

SquirrelMud World Builder's Guide

Mike Buland

SquirrelMud World Builder's Guide

by Mike Buland

Copyright © 2005 Xagasoft

Table of Contents

Introduction	v
1. Concepts	1
1.1. Zones	1
1.2. Things.....	2
1.2.1. Name and ID.....	2
1.2.2. Weight.....	3
1.2.3. Size	3
1.2.4. Worth	3
1.2.5. Durability.....	3
1.2.6. Descriptions	4
1.2.7. keywords.....	4
1.2.8. Types.....	5
1.2.9. Built-Ins	6
1.3. Creatures	7
1.3.1. Standing Description	7
1.3.2. Species	8
1.3.3. Sex	8
1.3.4. Hit Dice	8
1.3.5. Type	8
1.4. Rooms	9
1.4.1. Exits.....	10
1.4.2. Surfaces	10
1.4.3. Contents	11
1.4.4. Extra Descriptions	11
1.5. Room vs Zone Resets.....	11
1.6. Online Creation	12
1.7. Player Ranks	12
2. The Tools	14
2.1. Virtual Workspaces	14
2.2. Zone	14
2.3. Room.....	15
Glossary	18

List of Tables

1-1. Ranks13

Introduction

Welcome to the magical world of SquirrelMud World-Building. If you're reading this you're probably at least World Builder class or higher on a SquirrelMud server.

This book covers how to interact with the server to build game elements, and also covers conventions used in other areas to make everyone's life a little easier, and make the game flow well.

Even if you are familiar with other mud systems, I still recommend at least glancing through the concepts section. While a lot of the concepts come originally from the standard MUD way of doing things, a lot of them have also changed. We started this project with the thought that a MUD is cool, but antiquated, and anything that could be made to work better, should be.

Chapter 1. Concepts

This chapter covers some of the important base concepts and terms that will be used throughout the book and in the game. Most of these are things that players may take for granted, but will be very important for World Builders to understand.

1.1. Zones

One of the most important concepts to understand when starting to build world is the Zones. These are a little bit different than zones in other MUD systems, but they fulfill the same basic purpose.

You can think of a zone as a container that holds other game objects such as Creatures, Things, and Rooms. Zones are however different from all other entities in that they are completely intangible to the player. They don't effect game play at all. Zones are purely a tool to make World Building easier.

The first thing that Zone's provide is organization and security. The list of users allowed to work on a zone limits the number of people who can make mistakes or "mess" with someone else's area.

Zones also provide what's commonly known as a name spacing mechanism. This means that the name of the Zone becomes a part of the name of everything within that Zone. You can think of it as the city on a person's address. This is important because SquirrelMud adds the Zone's name to your objects automatically, which allows you to edit quickly and seamlessly.

Name spaces also mean that different objects can exist in different zones with the same IDs. Every Zone can have it's own Dog creature, or a room name Elm Street.

If you're working on Salourn, and you make a creature with the id "Dog", you can use it in any room in the Salourn zone by just entering the id "Dog", the game assumes that "Dog" has been defined in Salourn. In order to access a creature named Dog from the zone Pheligir in a room in Salourn you would need to preface the ID with "Pheligir" and two colons.

```
<contents>
  <creature type="Dog">
  <creature type="Pheligir::Dog">
</contents>
```

If this code appeared in a room in Salourn the first line would load the Dog creature from Salourn, and the second would load the Dog creature from Pheligir.

This same mechanism is used to connect rooms via exits across zones.

```
<exit dir="north" target="Elm Street 04">  
<exit dir="south" target="Pheligir::Elm Street 01">
```

As you can see here, this room, which is labeled as "Elm Street 05" in the Salourn zone connects on the north to "Elm Street 04", also in Salourn. To the south it connects to "Elm Street 01" in Pheligir.

There is never any problem using creatures, things, or connecting to rooms that aren't in one of your own zones except that you may not have access to edit it, and it may disappear at a later date. Be careful about using objects that are in other Zones.

