The FHQ TaskTracker

for Arma 2, Iron Front. Take On: Helicopters and Arma 3

Version 2.2

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 are not compatible with Arma 2, and have their limitations (like, it's not really possible to re-arrange entries, you have to add them in a certain order).
FHQ TaskTracker V1.x was a solution I came up with for this, but it had it's shortcomings when deployed in missions that allowed respawn and/or joining to disabled player slots (which is possible in GROUP, BASE and INSTANT respawn scenarios).
V 2.0 fixes this. It also has a cleaner internal representation, tracks briefing entries too (allowing dynamic updates to the briefing), and allows JIT on disabled clients and respawn.

History

- V 1.0 – Initial Release
- V 1.1 – Added Arma 3 notification for task hints using default task notification resources
- V 2.0 – Complete rewrite
- V 2.1 – Fixed a bug that would pop up the task name, not the short description on newly created tasks.
- V 2.2 – Fixed a bug that would pop up the task long description, not the short one, on newly created tasks (yeah, I really was that dense)

Usage

The FHQ Task Tracker is a single script that has to be executed on each client and the server that wants to share in tracking tasks. The easiest way is to execute it in init.sqf. This

is achieved by adding a single line to init.sqf like this:

```
call compile preprocessFileLineNumbers "fhqtt2.sqf";
```

This will set up the system and pre-initialize a number of variables that contain code that can be called from your own code.

All function variables have a prefix of `FHQ_TT_` to avoid clashes with other script packages.

Briefings are added by calling `FHQ_TT_addBriefing`. This code must be run on the server, and can be run on the clients, although it doesn't need to. The server will distribute the briefing to the clients that are currently connected, and any newly connected client-

Similarly, tasks are added by calling `FHQ_TT_addTasks` with the same constraints. Tasks and briefing entries can be added dynamically during the mission to adapt the mission to new circumstances.

Before going into the details of these functions, we need to discuss the concept of filters.

Concept: Filters

Filters are used to determine which unit will get which briefing entry respectively which task. By default, all playable units will receive entries if no filter is given.

If a filter is given in the input to `FHQ_TT_addBriefing` or `FHQ_TT_addTasks`, all subsequent entries are given only to units matching the filter, until a new filter is encountered.

There's a substantial difference between V1 and V2 of the task tracker: While in V1, a filter basically removed all filtered units from the pool, the V2 task tracker will always match all playable units against a new trigger. The V1 way of "hierarchical filtering" was removed because it proved to be not that useful. The example given below highlights a case that the old method could only implement by duplicating the whole array of entries.

A filter can be one of the following:

- Single object: A single unit that will receive the entries
- Group: All units in the group will receive the entries
- Side: All units of a side will receive the entries
- Faction (String): All units belonging to the faction will receive the entries
- Code: A piece of code that is evaluated for each unit with each entry. The unit is passed as `_this` parameter to the code piece, and the code should return either `true` or `false`, indicating whether the unit should receive the entry or not.

Example of all filter types:

- `westMan1_1`: The unit named `westMan1_1`. Note that you should NOT use `player` as filter, since this is only defined on a specific client, and not at all defined on a dedicated server.
- `group westMan1_1`: The group `westMan1_1` belongs to. All units of `westMan1_1`'s group will receive entries.

- `west`: All units belonging to side '`west`'.
- `"BLU_F"`: All units belonging to NATO (Arma 3)
- `{(side _this) != west}`: All units that are not on west.

Defining Briefings

A briefing entry always consists of an array of two or three strings. If two strings are given, the first string is the title to be put under the diary section (Mission in Arma 2, Briefing in Arma 3), and the second string is the text. With three strings, the first is the Subject, second is title, and third is the text.

The function to add a briefing is `FHQ_TT_addBriefing`. It's parameter is an array. Each entry of the array is either a filter, or a briefing entry. If it is a filter, all subsequent entries are applied to units matching the filter, until another filter is encountered.

Example:

```
 [
     west,
        ["Mission", "Get some!"],
        ["Enemy Forces", "There's lots of ruskies around"],
     east,
        ["Mission", "Get those imperialistic americans"],
        ["Current Supply of Vodka", "Low"],
     {true},
        ["Credits", "Mission by", "Some Dude"],
        ["Credits", "Uses scripts by", "Some other dude<br/>Yet
another dude"]
 ] call FHQ_TT_addBriefing;
```

The first two entries are added to the Briefing/Mission subject for all players on the `west` side. The second two are added to the same subject for all `east` players. The last two lines are added for each playable unit, under the subject "Credits" (not Briefing/Mission).

