

# Scripting the Nimitz

The USS Nimitz in Arma 3 is a heavily scripted vehicle. Many of the capabilities of the carrier are available as functions and can be used in missions. However, documentation on these functions is particularly scarce and the existing modules documentation is slightly outdated, though probably still useful. This document should help to bridge the gap and is of interest to anyone who wants to automate Nimitz capabilities via script.

## Parts and Pieces

Unlike most vehicles, the Nimitz is not a single object, but a few dozen objects combined to present the carrier. This is possible in Arma 3 as an object can define a init function that run as soon as the object is placed in Eden or spawned in a mission.

The Nimitz central object is called **JDG\_carrier\_spawner** and is itself an invisible object with a few properties defined. Its init function is really where stuff happens: all the individual objects that make up the Nimitz are spawned from the init function and hence the carrier is completely assembled in game.

The list of objects that are assembled is:

```
_parts =  
[  
    "JDG_carrier_nimspots",  
    "JDG_carrier_nimlights",  
    "JDG_carrier_nimlightsInternal",  
    "JDG_carrier_island",  
    "JDG_carrier_deck_0",  
    "JDG_carrier_deck_1",  
    "JDG_carrier_deck_2",  
    "JDG_carrier_deck_3",  
    "JDG_carrier_deck_4",  
    "JDG_carrier_deck_5",  
    "JDG_carrier_deck_6",  
    "JDG_carrier_deck_7",  
    "JDG_carrier_deck_8",  
    "JDG_carrier_deck_9",  
    "JDG_carrier_deck_10",  
    "JDG_carrier_deck_11",  
    "JDG_carrier_deck_12",  
    "JDG_carrier_deck_13",  
    "JDG_carrier_deck_14",  
    "JDG_carrier_ele_1",  
    "JDG_carrier_ele_2",  
    "JDG_carrier_ele_3",  
]
```

```
"JDG_carrier_ele_4",  
"JDG_carrier_hangar_0",  
"JDG_carrier_hangar_1",  
"JDG_carrier_hangar_2",  
"JDG_carrier_hangar_3",  
"JDG_carrier_hangar_4",  
"JDG_carrier_second_deck_0",  
"JDG_carrier_second_deck_1",  
"JDG_carrier_second_deck_2",  
"JDG_carrier_second_deck_3",  
"JDG_carrier_second_deck_4",  
"JDG_carrier_hangar_12r",  
"JDG_carrier_hangar_34r",  
"joe_nauticalbridge",  
"JDG_dynamicAirportNimitz"  
];
```

The current list of objects is best retrieved from inspecting JDG\_carrier\functions\fn\_init.sqf, as the list will probably change over time.

Of all these objects one is of particular interest to any scrippter: JDG\_carrier\_nimspots. I placed it unbinarized on <http://tetet.de/arma/arma3/nimitz/nimspots.p3d> for easier use. The nimspots object contains dozens of so called mem points that exactly pinpoint specific locations on the carrier. Such locations as the catapults, the catapults terminating locations and so forth.

## Accessing the pieces

If you want to manipulate any of the objects you can retrieve a handle to it via the JDG\_carrier\_spawner object. All pieces are set as variable in the spawner object.

```
// tie the part to the carrier spawner  
_nimitz setVariable [_x, _part, true];  
// same in reverse  
_part setVariable ["TTT_NimitzParent", _nimitz, true];
```

This means that if you name your carrier for example Nimitz, you can retrieve the nimspots object via:

```
private _nimspots = Nimitz getVariable "JDG_carrier_nimspots";
```

If for whatever reason you got the handle to an object of the Nimitz, you can get the JDG\_carrier\_spawner object via the *TTT\_NimitzParent* variable:

```
private _carrier = _someObjectOfTheNimitz getVariable "TTT_NimitzParent";
```

Popular objects you may need are the above mentioned nimsots, the JDG\_carrier\_hangar\_4 for setting the briefing room textures, and joe\_nauticalbridge for setting the monitor textures on the bridge.

