EmberMUD Area File Help Document

Prepared by Dorzak (dorzak@earthlink.net) for
Tempestuous Realms and the EmberMUD project

This document relies heavily on the excellent work by Ozymandias of MadROM at documenting the RoM 2.3 format, ozy.doc. Please see the original credits for that document[i].

Introduction

This document is being prepared to attempt to document in one place the changes to the area file format used by Ember. Ember is derived from RoM 2.3 but has several area file extensions incorporated into it. Among these are Breath Weapon strength modifiers, Random Objects loading to mobs, factions and our method of dealing with room, object, and mob progs.

An area is a place in the MUD, the world if you will. Each is written in a separate text file, which is loaded during the MUD startup. Areas can be edited offline or online. When naming the file the common file format is 8.3 combination of name and extension. The extension is .are. Spaces in file names should be avoided.

Each area file is broken up into several different sections. Starting with #AREA. Each section is in its own format. The sections used in Ember are: #AREADATA, #MOBILES, #OBJECTS, #ROOMS, #RESETS, #PROGS, #SHOPS, and #$ to end the file. Areas in a Rom 2.3 contained #AREA, #HELPS, and #SPECIALS as well. The original #AREA section has replaced with a longer #AREADATA which contains information used by OLC. All help files have been moved to a separate help file in EmberMUD, help.are. This is the only area which has a #HELPS section, and contains ONLY that section. Special procedures were used by Diku, Merc, and RoM, and most derivatives to provide special functions to specific mobs. EmberMUD relies almost exclusively on mobprogs to perform these functions.

Data Types

All of the data in an area file (even the section headers) consists of a series of values. Each value has a specific type. The server parses[ii] the file by reading in data values one at a time according to the types it expects. If it does not find the data it expects the MUD will not be able to work with the data, and will either shutdown, crash, or go into a loop that hogs up resources.

Blank characters (spaces, tabs, new lines, carriage returns) at the beginning of a data value are always ignored (this includes strings). Thus you can format the area files in whatever way suits your taste and your needs. The area sections are presented in the order that OLC will write them. This is done in order to keep things consistent. While it

may be possible to NOT do them in this order it is highly recommended. In any case #AREA has to be first.
Types of data you should know:

- A 'letter' is a single non-blank character.
- A 'word' is a sequence of non-blank characters terminated by a blank.
- A 'string' is a sequence of non-tilde characters terminated by a tilde. A 'tilde', by the way, is this character on your keyboard: '~ '. Thus strings may contain blanks, and may be multiple lines long. There is no limit on the length of an individual strings; however, all strings go into a common memory pool whose size is fixed when the server is compiled.
- A 'number' is a decimal number with an optional leading '-' or '+', as in the hit points section of a mobile, for example: 5d6+250. The 'd' in this example stands for (5) 6-sided dice PLUS 250, ranging for a hit point range of 255 to 280... 250 plus 5 to 30.

IMPORTANT: Mobiles, objects, and rooms are ALL IDENTIFIED by vnum (virtual number). The range of vnums is 1 to 32767, and each object, mobile, etc, has its own UNIQUE vnum. No two rooms, for example, can possess the same vnum. However, you can have an Object with vnum 3100 and a room with vnum 3100, for example. Just no two objects, two mobiles, or two rooms can have the same one as the other. Vnums no not have to be in increasing order.

Typically an area uses the same range of vnums for mobile vnums, object vnums, and room vnums, starting with a multiple of 100. This facilitates adding the area into an existing set of areas. So, you may pick something like 16000 to 16500 for your range of Vnums for all objects, mobiles and rooms, just as an example. Check with the implementer of the MUD your area is intended for BEFORE choosing a vnum range, as many vnums may already be in use in other areas.

I. The #AREADATA section.

This is the easiest section to do. The format is as follows:

```
#AREADATA
Name        <area name: string>~
Builders    <builders name(s): string>~
VNUMs       <lowvnum: number> <highvnum: number>
Security    <min security required: number>
End
```

This format does the following:
Identifies the section with #AREADATA, then it stores several fields of information. The 'area-name' can be any string. The 'areas' command provides a list of areas, so it's worthwhile to follow the standard Merc format for this string. This format would look as follows, to take an example from an area already designed:

```
Name    {30 60} Dorzak Elven Cavern~
```

The first two numbers are the recommended level range of the area. The first name is your own name or alias as it shows with the 'areas' command, and the last phrase is the name of the area itself.

