

Combat Medical System – Third Party Support Documentation

The Third Party Support (**TPS**) module is developed by the CMS team to provide third party mod developers the ability to write their own modules and features for CMS. The goal is to provide an easy to use set of tools with the ability to hook into the core functionality without having to modify the existing structure.

This way you can create a new set of pbos for use with the original set released by the development team of CMS and end up with a completely new modification in total.

TPS provides the following features:

- Event handlers, triggered a single time for specific actions
- Hooks, triggered many times, mainly in loops
- UI EventHandlers, triggered when actions on the CMS UI are being performed, also allows for inserting new actions in specific categories.

WARNING: TPS IS FOR MOD DEVELOPERS WHO UNDERSTAND THE BASICS OF ARMA SCRIPTING. MAKING USE OF THIS FUNCTIONALITY CAN POTENTIALLY BREAK CMS.

PLEASE FOLLOW THE INSTRUCTIONS IN THIS DOCUMENT IF YOU WOULD LIKE TO MAKE USE OF TPS FUNCTIONALITLY.

*The CMS Team reserves the right to change any and all functionality in this module. The CMS team is not responsible for non-functional third party software as a result of changes made to the **TPS** module.*

Inhoud

Combat Medical System – Third Party Support Documentation	1
Event Handlers	3
Examples of Eventhandlers:	3
Available Handlers	4
handleDamage	4
Killed	4
Unconscious	4
wake up	4
AI Switch	4
Fully healed	4
cms_treatingPerson	5
Being treated	5
Dragging	5
Drop person	5
Used CMS Item	5
Loaded in vehicle	5
Init	5
Open UI	6
Start UP	6
Hooks	6
Example of hooks	7
Available Hooks	8
cms_sys_core	8
cms_sys_effects	8
cms_sys_blood	8
cms_sys_ui	8
cms_*Options	9
Example Config:	10

Event Handlers

```
class cms_eventHandlers {
    class cms_handleDamage{};
    class cms_killed{};
    class cms_unconscious{ };
    class cms_wakeup{};
    class cms_aiSwitch{};
    class cms_fullyHealed{};
    class cms_treatingPerson{};
    class cms_beingTreated{};
    class cms_dragging{};
    class cms_carry{};
    class cms_dropPerson{ };
    class cms_loadedInVehicle{};
    class cms_init{};
    class cms_openUI {};
    class cms_startUp{};
};
```

Examples of Eventhandlers:

In a config file:

```
class cms_thirdPartySupport {
    class cms_eventHandlers {
        class cms_handleDamage{
            class cms_sys_airway {
                init = "_this call (compile PreprocessFile
                    'cms_sys_airway\event_handleDamage.sqf');"
            };
        };
    };
};
```

This adds a new event handler to handleDamage, named test. Once the eventhandler handleDamage gets called, the function test will be called as well, executing the file located at "test\trial.sqf".

You can add unlimited amount of functions to a handler, for example:

```
class cms_thirdPartySupport {
    class cms_eventHandlers {
        class cms_handleDamage{
            class cms_sys_airway {
                init = "_this call (compile PreprocessFile
                    'cms_sys_airway\event_handleDamage.sqf');"
            };
            class cms_test {
                init = "hint format['params: %1',_this];";
            };
        };
    };
};
```

Available Handlers

handleDamage

Params: [_unit,_selectionName,_amountOfDamage,_sourceOfDamage,_typeOfProjectile]

_unit	=	OBJECT, unit that triggered the handler
_selectionName	=	STRING, part of the body that triggered the handler
_amountOfDamage	=	NUMBER, newly added damage
_sourceOfDamage	=	...
_typeOfProjectile	=	

Locality: unit

Killed

Params: [_unit,_newUnit]

_unit	=	OBJECT, unit that triggered the handler. Dead body
_newUnit	=	OBJECT, new unit.

Locality: unit

Unconscious

Params: [_unit,]

_unit	=	OBJECT, unit that triggered the handler.
-------	---	--

Locality: unit

wake up

Params: [_unit,]

_unit	=	OBJECT, unit that triggered the handler.
-------	---	--

Locality: unit

AI Switch

Params: [_unit,_oldBody]

_unit	=	OBJECT, unit that triggered the handler.
_oldBody	=	OBJECT, dead body

Locality: _oldBody

Fully healed

Params: [_unit,]

_unit	=	OBJECT, unit that triggered the handler.
-------	---	--

Locality: unit

cms_treatingPerson

Params: [_caller, _unit, _treatment, _bodyPart, _item]

_caller	=	OBJECT, unit that is performing the treatment
_unit	=	OBJECT, unit that is receiving the treatment
_treatment	=	CODE, code/function that is being called with the treatment
_bodyPart	=	STRING, bodypart selected
_item	=	STRING, classname of item being used

Locality: caller

Being treated

Params: [_caller, _unit, _treatment, bodyPartSelected, itemRemovedOnSuccess]

Locality: unit

Dragging

Params: [_unit, _caller]

_unit	=	OBJECT, unit that is being dragged.
_caller	=	OBJECT, unit that is performing the dragging

Locality: _caller

Drop person

Params: [_unit, _caller]

_unit	=	OBJECT, unit that is being dropped.
_caller	=	OBJECT, unit that is dropping the other

Locality: _caller

Used CMS Item

Params: [_unit, _item]