## Using a mempoint

As mentioned above, the nimsots object contains a number of mempoints that can be used in scripts. For example, for spawning a F/A-18E on catapult 1:

```
private _spawnPos = _nimsots modelToWorld (_nimsots selectionPosition "C1");  
private _spawnDir = getDir Nimitz;  
private _plane = "JS_JC_FA18E" createVehicle _spawnPos;  
_plane setPosASL [getPos _plane select 0, getPos _plane select 1, 17.5];  
_plane setDir _spawnDir;
```

Note that the code deliberately places the plane on the deck (approx 17.5 meters above sea level). The spawnDir is only approximate either, as it should actually be towards the end of catapult 1 ("C1T" mempoint). This is left to the reader's exercise.

## Setting a texture

In Arma 3 objects can define so called hiddenSelections and on those one can apply textures with setObjectTextureGlobal. For example, most of the monitors on the bridge can hold a custom texture now. A sample code for this is:

```
_bridge setObjectTextureGlobal [0, "\path\to\texture\in\mission.paa"];
```

The 0 number is a reference to the first defined monitor, 1 to 14 are other possibilities.

## Operating an elevator by script

One of the most prominent features of the Nimitz are the moving elevators. The code for operating them is in a separate pbo from the main carrier: **tnt\_nimitzfunctions.pbo**. This pbo contains some graphics and a lot of code. The code is arranged in functions starting with tnt\_ and can be viewed from the in game functions viewer.

The relevant function for operating an elevator is **fn\_elevator\fn\_elevator.sqf**. The function takes two arguments:

1. The elevator object
2. One Number from 0,1,2,3

The numbers have special meanings:

- 0 - send elevator up for elevators 1 - 4
- 1 - send elevator down for elevators 1 - 4
- 2 - send weapons elevator up
- 3 - send weapons elevator down

Note that this is not a particular nice piece of code, but it was kept like this for backward compatibility.

To complete our tour on operating an elevator, here is some sample code to send elevator 4 to the hangar level:

```
private _elevator4 = Nimitz getVariable "JDG_carrier_ele_4";
[_elevator4, 1] spawn ttt_fnc_elevator;
```

## Launching planes

Starting planes via the catapults is a nice feature to add some ambiance to the carrier operations. In `ttt_nimitzfunctions` there is the function **fn\_ambiance\fn\_planeStart.sqf** that can be used. It is pretty much hardcoded what happens there, so if you want to modify the start sequence, you best copy this function and adjust it to your needs.

With the released function you can start for example a plane from catapult 2 via:

```
[Nimitz, _plane, 2] spawn ttt_fnc_planeStart;
```

You can then add additional waypoints to `_plane` and give it a tasking on your mission.

## Landing planes

With help of the dynamic airport object in the Nimitz AI can now land on the carrier. Unfortunately I haven't figured out to use the arresting hook script automatically, so you need to specify this in the script:

```
[_plane] spawn TTT_fnc_arrest;
_plane landAt (Nimitz getVariable "JDG_dynamicAirportNimitz");
```

## Placing parked planes

The different ambiance modules of the Nimitz take care of placing aircraft and associated vehicles on the carrier. However, at times the need for placement of only a few vehicles might arise. For this purpose a set of functions in **ttt\_nimitzfunctions\fn\_parking** exist. The different functions place aircraft and other vehicles on the carrier at the specified positions. For that they make use of the **fn\_parkPlane.sqf** and **fn\_parkVehicle.sqf** functions. An example from the ambiance module to park planes on the street:

```
[Nimitz, "JS_JC_FA18E", "cas", true] spawn TTT_fnc_parkStreet;
```

An overview of the different area and their names can be found at <http://www.combat.ws/S4/SAILOR/APNDX6.HTM>

## Resetting catapult and arresting wires

Unfortunately not all operations are smooth and sometimes it is required to do a hard reset on specific subsystems of the Nimitz. This can either be caused by script errors or by human error. Two functions exist for this purpose: **fn\_resetCat.sqf** and **fn\_resetWires.sqf**, both found in **ttt\_nimitzfunctions\fn\_reset**. To reset both systems issue:

```
[Nimitz] call ttt_fnc_resetWires;  
[Nimitz] call ttt_fnc_resetCat;
```