# **Hunter’z Persistency Module**

# **WARNING** This only serves as preliminary documentation for the module. You will only find API functions described in here. You will still need to be able to configure the module correctly from the editor and manually manage save files (copy-paste-rename) for them to be loaded correctly at server restart. This will be described in later versions of the manual when testing of the module is complete. For now, you’re on your own.

# **Introduction**

This module doesn’t work by just saving every vehicle/soldier on the battlefield. In most cases, that’s not what mission makers are after, and it’s highly inefficient. Instead, it saves whatever you want it to save. This is best described as the vehicles, ammo crates and objects (ranging from fortifications, to lamps and campfires – basically could be anything you want) that you “own”, or otherwise want to save.

In addition, you are able to configure variables as persistent. Variables can be in global or object namespace, and each will have their own API function to easily configure them as persistent. Persistent global variables can be used to store gameplay/campaign information, such as how many civilians have been killed in total, or how much money your side has, while object namespace variables can be made persistent to store “getvariable/setvariable” values together with the persistent vehicles/objects themselves.

Player persistency (like in DayZ) works automatically, unless you choose the scripted wait option in the module configuration in the editor.

As this is a persistency framework, it is designed for complex dynamic missions in mind. Anything you want to save, is expected to be dynamically spawned during mission execution, and not placed into the mission using the editor. For example, it’s sensible to save vehicles that you spawn or “buy” using a vehicle spawner/store, but not a vehicle that you place in the editor that always re-appears when you restart the server anyway.

This module assumes you have scripting and server configuration knowledge. It’s not intended to be a strictly editor-only mission making mod.

# **API Functions**

With regards to what is written above on how the module functions, you’ll have to use API-functions provided by the module to add whatever vehicle, ammo crate or object you want to set as “persistent”. The following are the list of API functions available:

## Hz\_pers\_API\_addCrate

**Use:** Turns an existing ammo crate persistent.

**Argument(s):** \_this: reference to object type -> object

**Example:** \_crate **call** Hz\_pers\_API\_addCrate;

## Hz\_pers\_API\_addCrateVariable

**Use:** Makes an object namespace variable persistent for all of your persistent ammo crates. Can only be called by the server. The variable does not have to be defined for all crates.

**Argument(s):** \_this select 0: name of variable type -> string

\_this select 1: variable is publicvariabled type-> bool

**Example:** [“canBeMoved”, true] **call** Hz\_pers\_API\_ addCrateVariable;

Hz\_pers\_API\_addMissionVariable

**Use:** Makes a global variable persistent. Can only be called by the server.

**Argument(s):** \_this select 0: name of variable type -> string

\_this select 1: parsing descriptor type -> number

\_this select 2: variable is publicvariabled type -> bool

The **parsing descriptor** is a number between **0** to **3** that is needed for the module to understand what kind of variable this is.

Use **0** for anything that is not an array.

If it’s an array, depending on its dimension you need to use either:

**1** if it’s a one-dimensional array

**2** if it’s a multidimensional array in the form **[ [ ],[ ],[ ],… ]**

Or **3** if it’s an array of multidimensional arrays, as in **[** **[ [ ],[ ],[ ],… ] , [ [ ],[ ],[ ],… ], … ]**

It’s unlikely that you’ll have to store any data form this complex, so you’ll most likely be using either 0 or 1. However, if you don’t use the correct number, you can corrupt your save file! For example, if you put “0” there and you’ve actually got an array, it might work, but if that array grows to become very large, you’ll eventually corrupt your save file.

**Example:** [“numberOfTasksCompleted”, 0, false] **call** Hz\_pers\_API\_ addMissionVariable;

[“adminNames”, 1, true] **call** Hz\_pers\_API\_ addMissionVariable;

## Hz\_pers\_API\_addObject

**Use:** Turns an existing object persistent.

**Argument(s):** \_this: reference to object type -> object

**Example:** \_obj **call** Hz\_pers\_API\_addCrate;

## Hz\_pers\_API\_addObjectVariable

**Use:** Makes an object namespace variable persistent for persistent objects added using the Hz\_pers\_API\_addObject function. Can only be called by the server. The variable does not have to be defined for all objects.

**Argument(s):** \_this select 0: name of variable type -> string

\_this select 1: variable is publicvariabled type-> bool

**Example:** [“canBeCarried”, true] **call** Hz\_pers\_API\_ addCrateVariable;

## Hz\_pers\_API\_addPlayerVariable

**Use:** Makes an object namespace variable persistent for all player units. Can only be called by the server. The variable does not have to be defined for all units.

**Argument(s):** \_this select 0: name of variable type -> string

\_this select 1: variable is publicvariabled type-> bool

**Example:** [“isMedic”, false] **call** Hz\_pers\_API\_ addPlayerVariable;

## Hz\_pers\_API\_addVehicle

**Use:** Turns an existing vehicle persistent.

**Argument(s):** \_this: reference to object type -> object

**Example:** \_car **call** Hz\_pers\_API\_ addVehicle;

Hz\_pers\_API\_addVehicleVariable

**Use:** Makes an object namespace variable persistent for all of your persistent vehicles. Can only be called by the server. The variable does not have to be defined for all vehicles.

**Argument(s):** \_this select 0: name of variable type -> string

\_this select 1: variable is publicvariabled type-> bool

**Example:** [“isLocked”, false] **call** Hz\_pers\_API\_ addVehicleVariable;

## Hz\_pers\_API\_disablePlayerSaveStateOnDisconnect

Not documented yet.

## Hz\_pers\_API\_enablePlayerSaveStateOnDisconnect

Not documented yet.