1.2. Things

A thing in the world of SquirrelMud is anything that the user can interact with that isn't a Creature or Room. This includes, but is not limited to weapons, armor, clothing, containers, paper, maps, pens, magic accessories, and more.

No matter what type of Thing you contemplate, all things share a few common properties. These include name and id, weight, size, worth, durability, descriptions, keywords, built-in, and type. These are the most general concepts and probably the easiest to describe, so let's do that now.

1.2.1. Name and ID

The name of a thing and the ID are closely related, but do not have to be the same. The name of a thing is what is displayed on the screen when a user interacts with it. The ID is what Builders use to reference Things, and must be unique.

This distinction can be better understood through example. A World Builder may want to conceal the true identity of, say a set of magical rings by giving them a more generic name, such as "Brass Ring". The player may never realize that the ring is magical without doing more research on the subject in game. At this point it would make sense to at least make the ID something different, like "Magical Brass Ring" to distinguish it from any other rings you may have made. Since the player will never see the ID, it will make the World Builder's life easier and give nothing secret away.

This can also be taken advantage of if you want to have several different magical rings that all look the same. You may make three objects all named "Brass Ring" but give them ids that match their magical ability like "Brass Ring of Strength", "Brass Ring of Energy", and "Brass Ring of Speed". Now the World Builder can easily tell the rings apart while they work on the game, but until the player gets the ring identified they'll have no clue that it's magical or what it does.

This is a good time to note that IDs may contain any letters or symbols, including spaces, and be any length. They must be unique (the game will tell you if you try to make one that isn't unique), and are processed in a case insensitive manor. This means that "Magic Ring" is the same as "mAgIc RiNg" to the game.

1.2.2. Weight

Weight in game is expressed roughly in US pounds. This is done simply to make life easier on us here, since we're all in the US right now. (For other nationalities, or measurement systems that make sense, refer to the installation notes for how to change this.)

You can enter weights in decimal notation, allowing you to create objects that are fractions of a pound, or at least very specific. There is no upper limit on weight, but you cannot make a weight less than zero.

1.2.3. Size

Size is probably the single most abstract property of a Thing. It is expressed as a single decimal number instead of as dimensions. This may seem odd, but if we introduced dimensions than part of the game would have to become organizing your knapsack, which would be boring and difficult to handle through a terminal interface. Instead, think of the size as a total number of cubic inches. If you have a five foot sword you could easily say that it's two inches wide and one thick, so $5*12*2 = 120$. This is an odd scheme, but one that makes life much easier in the game world.

1.2.4. Worth

Worth is expressed as a single number, and can be thought of as the raw dollar value of an object. This value may never appear directly in the game world, since each currency also has a worth, but it's important to set, this is how shops figure out how much to buy and sell for, and how the game calculates a player's worth.

1.2.5. Durability

Durability defines how tough the thing is. That is, how much damage it can sustain before it's completely unusable. This is expressed as a decimal number which gets smaller the tougher the thing is. A number less than one, like 0.5 is much tougher than 1.0, or 5.0. The range for this number is 0.0 to 100.0. 0.0 means that the thing is invulnerable and will never be destroyed. This should never be specified except in very, very rare and specific circumstances, as it will impede the game's natural cleanup mechanisms. A value of 100.0 means that the thing is as fragile as it can get and can only sustain one point of damage.

A more mathematical explanation may help anyone who wishes to follow. It's not really very complicated. All things start their life with 100 hit points. If they are attacked, used in an attack, help defend an attack, or are left on the ground they suffer damage. Every point of damage a thing suffers is multiplied by their durability before being subtracted from their hit point total.

This means that if they have 100hp and their durability is 1.0 they can take 100 points of damage ($1.0 * 100 = 100$). However if they have a durability of 20.0 they can only take five points of damage ($20.0 * 5 = 100$).