Note that while it is possible to do this, don't overdo it. Players will expect the briefing at the usual place, so keep the briefing there and don't clutter the main menu.

Defining Tasks

Task entries are slightly more complex. A task entry is an array that can have a different amount of entries, some of them are optional:

- **Task Name**. The first entry is always the task name. This is either a single string ("taskDestroy"), or an array of two strings (["taskDestroy1", "taskParent"]). Single string means a simple task, while the two string array defines a task and it's parent. Parented tasks are always displayed below the parent task, but apart from that, they behave like normal tasks.
- **Task Long Description**. This string is the task's long description. Basically, it's what explains the task in detail. This can be structured text.

- **Task Short Description**. This string is the headline for the task. It's what is displayed in tasks hints, and in the list of task (center column of the briefing/map window).
- **Task Waypoint Description**. This string is displayed on the task's waypoint when hovering over the waypoint on the map, and on the waypoint marker in the 3d view (if extended HUD info is enabled).
- **Target**. This is the target of the task. It can either be a position (three-part array, for example getMarkerPos "markername"), or an object. A position is fixed, while an object makes the waypoint move along with the actual object. This entry can also be omitted, resulting in a task with no location attached to it (for example, tasks like "Find the ammo cache").
- **Initial State**. This is a string indicating the initial state of the task. It can be omitted as well, which means the state will be "created". You can set it to any valid task state, but the most useful one is "assigned", which makes the task the current one on creation.

Tasks can be of one of five states: "created", "assigned", "succeeded", "failed", or "canceled".

Example:

```
[
    west,
       ["taskBoard1", "Board your chopper", "Board your chopper",
              "BOARD", westHelo1, "assigned"],
       ["taskCAS", "Fly around", "Fly around", "CAS"],
       ["taskRetreat1", "Return to LZ", "Return to LZ",
         "RETREAT", getMarkerPos  "markLZ"],
    "BLU_G_F",
       ["taskSecret", "Secret: Betray NATO for whatever reason",
              "Secret: Betray NATO", ""],
       [["taskSecret1", "taskSecret"], "Because they are bad",
         "Bad", ""],
       [["taskSecret2", "taskSecret"], "Because I am evil",
              "Evil", ""]
] call FHQ_TT_addTasks;
```

The first three entries are added to all west units. The first entry is bound to an object ("`westHelo1`") and is set to assigned. The second one has no target and no initial state, so it's state is "created". The third one has a location set, `getMarkerPos "markLZ"`, so a waypoint would be shown for the first and third task.

The second batch of tasks are only added to FIA units (Faction "BLU_G_F"). These entries are an example of parented tasks: The first one is the parent task (without tarkget), the second and third are tasks that will have the first as parent and are displayed below them.

Note: In V1, you needed to revert the order of parent and subtasks for Arma 2. This is no longer necessary in V2. All tasks will appear in the order they were given, i.e. `taskBoard1` in the above example will be the first (topmost) task.

Like the briefing, this code has to run on the server and can be run on the clients (where it doesn't have any effect). Note that setting tasks from a client is not supported. Also, similar to briefings, it's best to call this function in init.sqf or a script called from init.sqf.

Setting task states

As mentioned above, a task can be of five different states: "created", "assigned", "succeeded", "canceled", "failed". Transitioning from one state to the next is achieved with one of two functions:

`FHQ_TT_setTaskState` will set the state of a single task. It's input is a two parameter array, the first entry being a string defining the task name (as defined in `FHQ_TT_addTasks`), and the second one being a task state. If the state changed from it's previous state, the players that have this task will be notified.

Frequently, you will want to set a task to a terminal state (succeeded, canceled or failed) and select a follow-up task. For this purpose, you can use the function `FHQ_TT_setTaskStateAndNext`. This function receives a variable list of parameters. The first parameter is a task name, and the second one a task state. In that respect, the function is exactly like `FHQ_TT_setTaskState`. In addition, `FHQ_TT_setTaskStateAndNext` can accept an arbitrary number of additional task names. Each of the names in the list will be in turn checked, and if the state is not a terminal state (succeeded, canceled, or failed), the task will be set to assigned (and the player notified).

