scribe.doc

#### DD Area Scribe Users Manual

*Version 1.1.w, November 2000.*

# Contents

1. Introduction *What is Scribe? Why use Scribe?*

2. Usage *How to use the script.*

3. What's Perl? *What to do if you can't run the script.*

4. Producing source files *General format of Scribe source files.*

5. Area header and base vnum definition

6. Mobiles

7. Objects

8. Rooms

9. Mobile resets

10. Object resets

11. Helps

12. Shops

# 1. Introduction

*DD Area Scribe* or *Scribe* is a tool for producing formatted area files for the Dragons Domain Envy-style MUD. DD area files retain the Envy MUD format, with changes to the #AREA header information and the addition of new variables and flag values to other sections.

Scribe is a script that reads a simplified text source file containing all area information, validates the information, and if no serious errors are found, outputs a formatted area file. Scribe is written in Perl and is run from the command line (see Section 2).

Scribe was written as an alternative to writing area files by hand or by using the existing Windows tool *MZF*. The advantages and disadvantages of each of these systems might be summarised:

##### Writing areas by hand

Laborious; lots of redundant variables to include. Unsafe; no error or format checking.

##### Using the MZF tool

Easy to use; convenient and simple user interface. Format and variable checking provides some safety. Buggy; contains known bugs and bizarre behaviours. Incompatible with DD; MZF does not contain DD's special flags and values; output requires hand-editing. Win16/32 executable only.

##### Using the Scribe tool

Fairly laborious; all data hand written, although in a friendly, human-readable format. Safe; variable checking and correct formatting. Compatible; contains all DD special variables and flags. Requires Perl 5 for compilation.

Scribe meets writing areas by hand and using MZF midway, providing full DD-server compatibility and error checking, but not including a simple to use graphical interface.

# 2. Usage

Producing an area file with Scribe involves the following steps:

##### 1. Source file is written

The area source file contains all of the relevant information to be included in the final, correctly formatted area file. It is a text file with a simple, human-readable format that is easy to edit. Information can be added to the source file in any order.

##### 2. Scribe is used to check the source file and write the area

Scribe is used to process the area source file. It reads through the source file checking for obvious formatting errors and then validates this information, filling in any missing variables where it can. If there are serious errors then Scribe will indicate where they occur in the source file and then exit. If there are no serious errors, then the area is formatted and written to file.

Scribe is a Perl script that is run from the command:

scribe.pl *<source file> <destination file>*

For example,

scribe.pl kahsis.src kahsis.are

Area files traditionally have the .are extension. Both source and destination files must be indicated for the script to run. Scribe will create a backup copy of any existing destination file.

# 3. What's Perl?

If you are not able to run Perl but would like to use this script to produce your areas, you can write your source files and send them to one of the DD Immortals, who will test the file and inform you of any errors.

# 4. Producing source files

The area source file consists of discrete blocks of data that each define area information (area name, author, etc), one mob (mobile, creature), one room, one object, one mobile reset, one object reset, one shop or one help file entry. Some reset information (e.g. shutting doors and mob special functions) are included in the above blocks. Blocks may occur in any order within the source file.

A **block** consists of a **header** indicating the type of data it defines, followed by a list of **field** and **value** pairs. Fields are short keywords representing room, object, mobile or area variables. A block is terminated by the start of another block or the end of the source file.

**Headers** are single words preceded by a dollar sign $ and occupy one line. Field keywords and values are separated at least one space or tab (one tab is nice). Field/value pairs occupy one line unless they define multiple-line text blocks, in which case they are terminated by a line containing a single tilde ~. You should not terminate single line strings of text with a tilde: these will be automatically added by Scribe.

Blocks have the following format:

$header

field1 value1

multiple\_line\_field

*<lines of text>*

~

field2 value2

...

field*N* value*N*

For example, the following block defines the name, short name, long name, description, sex, affect flags, vnum, level and act flags variables for a mobile:

$mob

nm octopus

sh an octopus

lo An octopus crawls along the coral.

de

The octopus' slimy body is purple with bright red blotches.

It crawls over the coral, its tentacles writhing horribly.

~

sx female

aff hide sneak

vn 1

lv 5

act scavenger wimpy stay\_area

(Field/value pairs for each type of block are discussed in detail in Section 5 onwards.)

Field/value pairs can be arranged in any order within the block. Multiple definition of the same field can be made, although only the last instance will be used. The dollar sign $ at the beginning of a line indicates the beginning of a new block, so should be avoided.

Your source file will consist of a series of blocks of any type in any order. Use your favourite text editor to produce the source file. All text before the first block header is ignored, so you can add comments if you wish. Your source file must contain an **$area** block in order to be compiled without error.

Source file format:

Any number of comments at the start of the file.

$area *<area header information>*

$... *<mobile, room, object, etc blocks as desired>*

You may add comments at any point in your area file by prefixing them with a non-alphanumeric symbol that is not a dollar sign $, e.g. #, ; or whatever symbol you prefer.

. This is a comment

# So is this

; The line below is a header

$mob

. This is a comment within a $mob block

###### Note

Make sure that you hard-wrap lines of text in multi-line text blocks to under 80 columns, i.e. supply a carriage-return at the end of every line. Make sure your text editor isn't soft-wrapping your text!

