

Newbie Wizards FAQ

Contents:

- [1.1 - Intro](#)
- [1.2 - So, how much do I have to know about MOO programming to run my own MOO?](#)
- [1.3 - I just got my MOO server going, now what?](#)
- [1.4 - How do I...?](#)
- [1.5 - How do I restrict creation of new characters to the @make-player command](#)
- [1.6 - How do I edit my welcome screen?](#)
- [1.7 - Why does mail to the Internet fail when I create a character?](#)
- [1.8 - What's the deal with @make-player? If you try to add alias arguments, they just concatenate into a long player name with no aliases](#)
- [1.9 - What's the deal with @make-player? There's no help](#)
- [1.10 - How do I make a trusted player into a wizard?](#)
- [1.11 - How come no one can connect as guest?](#)
- [1.12 - What should the new ArchWiz understand about permissions/security?](#)
- [1.13 - No player's disconnect date is getting modified, what's up?](#)
- [1.14 - LambdaCore is awfully bare. How can I get some other objects and system features into my MOO without recoding them from scratch?](#)
- [1.15 - I dumped this object, but it has a few hundred properties containing object numbers from objects and players on the source MOO. How can I get rid of these?](#)
- [1.16 - My client totally hashes my @dump script when I upload it, putting in linebreaks where it shouldn't. What do I do?](#)
- [1.17 - How can I make all new players children of a specific player class?](#)
- [1.18 - Why do I get a range error on line 3 of #15:length_date_gt, which is used by commands such as @subscribe?](#)
- [1.19 - I've been trying to get @mail-options +netmail to work on my MOO and I always get this: Mail sending failed: 503 Need MAIL before RCPT. I then get both an emailed and @mail copy of the msg. What's up?](#)
- [1.20 - Whats up with this my_huh: security hole?](#)
- [1.21 - There is always new news. What's going on?](#)
- [1.22 - Where's the help on \\$news? I've @examined it, but nothing works.](#)
- [2 - Credits and Notes](#)

1.1 - Intro

This FAQ list is designed to help new MOO arch-wizards (MOO owners) get answers to their questions. Entries come from items on the moo-cows mail list, personal questions from experience on our MOO, and from your contributions. The questions on this list most directly pertain to the core database, LambdaCore; to the extent that the core db you start with differs from LambdaCore, answers to these questions may vary from what is presented here.

Some items below only pertain to the 21-Oct-93 LambdaCore.

≤-

1.2 - So, how much do I have to know about MOO programming to run my own MOO?

Some people might answer this way: If you're not a programmer with plenty of experience on MOO, creating your own objects and helping others fix theirs, you really should hire, bribe, or marry such a person before you throw up a MOO. You'll need this person as an ongoing partner. Many things just don't have any documentation beyond their MOO code. There isn't much chance of getting all the help you need from any mailing list.

≤-

1.3 - I just got my MOO server going, now what?

Telnet to the server on the port you started it on (You did read the included documentation on starting the server, yes? You did use the included restart script, yes?).

Now you should see:

```
Trying...
Connected to SERVER
Escape character is '^]'.
Welcome to the LambdaCore database.

Type 'connect wizard' to log in.

You will probably want to change this text, which is stored in
$login.welcome_message.
```

So you'll type "connect wizard" and you'll see:

```

*** Connected ***
The First Room
This is all there is right now.
You see a newspaper here.

```

At this point you can do almost anything. You'll first want to set the wizard's password (help @password). Then you can type

```

@display $login.
help $login

```

to see other properties that you might want to set immediately.

<-

1.4 - How do I ... ?

Wait! Did you 'read the manual'? The following are very useful:

```

help wiz-index
    [contains all of the currently existing wizard-help topics]
help core-index
    [contains all of the currently existing help topics about various core objects]
help $login
    [will tell you about welcome messages, blacklists, redlists, enabling automatic player creation (allowing logins to create players immediately, see next)]

```

<-

1.5 - How do I restrict creation of new characters to the @make-player command (ie. prevent create <playername> <password> from the login screen?).

```

@set $login.create_enabled to 0

```

This stops people from creating their own characters at the welcome message. If you have a network active, the default seems to be that people can still login as guest and create their own character with @request. We hacked the @request verb to change this, but that may not be the easiest of best solution.