II. The #HELPS section.

This section is never used in standard EmberMUD areas, if a #HELPS exists in an area there will be an error message in the logs, and the MUD will shutdown before coming completely up. However, the format for this section is retained in this document for reference in case you may end up having to edit the help.are file. The format is unchanged from how it appeared in Merc; its format as it appeared in the Merc 2.1 code is as follows:

```
#HELPS

<level:number>  <keywords:string>~
<help-text:string>
~
0 $
```

- The 'level' number is the minimum character level needed to read this section. This allows for immortal-only help text.
- The 'key-words' are a set of keywords for this help text.
- The 'help-text' is the help text itself.

Normally when a player uses 'help', both the keywords and the help-text are shown. If the 'level' is negative, however, the keywords are suppressed. This allows the help file mechanism to be used for certain other commands, such as the initial 'greetings' text.

If a 'help-text' begins with a leading '.', the '.' is stripped off. This provides for an escape mechanism from the usual leading-blank stripping of strings, so that picturesque greeting screens may be used.

Remember, ALL strings must be followed by a tilde, the '~' symbol, to end that particular string. FAILURE to place the tilde will result in an error in your help file.

III. The #MOBILES section.

Let's start with a common mob from midgaard.are. This is the most commonly seen cityguard, and we will analyze him step by step.

```
#MOBILES
#3060
cityguard guard~
the cityguard~
A cityguard stands here.
~
A big, strong, helpful, trustworthy guard.
~
Human~
```

```
ACGT 0 1000 S
15 3
100 0 0 0 15d15+160 1d1+149 2d4+4 17
0 0 0 8
CDEKLTU 0 C 0
8 8 3 150
AHMV ABCDEFGHIJK M 0
#3061<----- THIS BEGINS THE NEXT MOBILE, AND SO ON....
```

Explanation of each line:

`#MOBILES` ..... Beginning of the Mobiles section

`#3060`.....Vnum of the Mobile

NOTE: The 'vnum' is the virtual number of the mobile.

`cityguard guard~`.....Keywords for the Mobile, followed by ~

NOTE: The 'keywords' are the words that can be used in commands to identify the mobile.

`the cityguard~`..... Short Description of Mobile

NOTE: The 'short-description' is the description used to identify the mobile.

`A cityguard stands here`..... Long Description of mobile

NOTE: The 'long-description' is the description used when a character walks in the room and the mobile is visible.

`~`.....Tilde closing the long description

`A big, strong, helpful, trustworthy guard` ..... The extended description

`~`..... Tilde closing the extended description

NOTE: The 'extended description' is the longest description. It is used when a character explicitly looks at the mobile.

`Human~`.....Race of the Mob

NOTE: This is case sensitive and should appear exactly as in the tables below, or listed in const.c for the EmberMUD you are building for.

`ACGT 0 1000 S`..... <ACT> <AFF> <ALIGN> S

NOTE: Act is the act_flags that apply to the mob, aff is the affected by flags that apply to the mobile, align is the alignment, and S ends this line to help in generating useful error messages when loading areas. The 'Act-flags' define how the mobile acts, and the 'affected_by flags' define more attributes of the mobile. Affected_by flags generally define what the mobile is 'affected' by. Therefore the affected_by flag of AFF_BLIND for example means that this particular mobile will be blind, and NOT that the mobile casts a blindness spell. The 'alignment' of the mobile ranges from -1000 to 1000, the higher being good, the lower being evil, with '0' being true neutral. Keep in mind that certain spells ('protection' and 'dispel evil') give characters fighting evil

monsters an advantage, and that experience earned is influenced by alignment.

`15 3` .....Level of the Mobile and Hitroll bonus

NOTE: The 'level' is typically a number from 1 to 100, although there is no upper limit. The default in EmberMUD is 60, and the experience tables cover 1-100 in EmberMUD.