Example:

```
["taskBoard1", "succeeded"] call FHQ_TT_setTaskState;
```

```
["taskCAS", "succeeded", "taskRetreat1", "taskSecret1",
"taskSecret2] call FHQ_TT_setTaskStateAndNext;
```

The first example will simply set taskBoard1 to succeeded. The second example will do the same to taskCAS, but will then check taskRetreat1, taskSecret1, and taskSecret2, and will set the first one it finds to be in a non-terminal state to "assigned".

Note: These functions can only be called from the server. Calling them on a client will have no effect.

Note: The V1 function `FHQ_TT_markTaskAndNext` exists as an alias for `FHQ_TT_setTaskStateAndNext`, for backward compatibility.

Getting task states

There are several possibilities to query task state. The most basic is `FHQ_TT_getTaskState`. It's parameter is an array containing one string, the task name to be queried. It will return it's state, or an empty string if the task does not exist.

Example:

```
diag_log ["taskBoard1"] call FHQ_TT_getTAskState;
```

This will write "assigned" into the RPT file if called at the beginning of the mission.

Sometimes, it might be necessary to get all tasks with a certain state. This is achieved with the function `FHQ_TT_getAllTasksWithState`. It's input is an array containing one string, a task state.

Example:

```
diag_log str (["created"] call FHQ_TT_getAllTasksWithState);
```

This would output ["taskCAS", "taskRetreat1", "taskSecret", "taskSecret1", "taskSecret2"] to the RPT file if called at the beginning of the mission.

In addition, there are a number of functions that are specialized state queries. They are listed below:

- `FHQ_TT_isTaskCompleted`: Return true or false depending on whether the argument is in a terminal state or not.
  Example: `["taskCAS"] call FHQ_TT_isTaskCompleted;`

  This would return false at the beginning of the mission.
- `FHQ_TT_areTasksCompleted`: Like above, but a number of tasks is checked.
  Example: `["taskBoard1", "taskCAS"] call FHQ_TT_areTasksCompleted;`

  This would return true only if both taskBoard1 and taskCAS are in a terminal state.
- `FHQ_TT_isTaskSuccessful`: Like `FHQ_TT_isTaskCompleted`, but return true only if the task is successful.
- `FHQ_TT_areTasksSuccessful`: Like `FHQ_TT_areTasksCompleted`, but return true only if all tasks are successful.

Customization

There are several ways the task tracker can be customized.

- **Display of child tasks in Arma 2**: Currently, there is no way to discern child tasks in Arma 2. FHQ TaskTracker prefixes all child tasks with the contents of the variable `FHQ_TT_subtaskPrefix` *in Arma 2 only*. The default is " > " which basically indents the child tasks and places a ">" character in front.
- **Display of new briefing notifications in Arma 3**: New briefing entries are by default shown with the same notification as a newly assigned task. However, the code checks for a `CfgNotification` entry called NewBriefing and uses this instead of the TaskCreated notification. An example for this notification 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;
 };
};
```

This has to be added to your `description.ext` file.

- **Completely replacing the task hints function in any game**: FHQ TaskTracker calls the function `FHQ_TT_TaskHint` when it wants to notify the user about task and briefing updates. This function receives two arguments, the task title, and the task state. So, a primitive replacement would look like this:

```
FHQ_TT_TaskHint = {
  player sideChat format ["Task %1 changed to %2", _this
select 0, _this select 1];
};
```

Terms of Use

You are free to use this code in your own missions. I only ask that you do not modify the script without asking for permission first, and that you do not distribute modified versions of it. Getting a mention in the credits is appreciated, but not a requirement.

Credits and Thanks

FHQ TaskTracker was written by Varanon. I can be reached as Varanon on the BI forum, and via email at thomas@friedenhq.org.
I want to thank all people who used FHQ TaskTracker in the past and gave me valuable feedback on how to improve the script. Thanks also go out to Alwarren for the ArmaDev Eclipse plugin, which I don't want to miss anymore when developing missions for Arma. Thanks to Comrades in Arms for being the best multiplayer community I ever had the pleasure of playing with.
Finally, thanks to Bohemia Interactive for creating Arma. I hope we will still see many years of Arma development, Arma 3 and beyond. I know DayZ sold better, but I sincerely hope you won't let Arma die because of the success of that title. Thanks for countless hours of tactical fun in a game that has no equal.