_unit	=	OBJECT, unit that uses the item.
_item	=	STRING, classname of item being used

Locality: _unit

Loaded in vehicle

Params: [_unit, _vehicle, _caller]

_unit	=	OBJECT, unit that is being loaded into a vehicle
_vehicle	=	OBJECT, vehicle that unit is being loaded in
_caller	=	OBJECT, person that is loading the unit into a vehicle

Locality: _unit

Init

Params: [_unit]

_unit	=	OBJECT, unit that starts up CMS. Ran on start-up and respawn
-------	---	--

Locality: _unit

Open UI

Params:[_caller]

 _caller = OBJECT, unit that opened the UI

Locality: _caller

Start UP

Params:[_unit]

 _unit = OBJECT, unit that is starting CMS

Locality: _unit

Ran on every unit at (re)spawn, that has CMS enabled.

Hooks

Hooks executed during loops:

- cms_sys_core
- cms_sys_effects
- cms_sys_blood
- cms_sys_ui

Hooks for inserting new options in the UI:

- cms_bandageOptions
- cms_medicationOptions
- cms_advancedOptions
- cms_examineOptions

Example of hooks

```
class cms_thirdPartySupport {
    class cms_hooks {
        class cms_sys_core{
            class cms_hookInSysCore{
                file = "file.sqf";
            };
        };
        class buttons {
            class cms_advancedOptions{
                class cms_airwayTreatment {
                    name = "Use NPA";
                    condition = "[player,'cms_nasopharyngeal_tube'] call cms_fnc_hasCMSItem";
                    file = "cms_sys_airway\button_airwayTreatment.sqf";
                    priority = 1;
                };
            };
            class cms_examineOptions{
                class cms_airwayExamine {
                    name = "Airway";
                    condition = "true";
                    file = "cms_sys_airway\button_airwayExamine.sqf";
                    priority = 1;
                };
            };
        };
    };
};
```

Available Hooks

cms_sys_core

Used to tap into the core system loop. This is where everything gets calculated. It runs on every unit that is using CMS. Minimum time between laps: 1 second

Params: [_unit]

 _unit = OBJECT, unit on which the loop is running

Locality: _unit

cms_sys_effects

Used to tap into the effects system loop. This is where all the on screen effects are running from. It runs on every unit that is using CMS. Minimum time between laps: 1 second

Params: [_unit]

 _unit = OBJECT, unit on which the loop is running

Locality: _unit

cms_sys_blood

Used to tap into the blood system calculation. This is running from the core loop. It runs on every unit that is using CMS.

Params: [_unit]

 _unit = OBJECT, unit on which the loop is running

Locality: _unit

cms_sys_ui

Used to tap into the ui loop. This will only run on a player client. Executed while a player is looking in any CMS UI.

Params: [_unit]

 _unit = OBJECT, unit on which the loop is running

Locality: _unit

cms_*Options

- cms_bandageOptions
- cms_medicationOptions
- cms_advancedOptions
- cms_examineOptions

All three of these work the same. Are executed the moment a player clicks on any of the four (4) main buttons in the UI, bandage, medication, advanced, examine.

These hooks are meant to add extra options to the UI. The file that is listed here is what is being called upon button pressed. This file can contain the functionality of your button's action, or (CMS standard) in case of treatment switch the locality.

Example:

```
[_this select 0, _this select 1, cms_fnc_AirwayTreatment, "cms_nasopharyngeal_tube"] call cms_fnc_performTreatment;
```

Params: [_caller, _target]

 _caller = player object
 _target = OBJECT, unit that the action is performed with. Can be any other unit or player object.

An example of how to make use of this can be found at GlowbalProductions.com

http://glowbalproductions.com/files/cms/cms_sys_airway.zip

Example Config:

```
class CfgPatches
{
    class cms_sys_airway
    {
        units[] = {};
        weapons[] = {};
        requiredVersion = 0.1;
        requiredAddons[] = {"cms_main", "cms_ui"};
        version = "0.5";
        author[] = {"Glowbal"};
    };
};

class CfgAddons {
    class PreloadAddons {
        class cms_sys_airway {
            list[] = {"cms_sys_airway"};
        };
    };
};

class cms_thirdPartySupport {
    class cms_eventHandlers {
        class cms_handleDamage{
            class cms_sys_airway {
                init = "_this call (compile PreprocessFile 'cms_sys_airway\event_handleDamage.sqf');"
            };
        };
        class cms_init{
            class cms_sys_airway {
                init = "_this call (compile PreprocessFile 'cms_sys_airway\init.sqf');"
            };
        };
        class cms_startUp{
            class cms_sys_airway {
                init = "_this call (compile PreprocessFile 'cms_sys_airway\event_startUp.sqf');"
            };
        };
    };

    class cms_hooks {
        class cms_sys_core{
            class cms_sys_airwayExample {
                file = "cms_sys_airway\sys_airway.sqf";
            };
        };

        class buttons {
            class cms_advancedOptions{
                class cms_airwayTreatment {
                    name = "Use NPA";
                    condition = "([player,'cms_nasopharyngeal_tube'] call cms_fnc_hasCMSItem)";
                    file = "cms_sys_airway\button_airwayTreatment.sqf";
                    priority = 1;
                };
            };
        };
    };
};
```