`100 0 0 0 15d15+160 1d1+149 2d4+4 17` <Breath Weapon Str> <Random Chance> <Random Number> <Random Tytpes> <hit points> <mana> <damage> <DAMAGE TYPE, for when not using a weapon>

NOTE: Breath Weapon strength allows you to affect the amount of damage done by breath weapons on some mobiles. The Random numbers give the chance of the mob loading with a random object, the number of objects, and the types of objects. A chart of these appears later in this document.

`0 0 0 8`.....Armor Class (AC) values for <piercing> <bashing><slashing> <magic>.

NOTE: A common misconception is that these values you install in this line will be the mob's TOTAL ACs for the 4 AC values. This is not so. Simply put, an AC that reads in your file as: -15 -15 -15 -15 0 will read on the mud as: -150 -150 -150 -150 0

`CDEKLTU 0 C 0`.....this line is:

<Off_bits> <immunities> <resistances> <vulnerabilities>. A list of the Off_bits, and the immunities, resistances and vulnerabilities is at the end of this section, as well as listings of the Affected_by and ACT bits of above. The 'Off_bits' are composed of the 'skills' that the mobile has, as well as whether or not this mobile will assist other mobiles of the same vnum, alignment, etc. In the case here, with this mobile having CDEKLTU gives this mobile the following skills: bash, berserk, disarm, parry, rescue, assist-guard,, and assist_vnum. A nice combination for a warrior-type mobile, over all. This guard has no particular immunities, resists magic, and has no particular vulnerabilities.

`8 8 3 150`..... This line is <position number> <position number> <sex> <gold>

The first number is the position they start in when loaded, and the second is their default position. In this case 8 8 is used for standing/standing. Sex is 0 = neuter, 1 = male, 2 = female, and 3 = random. The last value is gold. On Tempestuous Realms the goal is to have no more than 10 per level of the mob.

`AHMV ABCDEFGHIJK M 0`..... This line is <form> <parts> <size> <0>

Note: In this case the mob is edible, sentient, biped, and a mammal for form. It has a head, arms, legs, heart, brains, guts, hands, feet, fingers, ear, and eye. The size is M for medium, the most common. The sizes are: : T=tiny, S=small, M=medium, L=large, H=huge, and G=giant.

#3061..... THIS BEGINS THE NEXT MOBILE, AND SO ON....

IMPORTANT: When writing your mobiles, you may be having trouble deciding how many hit points and mana to give your monsters and NPCS. For mana, this is easy: the mobiles don't use the mana really for spell casting..so you must put an amount, but it is not vital that you spend hours worrying about it.. Tempestuous Realms maintains a chart for builders to use for typical mob values.

For Very Tough mobiles - For inherently TOUGH mobiles, such as dragons, demons, Gods, devils, very large giants, titans, and other unusually powerful beings, hit points may be much higher than other mobiles of similar levels. This should be used extremely sparingly. This is entirely up to the goals of the area builder. A tough mob may add flavor to an area, an area full of them will not get many visitors.

Now here are the listings of all the necessary values for making your mobiles; the flags will be on the left, and the letter or number values you use are on the right column:

ACT FLAGS:

ACT_IS_NPC	(A)	*Auto set for mobs*
ACT_SENTINEL	(B)	*Stays in one room*
ACT_SCAVENGER	(C)	*Picks up objects*
ACT_AGGRESSIVE	(F)	*Attacks PC's*
ACT_STAY_AREA	(G)	*Won't leave area*
ACT_WIMPY	(H)	*Flees when drops below half hp*
ACT_PET	(I)	*Auto set for pets*
ACT_TRAIN	(J)	*Can train PC's*
ACT_PRACTICE	(K)	*Can practice PC's*
ACT_UNDEAD	(O)	*Saves as an undead*
ACT_CLERIC	(Q)	*Save and hitroll as cleric*
ACT_MAGE	(R)	*Save and hitroll as mage*
ACT_THIEF	(S)	*Save and hitroll as thief, will attempt to steal*
ACT_WARRIOR	(T)	*Save and hitroll as warrior*
ACT_NOALIGN	(U)	*Mob will not affect align*
ACT_NOPURGE	(V)	*Mob will not be purged with room*
ACT_MOBINVIS	(W)	*Mob is invis to mobs*
ACT_IS_HEALER	(aa)	*IS_HEALER – only special left in Ember*
ACT_GAIN	(bb)	*Gain can be used in room with mob*
ACT_UPDATE_ALWAYS	(cc)	*Mob will update even if area empty*
ACT_NO_KILL	(dd)	*Mob can not be attacked/killed*

