

## Additional Admin Variables

**(By: \$or)**

Hi guys,

I put this little mod together while working on another project, but I decided to release it seperately for all those who might benefit from this.

Basically this mod adds extra functionality to ClientAdmin players with 4 (technically 3) new variables.

Here's the readme, I recommend reading it 😊

Code:

[illegible]

If client logs in as ClientAdmin, his player entity will be tagged with the following variable:

```
.isAdmin
```

NIL => no admin

```
1 => authorized and recognized as admin
```

```
.adminLogin
```

NIL => error (check console)

```
"" => no login data has been found for detected admin (check console)
```

```
string => name that client used to log in the clientAdmin system
```

```
.adminRights
```

NIL => error (check console)

```
0 => no rights found for specified ClientAdmin? (check console)
```

```
int => nonzero numeric rights of the specified ClientAdmin
```

```
.adminAccess = array (of the ClientAdmin's ACCESSLEVELS)
```

NIL => error

```
size 1 array => ACCESSLEVEL MAX rights for the specified
```

ClientAdmin

size X array => slot 1 to slot X will sum up the calculated

ACCESSLEVEL rights for the specified ClientAdmin

[illegible]

## INSTALLATION REQUIREMENTS:

Reborn 1.12 Patch Release Candidate 3.105 or higher.

MoHAA server

INSTALLATION:

1. place the .pk3 file in the main/ directory.

```

2. add/combine the supplied DMprecache script with yours;

    exec global/adminMain.scr

3. add/combine the connected event (global/events/connected.scr)
with yours, only 3 lines need to be copied/added:

    if (level.adminFunctions) {
        thread global/adminMain.scr::newPlayer
local.player
    }

4. use described vars above in your scripts!
5. advanced users may just use the adminFunc.scr script or
functions for their mods.

////////////////////////////////////
////////////////////////////////////
//////////////////////////////////// N O T E S
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

Prefab for parsing the .adminAccess player variable:

    if (<player>.adminAccess) {
        // absolute MAX, more than all available ACCESSLEVELS
added together.
        if (<player>.adminAccess.size == 1) {
            // do stuff
        } else {
            local.start = 1 // start at 0 if you want to
include "ACCESSLEVEL_LOGIN", else use 1
            for (local.i = local.start; local.i <=
(<player>.adminAccess.size - local.start); local.i++) {
                // do stuff
            }
        }
    }

IF one of the following situations apply to you:

'Help! Console is displaying errors when trying to access the new vars,
and player entity .isAdmin == 1??'
'Help! Console displayed the following error:
  ~^~^~^ ERROR|ADMIN_MAINFUNC[adminMain.scr]: Admin login detected,
but no data has been found by function 'findLastAdminData' using
searchrange: 1024! Could it be RCON login? ~^~^~^
  but the admin logged in via ClientAdmin??'

THEN you might want to:
1. open the .PK3 file and open global/adminMain.scr
2. find ' level.QCLog_searchrange ' and
3. read the comment above this line very carefully!

```

In essence I simply wrote a function which will, upon detection of admin, seek out its login data from the qconsole.log.

Reborn INFO: 127.0.0.1 (Sor) logged as admin using login "Sor" and password "xxxxxx"
--

To avoid lag, crashes and infinite loops, the function will go to the end of the qconsole file and start a specified range of bytes (default: 1024) before the EOF.

That's why the mod requires RC3.105, because these consoleprints have only been added then.

But after that it becomes easy, I just use the loginname to find the corresponding rights in admins.ini and decipher it.

Have only tested the functions, not the entire thing. So warn me if I messed up or forgot something.

**Q:** How does it detect boundaries?

**A:** In essence I use a custom trace function along the positive and negative X, Y and to a lesser extent, Z axes first along the Z axis upwards and once I retrieve X,Y estimations from doing that, I do the same along the X and Y axes as well. After that, I use all the outliers found in that data to compile the map boundaries. The custom trace function itself can trace along -X, X, -Y, Y, -Z & Z right through all geometric obstacles until it detects that it has gone outside of the map.

The way it works is I trace approximately 32000 units in a particular direction. Once outside of the map box, you can actually trace however far you like, even though there's no map that large. The edges of the map box are also regarded as geometric obstacles by the trace command (probably because of clipping), so once the length of the new trace indicates that the function has gone past the map boundary, the last trace is ignored and the length of all the traces it did so far, is returned. The reason I do this multiple times for each dimension is because renderless zones (like fake, locked houses at the edges of the playing map or maps with geometric 'bottlenecks') mess with the data (specifically my custom trace) by making the boundary into the unrendered infinity closer than the actual boundary of the map box.