# 5. Area header and base vnum definition

The **area header** block defines the #AREA section of the final area file. It describes the name and author of the area, and character access information. One definition is required, or your source file will produce errors.

###### Note on Vnums

*Vnums*, or *virtual numbers*, are the unique identification numbers for the mobiles, rooms and objects in your area. No blocks of the same type (mobs versus rooms versus objects) may share the same vnum; blocks of different types may share a particular vnum. Vnums are indicated as relative numbers in the area source file (usually beginning at zero).

Scribe uses *relative vnums*: the vnums used in the source file are numbered from 0 upwards. A base vnum is defined in the area header block and is used to calculate absolute vnums from the relative values:

Final vnum in area file = base vnum + relative vnum

If you have been allocated a range of vnums from the DD Immortals for use in your area (e.g. 2600-2799), use the lowest vnum as your base (2600), and number your mobs, rooms and objects from zero upwards. If you haven't been given a range of vnums, you should still number mobs, rooms and objects from zero up, and just use any value for your base value. The use of a base value allows you to easily renumber your area if necessary, e.g. when you are finally given some to use by those lazy Imms.

If you need to use vnums for rooms, mobiles or objects that are not defined in the same source file (i.e. are defined in other areas), you need to use temporary holding values and manually edit the formatted area file produced by Scribe.

**Header** $area

**Fields** *Field Description Type*

au author text

ti title text

ls lower level suggested number

us upper level suggested number

le lower level enforced number

ue upper level enforced number

bv base vnum number

Description au Author

Line of text

The person or people responsible for writing the area.

ti Title

Line of text

The name of the area. Don't make it too long (25 characters maximum).

ls us Suggested level range

Number: 0 or higher

The suggested level range for travelling to your area, as shown in the online *AREAS* command. Use these values to indicate what level of character would profit from visiting your area; don't use them to indicate the lowest and highest level mobs (1, 100 is fairly unhelpful).

le ue Enforced level range

Number: 0 or higher

The level range for permitted entry to your area. Characters outside this range may not access the area and get the ‘God prevents you from entering there’ message when they try.

Note: only the lower level limit is currently enforced.

bv Base vnum

Number: 0 or higher

The base value used to calculate absolute vnums from relative vnums.

Example

$area

ti The Planet Vulcan

au Mr Spock

ls 75

us 90

le 0

ue 100

bv 2600

# 6. Mobiles

Mobiles are the creatures that populate your area. They are defined in the #MOBILES section of the final area file. You don't have to define any mobs in your area for it to be valid. Mobs are individually defined in single blocks. You can have as many mobile blocks as you wish. Mobiles may not share the same vnum: this will produce an error when you run Scribe over your source file. Remember to format your descriptive text fields to fit within an 80-column screen!

**Block header** $mob *or* $mobile

**Fields** *Field Description Type*

nm name (keywords) text

sh short description text

lo long description text

de description multi-line text block

vn vnum number

lv level number

al alignment number

sx sex keyword

bf body form keyword list

act act flags keyword list

aff affect flags keyword list

sp special function keyword

mp mob program multi-line text block

te teacher skill text

Description nm Name (keywords)

Line of text

The keywords that can be used to indicate the mob.

E.g. nm wraith hazy shadow

sh Short description

Line of text

The short name of the mob, used whenever an action is performed involving the mob. Don't capitalise any leading ‘a’, ‘the’, ‘an’, etc: the DD server does automatically where necessary.

E.g. sh an alligator

sh the Gatekeeper

sh Mycroft

lo Long description

Line of text

The description of the mob as it appears in room after the *LOOK* command is issued. Capitalise the initial letter and don't make the description too long if the mob will have many affect flag labels like *(White Aura)* and *(Flaming)*. Remember full stops etc.

E.g. lo A snake lurks in the grass.

de Description

Multiple-line text block

The description of the mob as it appears after the *LOOK <mobile>* command is issued. Descriptions can span multiple lines; the de text block is terminated by a line containing a single tilde ~. Text on the same line after the de keyword is ignored. It is best to justify your text hard up against the left margin. Leading space before the first character in the block is removed by the DD server.

E.g. de

The alligator is immense and ferocious,

thrashing its tail in the water and

baring its razor-like teeth.

~

vn Vnum

Number: 0 or higher

The virtual number of the mob: its unique identifier. Remember this is a relative value, and will most likely begin at 0 (see Section 5).

lv Level

Number: 0 or higher

The level of the mob.

al Alignment

Number: ­-1000 to 1000

Default: 0

How good or evil the mob is. -1000 is absolutely evil, 0 is true neutral, 1000 is absolutely good. Defaults to zero if not indicated.

sx Sex

Keyword

Default: neuter

The sex of the mob. Defaults to neuter (sexless) if not present. Indicated by a single keyword:

neuter

male

female

E.g. sx female

bf Body form

Keyword list

Default: none

Describes the morphology (physical structure) of the mob. The default value of none describes a humanoid of normal size that is capable of speech and is made of flesh and blood.