RACES:

Here are the races that are in stock Ember for NPC's. The race of the mobile affects the default parts primarily. However, it also affects their default offensive, affect and act values. You can have Ogres whose race is human. Many of these exist for compatibility with legacy areas.

unique	Drow Elf	Goblin	school monster
Dwarf	Bat	Hobgoblin	shiriff
Elf	Bear	Kobold	snake
Giant	Cat	Lizard	song bird
Hobbit	Centipede	Modron	thain
Human	Dog	Orc	water fowl
Troll	Doll	Pig	
Wolf	Fido	Rabbit	
Wyvern	Fox	Sailor	

Mobile Size:

T = tiny	M = medium	H = huge
S = small	L = large	G = giant

FLAGS:

AFF_BLIND	(A)	AFF_SNEAK	(P)
AFF_INVISIBLE	(B)	AFF_HIDE	(Q)
AFF_DETECT_EVIL	(C)	AFF_SLEEP	(R)
AFF_DETECT_INVIS	(D)	AFF_CHARM	(S)
AFF_DETECT_MAGIC	(E)	AFF_FLYING	(T)
AFF_DETECT_HIDDEN	(F)	AFF_PASS_DOOR	(U)
AFF_HOLD	(G)	AFF_HASTE	(V)
AFF_SANCTUARY	(H)	AFF_CALM	(W)
AFF_FAERIE_FIRE	(I)	AFF_PLAGUE	(X)
AFF_INFRARED	(J)	AFF_WEAKEN	(Y)
AFF_CURSE	(K)	AFF_DARK_VISION	(Z)
AFF_FLAMING	(L)	AFF_BERSERK	(aa)
AFF_POISON	(M)	AFF_SWIM	(bb)
AFF_PROTECT	(N)	AFF_REGENERATION	(cc)
AFF_PARALYSIS	(O)	AFF_WEB	(dd)

REMEMBER: AFF_FLAG do not cause them to use them in combat, but are things that are currently affected the mobile.

OFF_BITS:

OFF_AREA_ATTACK	(A)	OFF_DODGE	(F)
OFF_BACKSTAB	(B)	OFF_FADE	(G)
OFF_BASH	(C)	OFF_FAST	(H)
OFF_BERSERK	(D)	OFF_KICK	(I)
OFF_DISARM	(E)	OFF_KICK_DIRT	(J)

OFF_PARRY	(K)	ASSIST_ALIGN	(Q)
OFF_RESCUE	(L)	ASSIST_RACE	(R)
OFF_TAIL	(M)	ASSIST_PLAYERS	(S)
OFF_TRIP	(N)	ASSIST_GUARD	(T)
OFF_CRUSH	(O)	ASSIST_VNUM	(U)
ASSIST_ALL	(P)		

Note: ASSIST bits are part of the OFF_ items as well.

IMMUNITIES:

IMM_SUMMON	(A)	IMM_ACID	(K)
IMM_CHARM	(B)	IMM_POISON	(L)
IMM_MAGIC	(C)	IMM_NEGATIVE	(M)
IMM_WEAPON	(D)	IMM_HOLY	(N)
IMM_BASH	(E)	IMM_ENERGY	(O)
IMM_PIERCE	(F)	IMM_MENTAL	(P)
IMM_SLASH	(G)	IMM_DISEASE	(Q)
IMM_FIRE	(H)	IMM_DROWNING	(R)
IMM_COLD	(I)	IMM_LIGHT	(S)
IMM_LIGHTNING	(J)		

RESISTANCES:

RES_CHARM	(B)	RES_ACID	(K)
RES_MAGIC	(C)	RES_POISON	(L)
RES_WEAPON	(D)	RES_NEGATIVE	(M)
RES_BASH	(E)	RES_HOLY	(N)
RES_PIERCE	(F)	RES_ENERGY	(O)
RES_SLASH	(G)	RES_MENTAL	(P)
RES_FIRE	(H)	RES_DISEASE	(Q)
RES_COLD	(I)	RES_DROWNING	(R)
RES_LIGHTNING	(J)	RES_LIGHT	(S)

VULNERABILITIES:

VULN_MAGIC	(C)	VULN_NEGATIVE	(M)
VULN_WEAPON	(D)	VULN_HOLY	(N)
VULN_BASH	(E)	VULN_ENERGY	(O)
VULN_PIERCE	(F)	VULN_MENTAL	(P)
VULN_SLASH	(G)	VULN_DISEASE	(Q)
VULN_FIRE	(H)	VULN_DROWNING	(R)
VULN_COLD	(I)	VULN_LIGHT	(S)
VULN_LIGHTNING	(J)	VULN_WOOD	(X)
VULN_ACID	(K)	VULN_SILVER	(Y)
VULN_POISON	(L)	VULN_IRON	(Z)

POSITIONS:

Note: Only use positions 4, 5, 6, and 8 in area files.

POS_DEAD	0	POS_STUNNED	3	POS_SITTING	6
POS_MORTAL	1	POS_SLEEPING	4	POS_FIGHTING	7
POS_INCAP	2	POS_RESTING	5	POS_STANDING	8

SEX of mobile:

NEUTRAL	0	MALE	1	FEMALE	2	RANDOM	3

DAMAGE TYPES:

0 - HIT
1 - SLICE
2 - STAB
3 - SLASH
4 - WHIP
5 - CLAW
6 - BLAST
7 - POUND
8 - CRUSH
9 - GREP
10 - BITE
11 - PIERCE
12 - SUCTION
13 - BEATING
14 - DIGESTION
15 - CHARGE
16 - SLAP
17 - PUNCH
18 - WRATH
19 - MAGIC
20 - DIVINE POWER
21 - CLEAVE
22 - SCRATCH
23 - PECK (pierce)
24 - PECK (bash)
25 - CHOP
26 - STING
27 - SMASH
28 - SHOCKING BITE
29 - FLAMING BITE
30 - FREEZING BITE
31 - ACIDIC BITE
32 - CHOMP
33 – WEAPON'S BITE (Vampiric)
34 – WEAPON'S FLAME (flaming)
35 – WEAPON'S FROST (freezing)

Note: The last three are used in the special damage messages of vampiric, flaming, and freezing weapon's. They should probably not be used in the area files.

IV. The #OBJECTS section:

The objects section uses similar layout to the #MOBILE section, except the exact format changes with EACH type of object. I am going to be using objects from the stock EmberMUD areas to illistrate this. However, to make the illistration more affective I am going to modify the objects a bit.

Example:
#3350
sword standard merc~
a standard issue sword~
You see a standard issue sword here.~
unknown~
5 GI AN
A C ABC AB C
8 5 500 P
A

18 5
A
19 5
E
sword~
You see a sword of great craftsmanship. Imprinted on the side is:
Merc Industries
~
#3351 <marks the start of the next object>

Explanation of each line:

#3350 ..... this is the vnum unique to this object
sword standard merc~ ..... these are the keywords/name for the object
sword standard merc~ .....short description of the object
You see a standard issue sword here.~ .....long description
unknown~ ..... the 'material' of the object.
NOTE: Materials are not fully implemented and so are always saved as unknown. Hopefully they will begin to be implemented more thoroughly in Ember 0.9.46
5 GI AN ..... <ITEM TYPE> <flags> <wear flags>.
NOTE: A listing of item types, flags and wear flags is provided at the end of the help for #OBJECTS section. The item type is just that; 5 = weapon, as reflected here. The flags add additional flags to the item, i.e., evil, magic, humming, glowing, etc. The wear flags tell where and if an item can be worn and where. A and N tell the code that this sword can be 'taken' and 'wielded' as a weapon; A = take, and N= wield. Without the 'A', this item could not even be picked up from the ground.
A C ABC AB C.....*IMPORTANT* This is the VALUES line.
NOTE: This is the section that has created the biggest headache for me in regards to figuring out what is what. It should be Weapon Class, Number of Dice, Size of Dice, Type of Damage, and then Special Type. However, we can read numeric values for all but the last value.
8 5 500 P ..... <level of obj.> <weight of object>
.....<value of object> <condition of object>.
A.....Start of and AddAffect Section
18 5..... Affect What/How Much
A..... Start of another Addaffect section
19 5..... Affect What/How Much
E ..... Marks the start of an optional extra description for the object
sword underworld~ .....Keyword for the Extra Description