On the range below zero this is also true, and effective. With a durability of .5, a thing effectively has 500 hit points, with a durability of .1 it effectively has 1000. This scale was made this way so that there is effectively no upper limit to how tough a thing is.

1.2.6. Descriptions

Description is what really drives the fantasy and role playing elements of the game. All descriptions are visible upon request (if the player looks at the thing). And are displayed with minor formatting applied. For more information about how to write descriptions please see the style guide.

1.2.6.1. Line Descriptions

Things contain a type of description that doesn't show up anywhere else in the system, and follows a very specific formatting. The line description should only be a couple of words, maybe five or six max, and will be inserted into the statement:

```
[A/An] [your words] [is/are] on the [surface].
```

The system will decide weather it should use A or An at the beginning, and if it's plural or not based on weather or not [your words] ends with an 's'. The [surface] is whatever you set it down on in the room last.

A few examples of line descriptions include: "tubular piece of metal" or "dull, plain, silver fork". Try to not make these too long, and try it out before you use it in the world.

1.2.7. keywords

Keywords are very, very important because without them no player could actually interact with the game in a meaningful way. Keywords define what terms the player can use to interact with a thing. These are single words, and, like all game commands, the player can use shortened versions of them.

Selecting keywords that fit well can be difficult. It's not always easy to guess what the user will try to use to pick up or use a thing. Generally, it's best to wait until you've written the description and line description

1.2.8. Types

Types are what make things really fun and useful in the SquirrelMud system. Types provide more ways of interacting with things and allow you to really do some fun things with the game.

There is way too much to describe in this brief overview regarding what each type can do, so I've devoted an entire chapter to configuring and using things later on in the book. Check that out for all your type needs.

1.2.8.1. Misc

Misc or miscellaneous types are the most basic type of thing. The misc type provides no extra functionality and is mainly for treasure, garbage, interest items, keys, and anything else you don't want people to interact with in the ways listed below.

To make treasure or gems, simply make a misc thing with a high value. For garbage, do the opposite, make the value next to nothing, or nothing. Interest items can be setup however you'd like, and keys, well, you should read the section on exits and doors to find out how to make a thing a key.

1.2.8.2. Weapon

Weapons are one of the big things that makes a game like this fun. A weapon thing is anything that can be used as a weapon...such as a sword, club, stick, stalactite, doll, rock, ax...anything.

Weapons have several properties that make them interesting, including what type of damage they can inflict (bludgeoning, piercing, slashing), how much damage they can do on their own, and how to describe the damage they inflict to the player.

1.2.8.3. Wearable

Wearable covers the most ground of any type so far. A wearable can just be a shirt, some pants, a dress, full plate male, a helmet, fancy boots, or even jewelry.

A wearable can be worn on any combination of more than 25 different places on the body, and in any of three layers of clothing. Outer layers will cover up inner, and full coverage will keep a player from receiving as much damage as they might otherwise.

Making standard clothing involves making clothing that can't withstand very much damage (high durability score) and also doesn't protect you very well (low protection). Armor is usually the opposite of this. Jewelry should have absolutely minimal protection, since it will rarely be taking the brunt of an attack.

1.2.8.4. Container

A container is any thing that can hold other things. Containers can only hold other things. We may add a cage type later that can hold a creature, but there isn't one yet.

All containers have maximum weight and capacity restrictions, as well as their own weight and size. You can also make containers expandable or not. This means that they may start small, like a cloth sack, but increase in size as you put more in them. In other cases the size of the container is absolute, like a wooden box, and will not grow no matter how much you place within.

A container will always weigh as much as the original thing, plus everything within it. The maximum capacity for a container is the size of the container itself for non-expandable things, and arbitrary for expandable things.

All this really means is that containers behave how you would expect them to in real life. You cannot make a magical container with the base Container type at this time, although you will be able to very soon.

1.2.8.5. Vessel

Vessels are just like Containers but can only hold liquids. These are things like canteens, water skins, barrels, fountains, lakes, etc.

