**SERVER SIDE INSTALL**

**Step 1-- BattlEye Filters**

**publicvariableval.txt**

Near the top of the last page full of keywords you will find these three lines:

5 "createdialog"

5 "\(" !"TELEPORT REVERT"

5 "'"

HDAS causes problems with the middle of those three, because of the new public variable handler installed server-side and the initialization of the PV values. There’s almost certainly a more elegant and more functional way of fixing it, but I did it simply by changing this sequence to:

5 "createdialog"

1 "\(" !"TELEPORT REVERT"

5 "'"

=============ABOVE ONLY REQUIRED FOR 1.0.6.1 =========================

**setpos.txt**

There is only one kick level 5 line – the first one. Add to the end of this line the exception:

!"MetalFloor\_DZ"

**Step 2 -- Server Code**

**Step 2.1 -- \dayz\_server\compile\server\_updateObject.sqf**

Step 2.1.1 -- Near the top of the file, on the first page generally displayed, there is some code into which the “hangardoor” flag needs to be integrated in three places:

\_objectID = \_object getVariable ["ObjectID","0"];

\_objectUID = \_object getVariable ["ObjectUID","0"];

if ( ( ( typeName \_objectID == "SCALAR" ) || ( typeName \_objectUID == "SCALAR" ) ) && ( \_type != "hangardoor" ) ) then {

[ nil , nil , rTITLETEXT , "SCALAR TYPE FORCE FAIL" , "PLAIN DOWN" , 5 ] call RE;

#ifdef OBJECT\_DEBUG

diag\_log ( format[ "Non-string Object: ID %1 UID %2" , \_objectID , \_objectUID ] );

#endif

//force fail

\_objectID = nil;

\_objectUID = nil;

};

if ( !( ( typeOf \_object ) in DZE\_safeVehicle ) && !locked \_object && ( \_type != "hangardoor" ) ) then {

//diag\_log format["Object: %1, ObjectID: %2, ObjectUID: %3",\_object,\_objectID,\_objectUID];

if ( !( \_objectID in dayz\_serverIDMonitor ) && isNil { \_objectUID } && ( \_type != "hangardoor" ) ) then {

//force fail

[ nil , nil , rTITLETEXT , "SCALAR TYPE FORCE FAIL" , "PLAIN DOWN" , 5 ] call RE;

\_objectID = nil;

\_objectUID = nil;

};

======================= above is for 1.0.6.1 ===================

In about the middle of the first page, change code to include “hangardoor” in three locations;

// [\_object,\_type] spawn server\_updateObject;

#include "\z\addons\dayz\_server\compile\server\_toggle\_debug.hpp"

if (isNil "sm\_done") exitWith {};

private ["\_class","\_objectID","\_objectUID","\_object\_position","\_isNotOk","\_object","\_type","\_recorddmg","\_forced","\_lastUpdate","\_needUpdate","\_object\_inventory","\_object\_damage","\_objWallDamage","\_object\_killed","\_object\_maintenance","\_object\_variables","\_totalDmg"];

\_object = \_this select 0;

\_type = \_this select 1;

\_recorddmg = false;

\_isNotOk = false;

\_forced = if (count \_this > 2) then {\_this select 2} else {false};

\_totalDmg = if (count \_this > 3) then {\_this select 3} else {false};

\_objectID = "0";

\_objectUID = "0";

if ((isNil "\_object") || {isNull \_object}) exitWith {

diag\_log "server\_updateObject.sqf \_object null or nil, could not update object"

};

\_objectID = \_object getVariable ["ObjectID","0"];

\_objectUID = \_object getVariable ["ObjectUID","0"];

\_class = typeOf \_object;

if ( ( ( typeName \_objectID != "STRING" ) || ( typeName \_objectUID != "STRING" ) ) && ( \_type != "hangardoor" ) ) then {

#ifdef OBJECT\_DEBUG

diag\_log (format["Non-string Object: ID %1 UID %2", \_objectID, \_objectUID]);

#endif

//force fail

\_objectID = nil;

\_objectUID = nil;

};

if ( !( \_class in DZE\_safeVehicle) && !locked \_object && ( \_type != "hangardoor" ) ) then {

//diag\_log format["Object: %1, ObjectID: %2, ObjectUID: %3",\_object,\_objectID,\_objectUID];

if ( !( \_objectID in dayz\_serverIDMonitor ) && isNil { \_objectUID } && ( \_type != "hangardoor" ) ) then {

//force fail

\_objectID = nil;

\_objectUID = nil;

};

….