<-

1.6 - How do I edit my welcome screen?

You can set the welcome message with something like:

```

@set $login.welcome_message to {"Welcome to...", "", "..."}

```

or

```

@notedit $login.welcome_message

```

<-

1.7 - Why does mail to the Internet fail when I create a character?

Make sure you compiled the MOO server with "outgoing network connections" enabled (in the options.h file); you may also have to:

```

@set $network.active to 1

```

The network is necessary so the server can automatically e-mail the new user with their character name and password. You should also set the properties on \$network correctly for your MOO.

<-

1.8 - What's the deal with @make-player? If you try to add alias arguments, they just concatenate into a long player name with no aliases.

The aliases argument seems to be broken in the 1-Oct-94 core. Add aliases later. Aliases work with this command in the previous version.

<-

1.9 - What's the deal with @make-player? There's no help.

[The following help is from the help database on LambdaMOO, and is out in the current core, but you don't have that or you wouldn't ask, eh?]

```
>help @make-player
Creates a player.

Syntax: @make-player name[,aliases] [email-address [password]]

If no password is given, generates a random password for the player.

Email-address is stored in $registration_db and on the player object.

Comments can be added by specifying the email address "email@address comment"
```

Example:

```
@make-player George "sanford@frobozz.com George shares email with Fred Sanford (Fred #5461)"
```

If the email address is already in use, prompts for confirmation. (Say no, this is a bug: it will break if you say yes.) If you say no at one of the confirming prompts, character is not made.

If network is enabled (via \$network.active) then asks if you want to mail the password to the user after character is made.

[From Judy Anderson, who wrote the help for this on Lambda]

<-

1.10 - OK, I need some help here. How do I make a trusted player into a wizard?

Don't. Instead, create a fresh player #p and do

```
@chparent #p to $wiz
@programmer #p
;#p.wizard=1
```

and then tell your helper the password to the new player. Alex Stewart notes that adding the @programmer call avoids certain sticky situations that might be encountered by non-programmer wizards.

The reason you really only want to do this with a freshly created player is that every verb owned by the player will suddenly acquire wizard permissions. For an existing player, you'd need to go through and make sure that player in question owns no verbs that will do any damage if given wizard permissions (remember, anyone may call any verb...). [from Roger Crew]

<-

1.11 - How come no one can connect as guest?

Do

```
@make-guest <guestname>
```

then edit the description. Only ONE of the guest characters must be named or aliased as 'guest' for guest connections to work. Try this:

```
@add-alias guest to <object# of first guest you made>
```

Repeat @make-guest <guestname>, adding descriptions as you go, until you have a number of guests equal to your desired maximum number of guest connections.

<-

1.12 - What should the new ArchWiz understand about permissions/security?

A lot. For now refer to #34443 on LambdaMOO (lambda.moo.mud.org 8888). Or you can look through the moo-cows archive available [MOO-Cows](#) for an article with the following title (June 1994):

```
Newbie wiz FAQ: Permissions/security
```

Here's some code you could put into a verb to help you find writeable verbs that act with wizards authority (i.e. a verb anyone could modify, that has the power of a wizard):

```
(perhaps call it: "@seek*holes findholes" none none none)

count = 0;
wiz = "";
```

```

evil = " contains WWV";
for object in [1..tonum(max_object())]
$command_utils:suspend_if_needed(0, tostr(object, "..."));
object = toobj(object);
if (valid(object) && length(vrbs = verbs(object)) != 0)
  for n in [1..length(vrbs)]
    $command_utils:suspend_if_needed(0, tostr(object, "..."));
    if (index(verb_info(object, tostr(n - 1))[2], "w") > 0)
      count = count + 1;
      if (verb_info(object, tostr(n - 1))[1].wizard == 1)
        wiz = "WIZARDLY ";
      else
        wiz = "";
      endif
      player:tell(wiz, "Object ", tostr(object), evil, " ", vrbs[n]);
    endif
  endfor
endif
endfor
player:tell(count, " total.");

```

[contributed by MudOp@ripco.com, and modified by Scott Lynch]

Until you really grok permissions, it may be best to do all your programming as a programmer character.