You can specify a capacity, and weight, but the size never changes, and the weight when full is determined by the kind and amount of liquid contained. You can also specify how much liquid and how much liquid a container starts with.

It is possible to make a container never run out of liquid by setting the capacity to zero. This is not recommended for anything but fountains and rivers, since the player will not be able to change the liquid in the container, but can be fun for the occasional magical prize item (of impressive value, of course).

1.2.9. Built-Ins

Built-In things are very useful for a number of reasons. They allow you to create interactive things in rooms that the player cannot discern from the background of the room, but can still interact with. Players

may not take built-in things, or even try to. They can however interact with them in any fashion that is normally allowed for the thing type. Built-In things also do not show up in the visible thing listing, and should be described in the room description. This is the only case where this is actually safe, since creatures cannot take the thing out of the room.

The best two examples that I can think of for this type of thing, but certainly not the only two uses, are for a chest or locker, or a river or stream.

For a chest or locker you would make a standard container thing, mark it built in, and put it in the contents of a room. Once this is done, every player and creature can interact with it just like a normal container. They can open it, close it, unlock it, lock it, take things out of it and put things in it. They cannot take it, so it's a nice large storage mechanism for bedrooms or even castles. Especially for treasure when you don't want people to even try to take it or the container until they have the key.

For a river, stream, or even fountain you can make a very large vessel and fill it with whatever you'd like. Then make it built in and leave it in a room, describe it however you would like. Now players and creatures can drink from it and fill containers from it, but not even try to take it. (in other games it would tell you the river is too heavy.)

1.3. Creatures

Creatures in SquirrelMud encompass everything that can move around the system and interact with the world on their own. They are active, where things and rooms are passive. In many systems creatures would be known as NPCs or Non Player Characters, except that in SquirrelMud the player is a creature as well, which means that every rule that applies to them, applies to us.

Some of the elements that creatures possess are the same for rooms and things as well, but much is different. The things that you should take note of that are different are the standing description, the species, sex, hit dice, and type.

1.3.1. Standing Description

Standing descriptions are pretty simple to understand, especially if you've played the game. Whenever a creature is standing in a room, this is a one-line description that the system displays verbatim for the user after the room description.

These can be things such as: "A small puppy dog is sniffing about here." or "A cute little bunny is sitting here." The text does not need to describe what the creature is doing while standing, just while awake and wandering around.

In the future there will also be a sleeping description, but right now creatures can't sleep, so it's not important.

1.3.2. Species

Species is really just a reference to which species in the system you want this creature to be. Species sets quite a few parameters by default to each creature. To find out more check out the species section coming up.

1.3.3. Sex

The sex of a creature is purely cosmetic in game. There is no statistical distinction between male and female. All it really does is change the gender of the words that the game uses to describe what a creature is doing.

There are three genders that the system recognizes: male, female, and neuter. These correspond to the terms he/him/his, she/her/her's, and it/it/it's. While building you can also select "random" as the gender of a creature to allow the system to choose between male and female when generating creatures. The system will never choose neuter when randomly selecting.

1.3.4. Hit Dice

Hit Dice, is, unfortunately, an anachronism in the current system. It is still being used, but will soon be replaced by the new leveling mechanism. Still use this, but be aware that changes are imminent.

This defines the rough level of the creature. The species defines how large of a die the system rolls to determine how many hit points a creature has, this property defines how many dice to roll. For example, humans get to roll an eight sided die to determine their hit points. If you set the creature's hit dice to 5 and their species to human they will roll 5 eight sided dice and add up the values to figure out how many hit points they get total.

1.3.5. Type

The type of creatures works very much like the types of things. The basic, Misc, creature should be good most situations, but sometimes you need a creature to behave in very specific ways that you can't get with the basic type.

