a variable name indicates the given object to be used as a task target. The object must have this variable name assigned in its Attributes dialog.
- Initial State: The initial state of the task. Tasks start out as "Created" usually, but changing one to "Assigned" will automatically assign this task to the players.
- Icon: Finally, starting with 1.58, the task can have its specific icon. The combo box will list all available system icons (support for mission-specific icons will come in a future version).

7.4 Assigning Tasks and Briefings

In order to assign the tasks and briefings that a player or players see, select the unit or units and choose "Attributes" from the popup menu. You will see a window that looks somewhat like this:

There is a Section "Object: FHQ Briefings and Tasks" at the bottom with two fields: Briefing, and Tasks Groups.

Currently you have to enter the names by hand (this will be fixed in a later update). The briefing is the briefing name that you used in the editor. In our prior example, this was "common". Here it is "west_briefing". You can have a number of briefings, for example for each side of the mission if it is adversarial, or for different player teams (just to give an example, "west_briefing_infantry" and "west_briefing_armour").

The lower field is the active tasks groups. You can enter more than one task group here, separated by comma. The unit will get all of the tasks in all of the groups that are mentioned here. This allows you to e.g. set some common goals ("Liberate Kavala") as well as specific goals ("Attack Stronghold on the eastern flank").

7.5 Closing words

We're going to continue updating the EdenTT plugin for easier use. We are also looking into ways to make scripting tasks unnecessary, i.e. provide the means to define rules under which certain tasks are considered done, and the mission won or lost.

8 Terms of Use

Distribution of the software is only allowed within a mission. In addition, no changes are allowed to the software without the permission of the authors.

Distribution as a standalone package is prohibited.

Attribution (getting a mention in the credits section of your mission) is appreciated, but not a requirement

You are allowed to distribute missions containing the software on the Steam Workshop.

9 Credits and Thanks

FHQ Task Tracker was written by Varanon, with additional help by Alwarren. I can be reached at thomas@friedenhq.org, or on the Bohemia Interactive Forum under my screen name Varanon.

The FHQ Task Tracker Eden Integration Mod was created by Alwarren. He can be reached at hans-joerg@friedenhq.org, or on the Bohemia Interactive Forum as Alwarren.

I want to thank all people that used FHQ Task Tracker in the past, especially those that gave me feedback (both good and bad) and suggested things that could be done better.

Thanks also go to Comrades in Arms (ciahome.net) for beta testing the Task Tracker, using it, for being the best online community I ever met, and for being friends.

Finally, thank you, Bohemia, for creating my hobby of 15 years. 2000 hours of Arma 3, even more in Arma 2, and no end in sight. Onwards and Upwards.