Body form controls carnage and corpse production upon the mobile’s death, and affects what combat manoeuvres the mob may use or have used against it.

Body form is set using a list of any of the following keywords:

none default value

no\_head has no head

no\_eyes has no eyes

no\_arms has no arms

no\_legs has no legs

no\_heart has no heart

no\_speech cannot speak the common language

no\_corpse does not produce a corpse: body disappears upon death and loot falls to the ground

huge enormous in size

inorganic not made of flesh and blood

E.g. bf no\_arms no\_speech

bf no\_heart inorganic no\_speech huge

act Act flags

Keyword list

Default: none

Act flags define how the mobile behaves within the MUD world. If not included, a default value of none is used, describing a non-aggressive, non-wimpy creature that wanders between rooms and has no special interactions with players. The following flags may be used:

none, zero no flags

sentinel stays in one place

scavenger picks up objects from ground

questmaster can give players random quests

aggro aggressive: attacks players within sensible range

stay\_area does not leave the area

wimpy flees from combat if hurt

no\_quest will not be selected as a target for a quest

practice can train players

no\_charm cannot be charmed

famous fame rewarded if killed

lose\_fame fame subtracted if killed

wizinvis undetectable by players

mount can be mounted

banker runs a bank for players

identify identifies objects for players

E.g. act aggro scavenger

act no\_charm no\_quest lose\_fame

act mount

aff Affect flags

Keyword list

Default: none

Affect flags define any special abilities or magical/supernatural effects that the mob is affected by. If not included, a default value of none is used. The following flags may be used:

none, zero no flags

blind cannot see

sneak movement not reported

hide cannot be seen if still

passdoor may move through closed doors

invis is invisible (normal invis)

infrared has infrared vision

det\_evil can detect evil mobs or players

det\_invis can detect invis

det\_magic can detect magic

det\_hidden can detect hidden mobs or players

det\_good can detect good mobs or players

det\_traps can detect traps

det\_sneak can detect sneaking mobs or players

hold is trapped, cannot move

sanctuary has *sanctuary* spell

globe has *globe* spell

protection has *protection* spell

faerie\_fire has *faerie fire* spell

flaming has *fireshield* spell

meditate is meditating

fly is flying

E.g. aff sneak hide invis infrared

aff globe sanctuary flaming fly

aff poison

sp Special function

Keyword

*Special functions* give extra behaviours to mobs either during combat or outside of combat. If you don't wish to give your mob a special function, do not include the sp field at all; if you include an sp field and leave it blank you will produce an error.

Only one special function may be granted per mob:

spec\_breath\_acid breathes acid [combat]

spec\_breath\_fire breathes fire [combat]

spec\_breath\_frost breathes frost [combat]

spec\_breath\_gas breathes gas [combat]

spec\_breath\_lightning breathes lightning [combat]

spec\_breath\_any breathes any of the above at random

spec\_buddha random breath weapons and cleric spells [combat]

spec\_guard\_white attacks killers, thieves or evil players

spec\_kungfu\_poison poison-palm technique [combat]

spec\_warrior warrior skills [combat]

spec\_vampire vampire skills [combat]

spec\_bloodsucker sucks blood [combat]

spec\_clan\_guard guards clan entrance

spec\_cast\_adept healer

spec\_cast\_hooker sexy healer ;)

spec\_cast\_druid casts druid spells [combat]

spec\_cast\_cleric casts cleric spells [combat]

spec\_cast\_ghost undead ghost; appears only during night

spec\_cast\_judge fires explosive bullets (a la Dredd) [combat]

spec\_cast\_mage casts mage spells [combat]

spec\_cast\_psionicist casts psionic spells [combat]

spec\_cast\_undead casts undead spells [combat]

spec\_cast\_orb powerful healer

spec\_cast\_archmage casts powerful mage spells [combat]

spec\_cast\_priestess casts powerful cleric spells [combat]

spec\_cast\_chill casts spell chill touch [combat]

spec\_executioner attacks thieves and killers

spec\_fido eats corpses

spec\_guard cityguard

spec\_janitor gathers rubbish from ground

spec\_poison poisonous bite [combat]

spec\_repairman repairs broken doors

spec\_thief steals coins

spec\_scavenger gets objects from ground

spec\_cleaner gathers rubbish from ground

E.g. sp spec\_cast\_adept

mp Mob program

Multi-line text block

*Mob programs* (*mob progs*, *mprogs*) are scripts that add special functionality to mobs. What mob progs are available and what syntax they use is not discussed here. A single mob prog is defined by each mp tag; you can have as many mp tags per mobile as you like. Mob progs are defined using the following format:

mp *<mob prog name and arguments>*

*<mob prog code>*

~

E.g. mp death\_prog 100

mpecho The water begins to thrash!

mpmload 2601

mpmload 2601

~

te Teacher skill

Line of text

Some mobs are able to train particular skills for players. The mob must have the practice act flag set or these fields will be ignored. You can define as many teacher fields as you like per mob. Use the following format:

te *<percentage> <skill name>*

*Percentage* must be a number that is 0 or higher. Do not quote skill names if they contain multiple words.