1.3.5.1. Misc

The Misc or miscellaneous type of creature is the most basic and easy to use. By default it will simply wander around the world aimlessly and mind it's own business. If it is attacked, it will respond, but it is otherwise passive.

You can configure the misc creature type in a number of ways, including making them scavengers (they'll pick up things they find), greedy (they'll loot corpses), aggressive (they'll pick fights), and more. You can also provide them with additional user-space commands that they can execute randomly of their own accord. With this you could easily add interest by making dogs bark periodically, and scratch their ears.

1.3.5.2. Shopkeeper

Shopkeepers are probably the second most important creature type, if not the most important. Shopkeeper creatures encompass everything about shops in one easy package. Well, everything except store storage.

A shopkeeper can do everything a Misc creature can do, and can also sell things that you place in a special container. Shopkeepers can move around, but must start in the same room as their shop's container. Everything in the container is for sale in the shop, and anything they buy goes into the container.

Shopkeepers do not have an unlimited money supply. They start with what you give them, and that's basically all they get. They can sell things to make more, but when they're out of cash they can't buy anything else from you.

You can also configure shopkeepers to want certain types of things more than other types, and even look at the thing's description to see if they want it more or less.

Finally, you can tell shopkeepers how much markup and markdown to apply to all sales and purchases, respectively. This is important so they always make at least a little profit.

1.4. Rooms

Rooms, unlike creatures and things, only have the single active instance of each room. There are no room templates, when world building you edit the live rooms directly.

The term room is actually a little bit of a misnomer, and I believe comes from the days when the first MUDs were entirely in dungeons. A room in SquirrelMud can be outdoors, it can be in a tree, or deep

underground. A "room" is actually any place in the game where a creature or thing can exist. Rooms connect to one another through "exits" which can be empty space, like in an outdoor pathway, or have doors attached which can be opened, closed, and locked.

Rooms have many properties, most of which don't exist in any other entity. The most notable of these are exits, surfaces, contents, and extra descriptions.

One thing that makes SquirrelMud both entertaining and tricky is the fact that rooms do not have an absolute position. Their position can only be defined in relation to the rooms around them. That is, no room has what could be considered grid coordinates, or latitude and longitude. This means that, while you could draw a map of an area, there is no way in-game of determining where you are on the globe. This also means that distances between rooms are arbitrary and can be made up as you make rooms. In addition, you can seamlessly insert and rearrange large portions of the world simply by changing where an exit leads.

1.4.1. Exits

Exits are what link the world together and allow creatures to travel from place to place. An exit on it's own represents an open hallway, doorway without a door, or possibly empty space in the case of an outdoor area.

You can configure exits to have doorways, and those doorways can have locks on them. You can put an exit into any state by default, which is the state it will return to when the room it's in resets.

Each exit is in actuality a one-way passage. A doorway leading south from room A to room B will not allow you to travel from room B to room A, and will, in fact, show no sign of it at all in room B. A two-way or bidirectional exit is really two exits, and is the most common form of connection between two rooms, although one-way connections have their uses.

Exits may also contain what is known as a transitional description. This is a description that is shown to the player whenever they traverse the exit. It can be very useful for describing changes that occur while traveling such as bends in the road, traveling up stairs, falling down slopes, and so on.

1.4.2. Surfaces

While exits and contents are important to the operation of the game, surfaces are purely aesthetic. They represent places in a room that things can be set on. This brings an extra element to the game, being able to set things on tables, beds, and so on instead of just dropping them on the ground all the time.

Every room must contain at least one surface, and the first surface in the room is the default surface. That is, the first surface is the one that will be used if someone drops something in a room instead of placing it

on a specific surface. The default should always be something like floor, carpet, ground, grass, and so on.

Try to restrict surfaces to only one or two words, usually only one, a noun. This is for several reasons, but it also makes it harder for the user to interact with the surface since, at least for now, only the first word of the surface is used in command lines to interact with it.

Things are always described as being "on" your surface. This may seem restrictive at first, but makes sense if you play with it for a little while, and with built in containers and other elements of the game you can still do just about anything you want.