======================= above for 1.0.6.2 ==========================

Step 2.1.2 -- Then, near the end of the file, between the \_object\_variables and \_object\_setVariables definitions, add the following two blocks of code:

\_object\_HangarDoor = {

private [ "\_inventory" , "\_key" , "\_coins" , "\_lstDummies" , "\_lstVectorAndID" , "\_lstEmpty" ];

\_lstDummies = \_object getVariable [ "DummyNameList" , [] ];

\_lstVectorAndID = \_object getVariable [ "VectorAndID" , [] ];

\_lstEmpty = [];

\_inventory = [ [ \_lstEmpty , \_lstEmpty ] , [ \_lstDummies , \_lstVectorAndID ] , [ \_lstEmpty , \_lstEmpty ] ];

if ( \_objectID == "0" ) then {

\_key = format[ "CHILD:309:%1:" , \_objectUID ] + str \_inventory + ":";

} else {

\_key = format[ "CHILD:303:%1:" , \_objectID ] + str \_inventory + ":";

};

// required if the server is using coins

if ( Z\_SingleCurrency ) then {

\_coins = \_object getVariable [Z\_MoneyVariable, -1];

\_key = \_key + str \_coins + ":";

};

\_key call server\_hiveWrite;

};

\_object\_HangarDoorClear = {

private [ "\_key" , "\_coins" , "\_lstEmpty" ];

// an empty list

\_lstEmpty = [];

if ( \_objectID == "0" ) then {

\_key = format[ "CHILD:309:%1:" , \_objectUID ] + str \_lstEmpty + ":";

} else {

\_key = format[ "CHILD:303:%1:" , \_objectID ] + str \_lstEmpty + ":";

};

// required if the server is using coins

if ( Z\_SingleCurrency ) then {

\_coins = \_object getVariable [Z\_MoneyVariable, -1];

\_key = \_key + str \_coins + ":";

};

\_key call server\_hiveWrite;

};

Step 2.1.3 -- Finally, at the very end, add the last two new cases to the switch statement:

switch ( \_type ) do {

case "all": {

call \_object\_position;

call \_object\_inventory;

call \_object\_damage;

};

case "position": {

call \_object\_position;

};

case "gear": {

call \_object\_inventory;

};

case "maintenance": {

call \_object\_maintenance;

};

case "damage"; case "repair" : {

call \_object\_damage;

};

case "killed": {

call \_object\_killed;

};

case "accessCode"; case "buildLock" : {

call \_object\_variables;

};

case "objWallDamage": {

call \_objWallDamage;

};

case "hangardoor": {

call \_object\_HangarDoor;

};

case "hangardoorclear": {

call \_object\_HangarDoorClear;

};

};

For 1.0.6.2, add the two new blocks before the case ‘coins” block.

**Step 3 -- \dayz\_server\init\server\_functions.sqf**

3.1 Add one line of code near the top between the waitUntil and the BIS\_MPF… statements:

waitUntil {!isNil "bis\_fnc\_init"};

// load up the server-side HDAS event handlers

execVM "\z\addons\dayz\_server\init\server\_HangarDoorHandlers.sqf";

BIS\_MPF\_remoteExecutionServer = {…

**Step 4 -- \dayz\_server\init\server\_HangarDoorHandlers.sqf**

This is a new file.

**Step 5 -- \dayz\_server\system\server\_monitor.sqf**

About 40% of the way through the file, add this IF block statement between the assignment of a value to \_isSafeObject and the comment that says “Don’t add inventory for traps”:

\_isSafeObject = \_type in DayZ\_SafeObjects;

// handle incoming previously set up hangardoor configuration data , which will always load in the CLOSED position

if ( ( count \_inventory > 0 ) && ( \_type == "MetalFloor\_DZ" ) ) then {

\_object setVariable [ "DummyNameList" , ( \_inventory select 1 select 0 ) , true ];

\_object setVariable [ "VectorAndID" , ( \_inventory select 1 select 1 ) , true ];

\_object setVariable [ "OpenOrClosed" , "closed" , true ];

};

//Dont add inventory for traps.

if ( !\_isDZ\_Buildable && !\_isTrapItem && ( \_type != "MetalFloor\_DZ" ) ) then {

**Step 6 -- CLIENT SIDE INSTALL**

You’ll need someplace to store the new sounds. I use a \sounds folder located at mission root. You’ll need to load up these two files:

\sounds\DoorsClosing.ogg

\sounds\MachineNoise.ogg

**Step 6.1 -- Description.ext**

After a line of code that says “diagHit = 1;” and before an include statement “#include "\z\addons\dayz\_code\gui\description.hpp"” you’ll need to add, if you don’t already have one, a configuration class block for the new sounds:

class CfgSounds

{

sounds[] = {

oggVMCEngine,

oggDoorsClosing

};

class oggVMCEngine

{

name = "oggVMCEngine";

sound[] = {"sounds\MachineNoise.ogg", 1.0, 1.0};

titles[] = {};

};

class oggDoorsClosing

{

name = "oggDoorsClosing";

sound[] = {"sounds\DoorsClosing.ogg", 1.0, 1.0};

titles[] = {};

};

};

If you already have a config class block for other sounds, simply add in the class definitions for the two new ogg files.

**Step 6.2 -- \newdata\HDAS\_newdata.sqf**

I have all the new variables for all my add-ons in one location, a \newdata folder. They could just as easily be integrated into any other startup variables script or other convenient location. If you use the file provided, the call to it from init.sqf is provided below.

**Step 6.3 -- Init.sqf**

Immediately after this code:

call compile preprocessFileLineNumbers "Mods\init\variables.sqf";

progressLoadingScreen 0.05;

…place this code in init.sqf:

// set up new variables

call compile preprocessFileLineNumbers "newdata\HDAS\_newdata.sqf";

And at the very end of init.sqf – subject to the requirements of any other add-ons you have -- place this code:

// call hangar door HDAS client-side setup

[] execVM "HDAS\_Client\HDAS\_compiles.sqf";

**Step 6.4 -- fn\_selfActions.sqf**

You will need to modify fn\_selfActions.sqf. The typical A2 DayZ Epoch mod install places a copy of this file under mods\compiles for 1.0.6.1, in which case all you need to do is edit it in order that the changes become a part of your mission download. I have a copy in an overrides folder because my changes to this file are extensive. In any case, look in your version for a block of code blocked by: if (\_isPZombie) then { …. }; Insert the following code immediately after that block.

For 1.0.6.2, you can find fn\_selfActions.sqf in @Dayz\_Epoch\dayz\_code.pbo

// begin large metal floor / hangar door options

//-------------------------------------------------------------------------------------------------------------------------------------------------------

// a defining scope

if ( true ) then {

// copy cursor target to this scope's var

\_objHangarDoor = cursorTarget;

// can't be IN a vehicle....

if ( !\_inVehicle && ( ( typeOf \_objHangarDoor ) == "MetalFloor\_DZ" ) && ( gblObjLastHangarDoor != \_objHangarDoor ) ) then {

// check the option menu control variable

if ( gblbHangarDoorOptControl < 1 ) then {

// ok, we've got one and the options are NOT up, or the options belonged to the last floor segment -- stash the current pointed-to vehicle as the LAST one, for the next time around

gblObjLastHangarDoor = \_objHangarDoor;

// do the menu add action for the add options only and stash the menu item index in a temporary variable

\_handle = gblObjLastHangarDoor addAction ["Hangar Door Options", "HDAS\_Client\HangarDoors\_AddOptions.sqf", gblObjLastHangarDoor , 0 , false , false , "" , "" ];

// add the "Hangar Door Options" option to the remove option list

gblListHangarDoorOptions set [count gblListHangarDoorOptions , \_handle];

// set the admin vehicle option control flag

gblbHangarDoorOptControl = 1;

} else {

// take the options down

{ gblObjLastHangarDoor removeAction \_x } forEach gblListHangarDoorOptions;

// null out the options

gblListHangarDoorOptions = [];

// the flag WAS true, so set it false now

gblbHangarDoorOptControl = 0;

// force the selfactions menu to re-eval

gblObjLastHangarDoor = objNull;

};

} else {

// look for the converse of the conditions above

if ( \_inVehicle || ( ( typeOf \_objHangarDoor ) != "MetalFloor\_DZ" ) ) then {

// take the options down

{ gblObjLastHangarDoor removeAction \_x } forEach gblListHangarDoorOptions;

// null out the options

gblListHangarDoorOptions = [];

// the flag WAS true, so set it false now

gblbHangarDoorOptControl = 0;

// force the selfactions menu to re-eval

gblObjLastHangarDoor = objNull;

};

};

};

//-------------------------------------------------------------------------------------------------------------------------------------------------------

// end large metal floor / hangar door options

**Step 6.5 -- HDAS\_Client folder**

The remainder of the Hangar Door system is contained within the HDAS\_Client folder. Copy that folder intact to your root mission folder.

USAGE instructions can be found at <http://ericpeterson.info/doors.txt>