<-

1.13 - No player's disconnect date is getting modified, what's up?

(aka No one is getting mail notification, what's wrong?)

There seems to be a problem with the verb disfunc and confunc verbs on #6, generic player on the distribution of LambdaCore-21Oct93.db. To fix it, @edit #6:disfunc and also #6:confunc, and change each line 1 to this:

```
if (valid(caller_perms()) && caller != this && caller != #0)
```

Of course, this only works for sure if you're using the above db version, and haven't touched your disfunc and confunc verbs on this object. Be careful.

<-

1.14 - LambdaCore is awfully bare. How can I get some other objects and system features into my MOO without recoding them from scratch?

[Hmm. Well in the sense that you don't get some system features (weather, rpg system, diurnal environment, etc.) or objects (talking robots, morphing player class, etc.) that you've seen on established MOOs I guess LambdaCore could be called bare. LambdaCore is really a pretty complete base for coding a MOO, with LOTS of underlying necessities that some would be bored and frustrated to tears to have to code themselves. As more daughters of the LambdaCore appear I'll list them here. OK, enough editorial on with the question. ed.]

The recommended procedure for porting objects presently has two components: social and technical.

- Social: There are copyright & courtesy issues involved with porting other people's code to your MOO. The practical advice that results from these considerations seems to be:
 - If the object/code explicitly states you can copy it, or you have secured the author's permission, port it.
 - If not, don't port it.

This can be a bit frustrating if you want object xyz but xyz's author hasn't logged on in 8 months. The suggestion at that point is to look at the code and try to understand what is being done, then program an object with similar but improved functions.

Many people say porting MOO objects is a good way to learn MOO programming; others are hostile toward the idea of non-programmers porting objects at all. Let the reader decide, but understand that not everyone thinks porting is a great thing, even when you are courteous, and respect copyright.

- Technical: Currently there are lots of gotchas to porting objects, most of them related to coding irregularities on the original objects, and to MOO client operations. Only a couple of those are dealt with here; until someone develops a robust portability system for MOO, you *will* run into problems from time to time. When that happens, there is nothing like being, or being good friends with, an accomplished MOO programmer. Here is one recommended procedure:

1. On the MOO you're porting the object to, @create using the appropriate parent, and note down the new object number (#xxxx).
2. Turn on recording and turn off autowrap on your client; you do this to record dump your about to do on the source moo.
3. On the MOO you're porting object #nnnn from:
 1. @wrap off
 2. @dump #nnnn with create id=#xxxx
4. Edit the dump you recorded.
 1. Remove the @create from the first line from the first line of the dump, since you've already created the object.

2. Scan for hard-coded object numbers other than the one that is the object you created. Fix those, pointing them to the correct object on your MOO, or recoding so they're not needed.
3. Scan for things that might have inadvertently been captured along with the dump, e.g.:

```
You sense that Nern is looking for you in the Living Room:
Nern pages, "Come out and help me torture some guests."
```

5. Upload the dump to your MOO with your client; this dump script will put the properties and verbs on the object.
6. Now, test and/or debug the object.

[Thanks to Judy Anderson, for some of the nitty gritty details above; if something is in error, it is one of my ad-libs, -ed.]

<-

1.15 - I dumped this object, but it has a few hundred properties containing object numbers from objects and players on the source MOO. How can I get rid of these?

One obvious way is to edit them out of the dump with a text editor before uploading. This could cause errors and may be tedious. It might be better to redump the object, adding noprops to your @dump command. Then scan the verb code for what properties need to be on the object; add them back with @prop.

<-

1.16 - My client totally hashes my @dump script when I upload it, putting in linebreaks where it shouldn't. What do I do?

Check the @dump script file to make sure *it* doesn't have line breaks where it shouldn't. If it does, the problem is not with your upload but with your procedure for @dump (you did @wrap off, no?), with your client recording, or with the editor you used to scan the script; one of those or something else is breaking the lines.

If your file is OK the problem really is with the upload:

- Set your client to stop this egregious behavior, or
- Get a better client.