1.4.3. Contents

Room contents describe what will be in a room, or what should be in a room, but not necessarily what is in a room. The content definitions are used when a room is reset or when the game is started. Each room will create everything that it's content definitions describe.

You can place things on surfaces, and within things that are containers. You can also create creatures and give them an inventory. When creatures are created in this way they will auto-equip anything wearable or weapon type that they own.

1.4.4. Extra Descriptions

While rooms have a base description just like creatures and things, they can also contain any number of extra descriptions. These are accessed by a list of keywords and the look command.

Extra Descriptions are used mainly to add interest to rooms and to make the base description less overwhelming if a room has a lot of interesting things to look at inside it. You can also use extra descriptions to give hints to people and post signs and things of that nature.

An example of a normal extra description would be if you mention something that's in a room, like a fancy wall hanging. You don't need to describe it past mentioning the hanging in the base description, and then describe it in much more detail in an extra description.

There should be clues in the base description or in another extra description of how to access every extra description. Not everything in a room needs to be described with an extra description, and not every room needs one. These are, however, very useful and add a lot of depth to the game.

1.5. Room vs Zone Resets

When an area in a MUD "resets" it's state is returned more or less to what the world builder intended it to be. Things that were taken are re-created, creatures that were killed are brought back to life, doors that were unlocked can be re-locked, and so on.

In most systems that I've used the world operated on a per-zone resetting mechanism. This means that periodically each zone of the game resets every room, thing, and creature within it to it's original state. This also provides us with a few problems. We don't want to reset a room while a player is standing in it, and some rooms within a zone probably just won't need to be reset.

In SquirrelMud the game operates on a room-level resetting mechanism, which is handled efficiently in ways that we'll discuss in quite another guide. The end result of this is that each room can be reset independently of the rest, and the world builder then has a finer level of control over the entire process.

Rooms will not be reset while a player is in them, but since we operate on a per-room basis, that's ok. The system will simply wait until the given player leaves a room in order to reset it. It will recreate any creatures that were killed, but not any things that were taken unless very specific circumstances are met. Things are only recreated when they leave play and are no longer in the game. This can happen when a thing is sacrificed, junked, eaten, or destroyed by natural decay or through violence.

1.6. Online Creation

Online Creation, or OLC, is the system in SquirrelMud that allows world builders to interactively modify the world while the server is running, without having to learn the fairly complex datafile formats. It allows anyone with credentials to create new rooms, thing templates, creature templates, and generate things and creatures. Through it you can also change existing rooms and templates, and of course, delete them.

Online Creation currently exposes four commands to the world builder. Thing, creature, room, and zone. Each of these commands is used to interact with the corresponding element type. The actual format and usage of these commands is described later in this document.

These commands are also the only part of the system that are effected by zone security. If a player is not a member of a zone's worker list and is not the zone's steward they cannot edit or create anything within that zone.

1.7. Player Ranks

Every player, in actually every creature, has a rank. This is a value that defines what commands each

player has access to, and how the SquirrelMud system treats them. The list of ranks may change at any time, but a few fundamentals will never change.

The higher a player's rank, the more commands they have access to. Each rank provides more commands, but never takes them away, so gods can do everything mortals can do, and more. Below you'll find a list of the ranks in order from lowest to highest and what they mean.

Table 1-1. Ranks

Rank	Description
Trial User	A player that has not left the newbie zone. Their games will not be saved. This turns into mortal automatically when they leave the newbie zone.
Mortal	The standard player rank. All players who are there just to play will be mortal.
God	Gods are almost like police. They have limited ability to change creature's and thing's, and can reprimand players for breaking the rules in a number of ways.
World Builder	Identical to a god, but with access to the OLC system.
Greater God	Greater Gods have supreme control over reprimand and world modification. They also gain additional commands that allow them to interact with external stat tables and change the status of global quests and other things.
Implementor	Implementor rank should never be given out to unfamiliar people. It is the only rank with control over the server itself. The Implementor can shutdown and reboot the server. It is also the only rank that can create new zones and ignore zone security.