E.g. te 100 divine magiks

te 75 flamestrike

Examples

$mob

nm imp horrible

sh an imp

lo A horrible imp prances about the room.

de

The imp looks horrible, its hairless body a dirty brown

colour and its eyes a mucky yellow.

~

vn 0

lv 2

$mob

nm gezhp mighty warrior dwarf dwarven

sh Gezhp

lo Gezhp the mighty dwarven warrior stands afore!

de

What a fearsome yet attractive fellow this dwarven warrior

is... such a magnificent beard, etc.

~

vn 1

lv 150

act sentinel famous no\_charm practice

aff sanctuary globe flaming det\_evil

bf no\_heart

sx male

te 100 headbutt

te 100 charm

te 100 dwarven wrestling

al 750

mp rand\_prog 10

say Dwarvish? You're not wrong!

~

mp death\_prog 100

shout AIEE! I'm done for!

~

sp spec\_warrior

# 7. Objects

Objects are the items found in your area; they are either carried or equipped by mobiles or are placed in rooms or container objects; they can be carried and worn by players or be immovable fixtures or features of a room; they can also be trapped (or be traps themselves). They are defined in the #OBJECTS section of the final area file. You don't have to define any objects in your area for it to be valid. Objects are individually defined in single blocks. You can have as many object blocks as you wish. Objects may not share the same vnum: this will produce an error when you run Scribe over your source file.

**Block header** $obj *or* $object

**Fields** *Field Description Type*

nm name (keywords) text

sh short description text

lo long description text

vn vnum number

ty type keyword

v0 value0 variable

v1 value1 variable

v2 value2 variable

v3 value3 variable

wg weight number

ex extra flags keyword list

we wear flags keyword list

ed extra description multi-line text block

ap apply effect text

trt trap trigger keyword list

trd trap damage type keyword

trc trap charges number

Description nm Name (keywords)

*Line of text*

The keywords that can be used to indicate the object for manipulation.

E.g. nm potion red bubbling

sh Short description

*Line of text*

The description of the object as it appears when manipulated or in a character's inventory. Don't capitalise any leading ‘a’, ‘the’, ‘an’ etc.

E.g. sh a bubbling red potion

lo Long description

*Line of text*

The description of the object as it appears in the room after a *LOOK* command. Capitalise and terminate with a full stop, etc.

E.g. lo A bubbling red potion lies here.

vn Vnum

*Number: 0 or higher*

Relative vnum of the object. The first object would usually be zero.

ty Type

*Keyword*

The object's type; one of the following:

light light source

scroll *recite* for spells

wand *zap* for spells

paint *smear* for spells

staff *brandish* for spells

potion *quaff* for spells

weapon

armour

money coins

treasure valuables (not coins)

furniture

trash

container holds other items

drink\_container holds liquids

key

food

boat

corpse can be used as a container

fountain water fountain

climbing\_eq for scaling walls, cliffs

anvil used for refining armour

auction\_ticket allows participation in an auction

clan clan healing item

portal portal to other location

powder for poisoning weapons

lockpick for picking locks

instrument for singing songs (Bards)

E.g. ty potion

v0-v3 Values

*Numbers or text*

The four value fields v0, v1, v2 and v3 are used by some types of object. Some object types do not use any value fields, and you will not have to include them in the **$object** block. Other types expect certain value definitions and Scribe will report errors if they are absent or invalid. You must supply the relevant value fields for the following objects:

###### Lights

###### v2 Hours of light provided

###### Number

###### A value below zero indicates infinite hours of light.

###### E.g. ty light

###### v2 -1

###### Scrolls, potions, paints and pills

###### v0 Level of spell(s)

###### Number: 1 or higher

###### v1-v3 Name of spell(s)

###### Text

###### You should indicate between 1 and 3 spells.

###### E.g. ty potion

###### v0 10

###### v1 heal

###### v2 cure poison

###### Wands and staves

###### v0 Level of spell

###### Number: 1 or higher

###### v1 Maximum charges

###### Number: 0 or higher

###### v2 Current charges

###### Number: 0 or higher

###### v3 Name of spell

###### Text

###### E.g. ty wand

###### v0 30

###### v1 5

###### v2 3

###### v3 combat mind

###### Weapons

###### v3 Attack type

###### Keyword

###### One of the following:

###### hit slice stab

###### slash whip claw

###### blast pound crush

###### grep bite pierce

###### suction chop rake

###### swipe sting mash

###### E.g. ty weapon

###### v3 pound

###### Containers

###### v0 Capacity

###### Number: 0 or higher

###### How much weight (in pounds) the container can hold.

###### v1 Lid flags

###### Keyword list

###### Whether the container has a lid, and whether the lid is closed or locked:

###### none open, no lid

###### closable has a lid

###### pickproof lock can't be picked

###### closed lid is closed

###### locked lid is locked

###### Weirdness can result if the flags are incorrectly set (e.g. locked but with no lid).

###### v2 Key

###### Number: -1 or higher

###### The relative vnum of any key. Use -1 to indicate no key exists.

###### E.g. ty container

###### v0 50

###### v1 closable closed locked

###### v2 4

###### Drink containers

###### v0 Capacity

###### Number: 0 or higher

###### The maximum number of draughts the container can hold.

###### v1 Current capacity

###### Number: 0 or higher

###### Current number of draughts in the container.

###### v2 Liquid type

###### Keyword

###### One of the following:

###### water beer wine

###### ale dark\_ale whiskey

###### lemonade firebreather local

###### slime\_mold milk tea

###### coffee blood salt\_water

###### cola

###### v3 Poison

###### Number

###### zero not poisoned

###### non-zero poisoned

###### E.g. ty drink\_container

###### v0 5

###### v1 4

###### v2 blood

###### v3 0

###### Food

###### v0 Hours of nourishment

###### Number: 0 or higher

###### v3 Poison

###### Number

###### zero not poisoned

###### non-zero poisoned

###### E.g. ty food

###### v0 0

###### v3 -1

###### Money

###### v0 Copper coins

###### Number: 0 or higher

###### Default: 0

###### v1 Silver coins

###### Number: 0 or higher

###### Default: 0

###### v2 Gold coins

###### Number: 0 or higher

###### Default: 0

###### v3 Platinum coins

###### Number: 0 or higher

###### Default: 0

###### E.g. ty money

###### v2 50

###### v1 200

###### Portal

###### v0 Destination room

###### Number: 0 or higher

###### The relative vnum of the room where players who enter the portal will be transported.

###### E.g. ty portal

###### v0 22

All types not listed above do not require any values to be defined; all relevant variables are calculated by the DD server based on the item’s level.

wg Weight

*Number: 0 or higher*

The weight of the object in pounds.

ex Extra flags

*Keyword list*

Any special properties of the object are indicated using the following keywords:

none, zero no extra flags

glow glows (visual effect)

hum hums (visual effect)

evil is evil

invis is invisible

magic is magical

no\_drop cannot be dropped

no\_remove cannot be removed

bless has been blessed (weapon)

anti\_good cannot be worn by good players

anti\_neutral cannot be worn by neutral players

anti\_evil cannot be worn by evil players

poison is poisoned (extra damage if an item is a weapon)

anti\_mage cannot be used by mages, warlocks or necromancers

anti\_cleric cannot be used by clerics, templars or druids

anti\_thief cannot be used by thieves, bounty hunters or ninjas

anti\_warrior cannot be used by warriors, thugs or knights

anti\_psionic cannot be used by psionicists, witches or satanists

anti\_ranger cannot be used by rangers, barbarians or bards

anti\_brawler cannot be used by brawlers, monks or martial artists

anti\_shifter cannot be used by shapeshifters, vampires or werewolves

vorpal can be used to *decapitate* with

sharp has been sharpened (weapon)

bladethirst is thirsty (weapon)

forged has been forged (armour)

body\_part is a body part; may not be disarmed in combat

lance can be used to *joust* with

bow can be used to *shoot* with

E.g. ex glow magic evil

ex poison anti\_cleric anti\_good

we Wear flags

*Keyword list*

Wear location information for the object. Any number of the below locations can be given, but to avoid weirdness, choose one of the following combinations:

none cannot be picked up or worn;

take can be picked up, cannot be worn;

take *<pos>* can be taken and worn in position *pos* (only one position is given).

The following keywords can be used:

none, zero cannot be taken or worn

take can be taken

finger

neck

body usually heavy, e.g. armour

about\_body usually light, e.g. shirt

head

legs

feet

hands

arms

shield

waist

wrist

wield weapons except for lances and bows

ranged for lances and bows (required for joust and shoot to work)

hold held in hand

float orbits about head

pouch belt-pouch

E.g. sh a huge cast iron stove

we none

sh a sharp dagger

we take weapon

sh a longbow

we take ranged

sh a small potted plant

we take hold

ed Extra description

*Multi-line text block*

Extra descriptions are descriptive text seen by characters who examine the item. Extra descriptions consist of a list of keywords and a text block. The text block is printed when a character enters *LOOK <keyword>* when the item is visible to her. They have the following format:

ed *<keyword list>*

*<lines of text>*

~

The terminating tilde is required. You can give multiple extra descriptions to items.

E.g. ed pot plant cactus

A small ceramic pot contains a squat,

prickly cactus. A large pink flower

blooms from its spiny crown.

~

ed pink flower

The flower sprouting from the top of

the cactus is pretty and fragrant.

~

ap Apply effect

*Line of text*

Applied effects are bonuses, penalties or special enhancements given to characters when they wear an item. They have the following format:

ap *<apply type> <modifier>*

*Apply type* is one keyword from the list below; *modifier* is any number.

str strength

int intelligence

wis wisdom

dex dexterity

con constitution

hp hit point maximum

mana mana point maximum

move movement point maximum

ac armour class

hitroll to-hit modifier

damroll damage bonus

save\_spell save versus spell

save\_breath save versus breath

fly flight \*

sneak move undetected \*

pass\_door pass through doors \*

invis invisibility \*

det\_invis detect invis mobs or players \*

det\_hidden detect hidden mobs or players \*

flaming *fireshield* spell \*

protect *protection* spell \*

globe *globe* spell \*

sanc *sanctuary* spell \*

(\*) The value of the modifier for these applies is not important; 1 is usually used.