You see a sword of great craftsmanship. Imprinted on the side is:
Merc Industries
~ ..... End of the extra description

The A and E sections are optional. If they do not exist the next object starts with #VNUM after the last line with the condition. A object can have unlimited A and E sections.

Materials

This section is reserved from materials once they are implemented. Currently Ember saves all materials as unknown.

OBJECT TYPES:

ITEM_LIGHT	1	ITEM_CLOTHING	11	ITEM_BOAT	22
ITEM_SCROLL	2	ITEM_FURNITURE	12	ITEM_CORPSE_NPC	23
ITEM_WAND	3	ITEM_TRASH	13	ITEM_CORPSE_PC	24
ITEM_STAFF	4	ITEM_CONTAINER	15	ITEM_FOUNTAIN	25
ITEM_WEAPON	5	ITEM_DRINK_CON	17	ITEM_PILL	26
ITEM_TREASURE	8	ITEM_KEY	18	ITEM_PROTECT	27
ITEM_ARMOR	9	ITEM_FOOD	19	ITEM_MAP	28
ITEM_POTION	10	ITEM_MONEY	20	ITEM_PORTAL	29

Object FLAGS:

ITEM_GLOW	(A)	ITEM_NOREMOVE	(M)
ITEM_HUM	(B)	ITEM_INVENTORY	(N)
ITEM_DARK	(C)	ITEM_NOPURGE	(O)
ITEM_LOCK	(D)	ITEM_ROT_DEATH	(P)
ITEM_EVIL	(E)	ITEM_VIS_DEATH	(Q)
ITEM_INVIS	(F)	ITEM_ANTI_CLERIC	(R)
ITEM_MAGIC	(G)	ITEM_ANTI_THIEF	(S)
ITEM_NODROP	(H)	ITEM_ANTI_WARRIOR	(T)
ITEM_BLESS	(I)	ITEM_ANTI_MAGE	(X)
ITEM_ANTI_GOOD	(J)	ITEM_ANTI_MALE	(Y)
ITEM_ANTI_EVIL	(K)	ITEM_ANTI_NEUTER	(Z)
ITEM_ANTI_NEUTRAL	(L)	ITEM_ANTI_FEMALE	(aa)

WEAR FLAGS:

ITEM_TAKE	(A)	ITEM_WEAR_ARMS	(I)
ITEM_WEAR_FINGER	(B)	ITEM_WEAR_SHIELD	(J)
ITEM_WEAR_NECK	(C)	ITEM_WEAR_ABOUT	(K)
ITEM_WEAR_BODY	(D)	ITEM_WEAR_WAIST	(L)
ITEM_WEAR_HEAD	(E)	ITEM_WEAR_WRIST	(M)
ITEM_WEAR_LEGS	(F)	ITEM_WIELD	(N)
ITEM_WEAR_FEET	(G)	ITEM_HOLD	(O)
ITEM_WEAR_HANDS	(H)	ITEM_TWO_HANDS	(P)

CONDITION OF OBJECT:

Note: This is not fully implemented

PERFECT	P	AVERAGE	A	DAMAGED	D
GOOD	G	WORN	W	RUINED	R

WEAPON CLASS:

WEAPON_EXOTIC	0	WEAPON_SPEAR	3	WEAPON_FLAIL	6
WEAPON_SWORD	1	WEAPON_MACE	4	WEAPON_WHIP	7
WEAPON_DAGGER	2	WEAPON_AXE	5	WEAPON_POLEARM	8