Chapter 2. The Tools

In this chapter we will introduce you to the tools that are used in world building and how you use them. We'll assume that you're already familiar with the way the game operates and how to interact with the game console. That said, there are four commands that currently make up the online world building system, or OLC. They each have many subcommands, each of which allows you to alter an aspect of the creature, thing, room, or zone.

In the guide, I will always use the longest form of the commands, but just like every other command in the world of SquirrelMud, you can shorten the OLC commands, and their sub-commands to their shortest possible name.

2.1. Virtual Workspaces

Each OLC command, creature, thing, room, and zone has what we call a virtual workspace. These are areas that are outside of gameplay entirely, and allow you to create new game entities without worrying about them interfering with the players. You can think of the virtual workspaces as a separate dimension in the game world that you can create new entities within.

When you wish to edit an existing game entity you must first "select" it. When you do this you effectively bring that entity into your virtual workspace. In the case of live game entities this means that it now exists live in the game and in your workspace, waiting to be operated upon. Any changes you make at that point are immediately translated to the game as well.

You have effectively four virtual workspaces, one for each command, so you can be working on one zone, one room, one creature, and one thing all at the same time.

2.2. Zone

Only players of Implementor rank can create, delete, or set zone stewards so most world builders will only ever select and browse a zone. This must, however, be done (the selecting) before you can use any of the other OLC commands. Everything in OLC is done in the context of the currently selected zone.

Here follows a list of zone uses and what they do.

> **zone**

Display statistics relating to the currently selected zone, if any, including who the steward is, how many workers there are, how many rooms, creatures, and things there are, and so on.

> zone list

Display a list of all zones that are currently loaded into SquirrelMud. The names listed here are the names that you use in global IDs and when selecting zones.

> zone select [zone name]

Selects a zone in the system, provided you have authority to select it, and places it in your virtual workspace. This is the first thing that you should always do before using any of the other zone commands.

2.3. Room

Room is quite possibly the most complex of all of the OLC commands due mainly to the number of sub-commands that require sub-sub commands. We're still working on making this part a little easier to use, but for now, try to bear with us.

> room

Display an overview of the currently selected room, if any.

> room list

List all of the rooms in the currently selected zone. The listings are displayed with the full global ID of each room. When selecting you do not need to enter the global ID, only the local ID.

> room list [zone name]

List all of the rooms in the given zone. You can do this even when you do not have authority to select a zone, otherwise it works just like **room list**.

> room select [room local-id]

Select a room in the selected zone and place it into your virtual workspace. Once this is done you can use any of the other room commands to edit it's properties. If you have a new room that hasn't been committed yet in your workspace, it will be thrown away automatically.

> room show [room global-id]

Show a display just like the plain room command does for the given room, specified by global or local id. This allows a world builder to view other rooms for reference without changing the selected room. This can be used to view rooms that you do not have the authority to edit.

> room name [name]

Change the name of the room. The name is shown to players verbatim, and does not have to match the ID. If an ID has not been set for this room already, using this command will also set the ID to the same value.

> **room id [local-id]**

Change the ID of the room. The ID you enter should be a local ID (no zone name or colons), and will be automatically turned into a global ID later. If you have not yet set a name, setting the ID will also change the name.

This is the only property that you cannot change once a room has been committed.

> **room exit**

List all available exits in detail. This includes the direction it leaves the selected room, the room it links to, transitional text associated, if it's one-way or bidirectional, and door status.

> **room exit [direction] [room id]**

Create or change an exit leaving the selected room. If an exit exists, in that direction it will actually be deleted and a new one will be added in its place. This is important because it deletes all extra properties that you've set, such as transitional text. The room id that you enter can be either a local id or a global id. You only need to use a global id if the room you wish to link to is in a different zone.