You may give an item as many applies as you wish.

E.g. ap hitroll -4

ap int 5

ap fly 1

trt Trap trigger

*Keyword list*

*[Required if trap extra flag set]*

The event which will trigger a trap installed in the object. This field will be ignored if the trap extra flag has not been given to the object. Use one of the following triggers:

room trap will affect everyone in room

move movement in any direction triggers trap

north movement north triggers trap

south movement south triggers trap

east movement east triggers trap

west movement west triggers trap

up movement up triggers trap

down movement down triggers trap

object Trap triggered on *GET <object>* or *PUT <object>*

open Trap triggered on *OPEN <object>*

E.g. trt room open

trd Trap damage type

*Single keyword*

*[Required if trap extra flag set]*

The type of effect the trap produces after it is triggered.

sleep victim sleeps

teleport teleports victim away

poison poisons victim

fire

cold

acid

energy damage inflicted for these types

blunt

pierce

slash

trc Trap charges

*Number: 0 or higher*

*[Required if trap extra flag set]*

Number of charges left in the trap.

Examples:

$obj

vn 0

ty armour

nm wooden shield

sh a wooden shield

lo A wooden shield has been left here.

we take shield

wg 20

$obj

vn 1

nm jewellery box golden

sh a golden jewellery box

lo A golden jewellery box rests on the floor.

we take hold

wg 10

ty container

ex trap

v0 8

v1 closable closed locked

v2 3

trt open

trc 1

trd poison

$obj

vn 2

ty staff

nm staff serpent golden snake

sh the Staff of the Serpent

lo You see a long golden staff fashioned as a snake.

we take hold

wg 35

ex glow magic anti\_good anti\_evil

v0 75

v1 5

v2 5

v3 gas breath

ed staff serpent golden snake

The staff is made of solid gold with small emeralds inset

along the shaft. The top end has been fashioned into a

beautiful cobra's head, with a gaping jaw and long

protruding fangs. Large emeralds serve as the staff's

eyes, and the instrument glows softly.

~

ap int 5

ap wis 5

ap hp –100

# 8. Rooms

Rooms are distinct locations within your area, and need not be rooms in a literal sense (inside space with walls, ceiling, floor and doors). They are defined in the #ROOMS section of the final area file. You don't have to define any rooms in your area for it to be valid, although you'll probably want rooms if you want people to adventure in your area! Rooms are individually defined in single blocks. You can have as many room blocks as you wish. Rooms may not share the same vnum: this will produce an error when you run Scribe over your source file.

**Block header** $room

**Fields** *Field Description Type*

vn vnum number

nm name (title) text

de description multi-line text block

st sector type keyword

rf room flags keyword list

ed extra description multi-line text block

rnd random exits keyword

Exit fields In the following list, the symbol ? is replaced by n, s, e, w, u or d for north, south, east, west, upwards or downwards exits.

*Field Description Type*

? exit number

?nm exit name (keywords) text

?de exit description multi-line text block

?lo exit locks keyword list

?ke exit keys number

?ds exit door state keyword

Description nm Name (title)

Line of text

The title of the room as it appears after the *LOOK* command is issued. Don't use terminating punctuation. Capitalise as desired (although make the initial letter a capital).

E.g. nm The Dark Gate

nm A narrow, overgrown forest path

de Description

Multi-line text block

The descriptive blurb that is shown after the *LOOK* command is given.

st Sector type

Single keyword

Default: inside

Describes the type of terrain that room has; used to calculate movement penalties. Use one keyword from the following list:

inside (You probably want to set the indoors room flag too)

city

field

forest

hills

mountain

water\_swim Don't need boat/flight to enter

water\_no\_swim Need boat/flight to enter

underwater (Currently can't be entered)

air Need flight to enter

desert

rf Room flags

Keyword list

Default: none

Defines any special properties of the room. Use a list of any of the following keywords:

none, zero no room flags

dark need light source

no\_mob mobs may not enter

indoors sheltered from weather, sunlight

private space for only two creatures

safe can't pkill

solitary space for only one creature

pet\_shop pet store

no\_recall can't recall

silence can't cast spells

arena anyone can pkill, without penalty

healing accelerated healing within room

freezing players take cold damage per tick

burning players take heat damage per tick

E.g. rf safe healing no\_mob

ed Extra description

Multi-line text block

Extra descriptions are descriptive text seen by characters who examine the room. Extra descriptions consist of a list of keywords and a text block. The text block is printed when a character enters *LOOK <keyword>* within the room. They have the following format:

ed *<keyword list>*

*<lines of text>*

~

E.g. ed writing wall

You read the writing on the wall:

"Dwarves do it standing up."

~

rnd Random exits

Keyword

Randomise the exits in the room, so that the room becomes a maze. Use one of the following keywords:

2d two-dimensional maze (north, south, east and west exist scrambled)

3d three-dimensional maze (north, south, east, west, up and down exits)

E.g. rnd 2d

This field will also accept numbers from 0 to 6 if you require (although it is recommended you use the 2d or 3d keywords). You should avoid using random exits if you are going to include door resets (see below); these resets may produce unexpected results.