WEAPON TYPES:

WEAPON_FLAMING	(A)	WEAPON_SHARP	(D)
WEAPON_FROST	(B)	WEAPON_VORPAL	(E)
WEAPON_VAMPIRIC	(C)	WEAPON_TWO_HANDS	(F)

APPLY TYPES (for the 'A' section of #OBJECTS):

APPLY_NONE	0	APPLY_HIT	13
APPLY_STR	1	APPLY_MOVE	14
APPLY_DEX	2	APPLY_GOLD	15
APPLY_INT	3	APPLY_EXP	16
APPLY_WIS	4	APPLY_AC	17
APPLY_CON	5	APPLY_HITROLL	18
APPLY_SEX	6	APPLY_DAMROLL	19
APPLY_CLASS	7	APPLY_SAVING_PARA	20
APPLY_LEVEL	8	APPLY_SAVING_ROD	21
APPLY_AGE	9	APPLY_SAVING_PETRI	22
APPLY_HEIGHT	10	APPLY_SAVING_BREATH	23
APPLY_WEIGHT	11	APPLY_SAVING_SPELL	24
APPLY_MANA	12	APPLY_ALIGN	25

Container FLAGS (if OBJECT TYPE is container only):

CONT_CLOSEABLE	1	CONT_CLOSED	4
CONT_PICKPROOF	2	CONT_LOCKED	8

Furniture Flags:

STAND_AT	A	REST_ON	H
STAND_ON	B	REST_IN	I
STAND_IN	C	SLEEP_AT	J
SIT_AT	D	SLEEP_ON	K
SIT_ON	E	SLEEP_IN	L
SIT_IN	F	PUT_AT	M
REST_AT	G	PUT_ON	N

PUT_IN	O	PUT_INSIDE	P

Liquid values for DRINK CONTAINERS (value2 on the value line):

LIQ_WATER	0	LIQ_LOCAL_SPECIALTY	8
LIQ_BEER	1	LIQ_SLIME_MOLD_JUICE	9
LIQ_RED_WINE	2	LIQ_MILK	10
LIQ_ALE	3	LIQ_TEA	11
LIQ_DARK_ALE	4	LIQ_COFFEE	12
LIQ_WHISKEY	5	LIQ_BLOOD	13
LIQ_LEMONADE	6	LIQ_SALT_WATER	14
LIQ_FIREBREATHER	7	LIQ_COLA	15

Value lines for all the various OBJECT TYPES follow:

format -
<value0> <value1> <value2> <value3> <value4>

**note: if a value is shown to be 'unused', place a zero, '0', for
that value. DO NOT skip it or leave it blank.

ITEM_LIGHT:
<unused> <unused> <HOURS OF LGT. AVAILABLE <unused> <unused>
0=dead, 999=infinite
ITEM_SCROLL:
ITEM_POTION:
ITEM_PILL:
<spell lvl> <sn1> <sn2> <sn3> <unused>
NOTE: Potions, pills and scrolls can have up to 3 spells in them. If you use only 1 or 2, put a zero for each of the unused values. Try to use all 3 slots though, it makes the pill more interesting :)
NOTE: spell lvl is the level at which the spell will be cast, NOT the level of the reader of the scroll.
NOTE: 'sn' stands for SPELL NUMBER; this is the number unique to that particular spell in the game. If you are an immortal with the 'vnum' command, you can type 'vnum spell <spell name>' to get this number. If not, ask an immortal for the spell number of a particular spell you may be thinking of using for a potion, scroll, wand, staff, or pill in your #OBJECTS section.

ITEM_WAND:
ITEM_STAFF:
<spell lvl> <maximum charges> <current charges> <sn> <unused>

ITEM_WEAPON:
<weapon class> <# damage dice> <type of damage dice> <damage type>
<weapon types>

ITEM_TREASURE:
<unused> <unused> <unused> <unused> <unused>

ITEM_CLOTHING:
<unused> <unused> <unused> <unused> <unused>

ITEM_ARMOR:
<value for pierce> <value for bash> <value for slash> <value for magic> <unused>

ITEM_FURNITURE:
<max people> <max weight> <furniture flags> <Heal bonus> <Mana bonus>
NOTE: Max people is the maximum on it, max weight is the maximum weight of people plus their gear. Furniture flags include if you can sit on it, heal and mana bonus are expressed as a percentage value.