> **room bidirectional [direction]**

Convert an exit that exists in this room into a bidirectional exit. This will automatically create an exit in the room that the exit links to back to the selected room. It is safe to use this command even if you're unsure if the other, or target, room has an exit in that direction, it will not delete existing exits.

> **room bidirectional [direction] [room id]**

This works just like the above bidirectional command except that it will auto-create or replace an exit in the selected room first. You can think of it as working just like the standard **room exit** command, except that it will create an additional exit in the linked to room as well.

> **room transition [direction] [transition text]**

Add a transitional description to the exit in the given direction. Whenever a player passes through an exit, its transitional text is displayed for them.

> **room surface new [surface name]**

Create a new surface in the room. The name should only be one word, all lower case, with no punctuation. The system will do the rest for you.

> **room surface delete [surface name]**

Delete a surface that is in the room. Anything that was in the room, if it is live, and on that surface will be moved to the default surface, if any.

> **room surface default [surface name]**

Make the given, existing surface the first in the list, which makes it the default surface. Whenever a player drops something, it winds up on the default surface.

> **room content**

List all of the content in a room. This is done in a rough tree format for creatures and container type things that have contents of their own.

> **room content new**

> **room content type [index] [type]**

> **room content id [index] [thing/creature id]**

> **room content surface [index] [surface name]**

> **room extra**

> **room extra new**

> **room extra keyword [index] [keyword list]**

> **room extra desc [index] [description]**

> **room longdesc [description]**

> **room desc [description]**

> **room commit**

> **room delete**

> **room delete [room local-id]**

Glossary

A

B

C

Creature

A game entity that can actively interact with the world. The type of a creature can be changed to drastically change it's behavior. Creatures can be anything from bunnies and kitties to shopkeepers and daemons in the game. They do not have to be aggressive.

A player character is a standard creature who's type is set to the special type Player.

Creature Template

A template that describes how to generate a creature in game. You can only create creature templates through OLC, not actual creatures.

See Also: Template, Online Creation.

Creature Type

Any active entity in game. Creatures are the only type of entity that can interact with rooms and things. Players are in fact, a special type of Creature entity.

D

E

Exit

A one-way connection between two rooms. Exits can have properties like doors and transitional text applied to them. A two-way or bidirectional exit is, in reality, two exits leading in opposite directions between two rooms.

F

G

Generate

This is what happens when a thing or creature template creates a usable thing or creature. We say that the thing template generates a thing. This is done both when rooms reset and whenever a world builder requests it through OLC.

See Also: Online Creation.

H

I

J

K

L

M

N

O

Online Creation

A part of the SquirrelMud server that allows players who have been authorized to change the game while it is running, through a standard command interface. This allows the game to change while players are logged in and enjoying themselves, and enables the game designers to easily create a much larger and more dynamic environment.

P

Player

Any creature that has a person at a computer controlling it. The game cannot tell the difference between players and other creatures once they are created except in very special circumstances. There are some special rules that apply only to characters, but very few.

Q

R

Room

Actually any place in game where a creature or thing can be. Each room has a description and a set of properties, including exits to other rooms. A room can be outdoors, indoors, underground, in the sea or in the sky.

S

T

Template

Templates describe how creatures and things are generated. They contain descriptions, keywords, and other elements that apply directly to the thing and creature, but are not interactive.

A template also describes in what ways creatures and things may vary, for example, a Creature's stats will be randomly selected when the creature is generated, but not set explicitly in the template.
See Also: Generate.

Thing

Things rule!

Thing Template

A template that describes how to generate a thing in game. You can only create thing templates through OLC, not actual things.

See Also: Template, Online Creation.

Thing Type

Any passive non-room entity in game. Creatures interact with things in a number of ways, and help make the game interesting by allowing the creatures in the game to be more than just completely random, naked, punching machines.

U

V

W

X

Y

Z