Two clients recently noted as being good for uploads are:

- Mac - Mudweller 1.2 has file upload capability [per Scott Lynch]
- Unix - emacs

```
M-x telnet (to open a telnet session in the current editing window)
```

```
C-X I (to insert the ascii file of the dump script)
```

[per Dan Mehkeri]

<-

1.17 - How can I make all new players children of a specific player class?

Set \$player_class equal to that class. For example, to make every new player a builder class, you'd want:

```
; $player_class = $builder
```

<-

1.18 - Why do I get a range error on line 3 of #15:length_date_gt, which is used by commands such as @subscribe?

You encounter this when there are no messages in #18, the Player-Creation Log. This is typical when a LambdaCore-based MOO is brand new. Change the verb to this:

```
#15:length_date_gt this none this
if (this:ok(caller, caller_perms()))
  date = args[1];
  if (length(this.messages) < 2)
    return 0;
  endif
return this.last_msg_date <= date ? 0 | this.messages[2] - this._mgr:find_ord(this.messages, args[1], "_lt_msgdate");
```

```

else
    return E_PERM;
endif

```

[Courtesy Fran Litterio via post from Pavel Curtis]

<-

1.19 - I've been trying to get @mail-options +netmail to work on my moo and I always get this: Mail sending failed: 503 Need MAIL before RCPT. I then get both an emailed and @mail copy of the msg. What's up?

This appears to be a bug in the Solaris (SunOS 5.x) sendmail daemon). Everyone may want to make the following change anyway, as it makes the MOO's sendmail routines more technically compliant with the SMTP spec.

Edit \$network:raw_sendmail and change the following (line 48):

```

if (!index("23", line[1]))

```

to:

```

if (line[4] == "-")
elseif (!index("23", line[1]))

```

[Courtesy of Alex Stewart]

<-

1.20 - Whats up with this my_huh: security hole?

A security hole in the :my_huh process (the part of :huh parsing which handles feature objects, in LambdaCore) has been discovered, which permits player class owners to run feature commands with their children's permissions. This is an undertandably bad hole, especially if your MOO moves functions from player classes and moves them onto features, and uses caller_perms() to check permissions.

The permissions checks on \$player:my_huh reads:

```

if ((caller != this) && (!$perm_utils:controls(caller_perms(), this)))
    "Standard permissions check";
return E_PERM;
endif;

```

.. and then goes on to do a

```

set_task_perms(this);

```

later in the verb.

It is recommended to either change that if statement to remove the (caller != this) check, or leave it in and changing the set_task_perms() to a set_task_perms(caller_perms()). How your MOO uses :my_huh would be the best way to tell which is the better change.

[Courtesy Seth Rich]

<-

1.21 - Hey! There is always new news. What's going on?

Never mind. Do this as the wizard on your 1-Oct-94 core:

```

; $news:_fix_last_msg_date()
; $news.last_news_time=0
; $news:set_current_news($news.current_news)

```

<-

1.22 - Where's the help on \$news? I've @examined it, but nothing works.

Here is how to do it with your 1-Oct-94 core:

First, find the object number for \$news:

```

@d $news

```

To add new news:

```
@send *news      (exactly like sending mail to any mailing list)
```

Once you are finished composing and sending your message to *news, do:

```
@mail on *news    (that'll show you what message has what number).
```

To add that message, do:

```
@addnews [number of message] to [object # of $news]
```

If a message is old and you want it removed, do:

```
@rmnews [number of message] from [object # of $news]
```

[From posts to Moo-Cows by Seth I. Rich and Colin Moock]

<-

2 - Credits and Notes

This document was originally written by kcary@pepperdine.edu. However, it has not been changed since around 1998 when it was mirrored onto this server. However, change is always good.

To contribute to or comment on this FAQ, send email to kenny at the-b.org; corrections, additions, attributions gratefully received. I haven't found a way I'm comfortable with yet to add db patches for things that don't affect everyone using the vanilla lambda core.

To ask a question about MOO programming, subscribe to moo-cows (to [un]subscribe to the moo-cows mail list send email to moo-cows-request@moo-cows.com with your nice polite request in plain english). The URL to obtain this list is:

<http://www.moo-cows.com/>

This document does not necessarily reflect the opinions of the folks I work for; they are not responsible for mistakes, libel, or other untoward effects of other statements contained herein. Follow any procedure above at your own risk.