**Exits** There are six possible exits from each room; each room may contain between 0 and 6 exits.

###### Exit code Exit direction

n north

s south

e east

w west

u upwards

d downwards

To create an exit simply add the following to your **$room** block:

*<exit code> <destination room relative vnum>*

E.g. n 0

u 23

If you don't want a room to have an exit in a particular direction, just leave out the relevant field. If you do not add any further information for your exit it will be valid; however, it won't have a door or any form of description.

The following fields can be used to further define exits. They should be used in the following manner:

*<exit code><field> <data>*

There is no space between *exit code* and *field*.

?nm Exit name (keywords)

Line of text

A list of keywords that describe the exit. Usually used to indicate a door. The first name on the list will be used for generating messages by the MUD server.

E.g. wnm door iron reinforced

unm reinforced iron door

dnm path overgrown

Regarding the first two examples above, the first is preferred to the second, as any messages from the MUD look more natural:

Cf. "Crash! You bash open the door!"

"Crash! You bash open the reinforced!"

?de Exit description

Multi-line text block

The description of the exit given after the *LOOK <direction>* or *LOOK <exit keyword>* is given.

E.g. wde

The small trail wanders west into the

heavy forest.

~

?lo Exit locks

Keyword

This value is used to indicate whether an exit is a door, and whether the door can be forced open if locked or passed through if closed. It can also be used to indicate whether an exit is a wall or if it is secret (doesn't show up on *SCAN* etc). Due to the way the *locks* value is read by the server, the keywords used by this field are rather awkward. You may enter one of the following keywords; you can also use the relevant number between 0 and 12.

0 none no door

1 door door

2 pick pick-proof door

3 bash bash-proof door

4 pick\_bash pick-, bash-proof door

5 pass pass-proof door

6 pick\_pass pick-, pass-proof door

7 bash\_pass bash-, pass-proof door

8 pick\_bash\_pass pick-, bash-, pass-proof door

9 wall wall, able to be scaled using *climb*

10 door\_secret hidden door

11 door\_secret\_pbp pick-, bash-, pass-proof hidden door

12 secret hidden exit (not a door)

Door resets will be ignored if you fail to indicate that the exit is a door with one of these flags (all are doors except for none, wall and secret).

?ke Exit key

Number: -1 or higher

Default: -1

The relative vnum of the object that can be used to unlock the door. Use –1 if you wish there to be no key in existence. Any value above –1 indicates a relative vnum (including 0).

E.g. eke -1

nke 4

?ds Door state

Keyword

This field produces a door reset, i.e. will update the position of the door every time your area is reset. Indicate one of the following actions:

open open and unlocked

close closed and unlocked

lock closed and locked

The reset will only be used if the relevant exit has been defined and that exit is a door.

E.g. e 2

elo door

eds close

Unless otherwise desired (e.g. one-way doors), use the same door reset in both rooms sharing a door for consistency.

Examples

$room

vn 1

nm The Void

de

You float in the inky darkness of the Void.

~

$room

vn 10

nm An small, empty room

de

This tiny room is entirely empty except for a few pieces

of litter against the walls. You may return to the main

corridor through the southern archway.

~

st inside

rf indoors private dark

s 9

$room

vn 11

nm At the base of the Outpost Tower

de

You cross the courtyard to the base of the northern wall.

The lean stone tower rises above you; a sturdy wooden door

in the centre of the wall leads into its heart.

You may head north through the door into the tower,

or head west, south or east back across the courtyard.

~

ed tower outpost

The tower rises high into the air; it is at least half

a dozen stories high. Flashes of light flicker from its

peak every now and then.

~

st city

n 12

nlo pick\_bash

nke 23

nds lock

nnm door sturdy wooden

nde

The door leading into the tower is reinforced with heavy

iron bands. It looks very solid, and is covered in sharp

studs. You see a large keyhole in its centre.

~

w 9

e 8

s 6

# 9. Mobile resets

A *reset* refers to an action performed by the MUD to manipulate and update the world's areas. Mobile resets determine how mobiles are loaded into your area: what rooms they appear in, how many may appear and what objects they are wearing or carrying. The $**addmob** block is used to describe a single mobile reset. You may have as many $**addmob** blocks as you like in your area. You do not need to have any mobile resets for your area to be valid. You may use any particular rooms and mobs as many times as you wish.

**Block header** $addmob*or*$addmobile

**Fields** *Field Description Type*

mb mobile vnum number

rm room vnum number

num maximum mobile number number

inv item in inventory number

\* equip item number

\* See below for list of *equip* fields.

Description mb Mobile vnum

*Number: 0 or higher*

The relative vnum of the mobile you wish to load. A mob using this vnum must be defined otherwise Scribe will indicate an error.

rm Room vnum

*Number: 0 or higher*

The relative number of the room you wish to load the mobile into. A room using this vnum must also be defined otherwise an error will result.

You can add as many rm fields as you like to a single $**addmob** block, and identical mobiles carrying and wearing all equipment specified will be loaded into each room (up until the limit specified by the num field).

num Maximum mobile number

*Number: 1 or higher*

*Default: 1*

The maximum number of mobiles with this vnum that can be loaded into the world (all areas) at any one time.

inv Item in inventory

*Number: 0 or higher*

The relative vnum of any object you wish the mob to carry. An object using this must be defined otherwise an error will result. You do not have to include any inv fields in your reset; you may give a reset as many inv fields as you like.

\* Equip an item

*Number: 0 or higher*

Equip an object on the mob. The following format is used:

*<position> <relative vnum>*

*Position* is any of the following:

light light source

finger1

finger2

neck1

neck2

body

on body

head

legs

feet

hands

arms

shield

about about body

waist

wrist1

wrist2

wield primary weapon

dual secondary weapon

ranged ranged weapon (lance, bow)

hold held in hand

float orbiting head

pouch belt pouch

You may equip as many items as you wish; if you indicate the same position more than once, the last instance defined will be used.

Examples

*Load a single naked mob to a room:*

$addmob

mb 1

rm 3

*Load a naked mob to rooms 30, 32 and 33 (relative vnums); keep adding a mobile to these rooms every time the area resets until there are 6 mobs in the MUD:*

$addmob

mb 4

rm 30

rm 32

rm 33

num 6

*Load and equip a mob:*

$addmob

rm 3

mb 3

wield 7

inv 6

shield 9

body 8

inv 5

inv 5

# 10. Object resets

Object resets determine how objects other than those given to mobiles are loaded into your area: objects that appear on the ground in rooms, and objects that are placed inside other objects. The **$addobj** block is used to describe a single object reset. You may have as many **$addobj** blocks as you like in your area. You do not need to have any object resets for your area to be valid. You may use any particular rooms and objects as many times as you wish.

**Block header** $addobj*or*$addobject

**Fields** *Field Description Type*

ob object vnum number

rm room vnum number

lv level of object number

con container object vnum number

Description ob object vnum

*Number: 0 or higher*

The relative vnum of the object you wish to load. An object using this vnum must be defined otherwise *Scribe* will indicate an error.

rm Room vnum

*Number: 0 or higher*

The relative number of a room you wish to load the object into. A room using this vnum must also be defined otherwise an error will result.

lv Level of object

*Number: 0 or higher*

The level you wish the object to be. **Only applies if the object is reset to the ground and NOT to a mob or container**.

con Container vnum

*Number: 0 or higher*

The relative number of a container object you wish to place the item in. A container object with this vnum must be defined for the reset to be valid.

Only one of each kind of object may occupy any single container. The object is loaded to the most recently loaded container with the specified vnum. For best results, only load one of each container object into your area.

You need to have at least one rm or one con fields for the **$addobj** block to be valid.

**Examples**

$addobj

ob 4

rm 12

$addobj

ob 2

rm 6

rm 7

con 12

# 11. Helps

You can define help file entries in your area using the **$help** block. You don't have to include any **$help** blocks in your area for it to be valid.

**Block header** $help

**Fields**  *Field Description Type*

he Help text Multi-line text block

lv Level restriction number

Description he Help text

*Multi-line text block*

Define the help entry keywords and body text using the he field:

he *<keywords or phrases>*

*<help text>*

~

The *keywords or phrases* string is automatically capitalised by Scribe.

E.g. he fly levitate

This is the text displayed whenever the

commands HELP FLY or HELP LEVITATE

are entered.

~

he 'aura of fear'

This is the text displayed whenever the

command HELP 'AURA OF FEAR' is issued.

~

lv Level restriction

*Number: -1 or higher*

The minimum level a character has to be in order to access the help entry. 0 is used for general help entries. Use –1 if you want the help entry keyword header to be hidden.

# 12. Shops

Mobiles can be made to run shops: the inventory of the shop is the inventory of the mobile. So long as the mobile is alive, players can attempt to buy and sell items from the shop.

**Block header** $shop

**Fields**  *Field Description Type*

vn Vnum of shopkeeper number

t1 Traded item type 1 keyword

t2 Traded item type 2 keyword

t3 Traded item type 3 keyword

t4 Traded item type 4 keyword

t5 Traded item type 5 keyword

ps Profit-sell number

pb Profit-buy number

oh Opening hour number

ch Closing hour number

Description vn Vnum of shopkeeper

*Number: 0 or higher*

The relative vnum of the mobile that will run the shop.

t1-t5 Traded item types

*Keyword*

The item types that the shopkeeper will be prepared to buy from players. You may indicate up to five types; you don’t have to specify any types for the shop to be valid. Use the same keywords used to define item types.

E.g. t1 weapon

t2 armour

ps Profit-sell

*Number: 0 or higher*

The percentage markdown on items sold to the shopkeeper. 100 is the intrinsic value of the item, 75 is a 25% markdown, etc. This value should be at most 100%.

pb Profit-buy

*Number: 0 or higher*

The percentage mark-up on items bought from the shopkeeper. 100 is the intrinsic value of the item, 150 is a 50% mark-up, etc. This value should be at least 100%.

oh ch Opening and closing hours

*Number: 0* - *23*

The times when the store opens and closes (0 = midnight).

**Example**

$shop

vn 0

t1 wand

t2 staff

t3 potion

t4 scroll

oh 7

ch 18

ps 85

pb 120