ITEM_TRASH:
<unused> <unused> <unused> <unused> <unused>

ITEM_CONTAINER:
<wgt. capacity> <container FLAGS> <key vnum> <unused> <unused>
NOTE: Use the Container FLAGS listed earlier for this value; the key vnum is ONLY USED if there is a key to unlock the container - otherwise put a zero for that value.

ITEM_DRINK_CONTAINER:
<liquid capacity> <current quantity> <liquid #> <0> <unused>
NOTE: The second to last value is if it is poisoned. 0 for no, 1 for yes. Use already existing drink containers in the game to get an idea as to a reasonable liquid capacity. For example, the water skin for sale at the Grocers Shop in Midgaard has a liquid capacity of 20. Additionally, the 'liquid #' value here tells whether it is beer, water, ale, etc., that is in the container. I have provided a listing above of the liquid numbers found in the 'liq_table' in the const.c section of the mud code. Liqid # MUST be one of 0 through 15.

ITEM_KEY:
<unused> <unused> <unused> <unused> <unused>

ITEM_FOOD:
<hours of food value> <unused> <unused> <unused> <unused>
NOTE: if the second to last value is 1, food is poison.

ITEM_MONEY:
<value in gold> <unused> <unused> <unused> <unused>

ITEM_BOAT:
<unused> <unused> <unused> <unused> <unused>

ITEM_FOUNTAIN:
<unused> <unused> <liquid type> <unused> <unused>

ITEM_MAP:
< 1 > <unused> <unused> <unused> <unused>

ITEM_PORTAL
<vnum goes to> <unused> <unused> <unused> <unused>

V. The #ROOMS section:

Here is an example of a room from the #ROOMS section of an area:

```
#3005
The Temple Square~
You are standing on the temple square.  Huge marble steps lead up to
the temple gate.  The entrance to the Clerics Guild is to the west, and
the old Grunting Boar Inn, is to the east.  Just south of here you see
the market square, the center of Midgaard.
~
0 262144 1
D0
You see the temple.
~
~
0 -1 3001
D1
You see the good old Grunting Boar Inn.
~
~
0 -1 3006
D2
You see the Market Square.
~
~
0 -1 3014
D3
You see the entrance to the Clerics Guild.
~
~
0 -1 3004
D4
You see the air.
~
~
0 -1 3057
S
```
#3006

Explanation of each line:

#3005 <--- <room vnum>

```
The Temple Square~   <--- <room name>
You are standing on the temple square.  Huge marble steps lead up to
the temple gate.  The entrance to the Clerics Guild is to the west, and
the old Grunting Boar Inn, is to the east.  Just south of here you see
the market square, the center of Midgaard.
```

~ <--- <above, room description, followed by the tilde here>

0 262144 1 <--- <area number> <room-flags> <sector-type>

```
D0        <--- <door-number>
You see the temple.
```

~ <--- <exit description> tilde goes on next line, as here.

~ ←-- <door (if there is one) keyword>

```
0 -1 3001 ←-- <locks-number> <key-number> <to_room-number>
D1 ←-- <next door number, and so on..>
You see the good old Grunting Boar Inn. ←- <next exit desc…>
~
~
0 -1 3006
E   <--- <extended description section>
statue~ <--- <extended desc. keywords>
Here stands a statue inserted for this document.  <--- <the extended
description itself>
~
S  <--- the 'S' marks the end of the room. It is NOT optional.
#3006  <--- <next room vnum, and so on...>
```

Notes:

- When doing Extended descriptions, EACH object/description must start with an E. eg if your room has a sign, chair and plaque, each extended description for these should start with its own individual E., just like the example above.
- The 'area number' is the first 2 digits of the area's vnum range. However, the area number is obsolete and a zero is sufficient for it.
- The 'room vnum' is the virtual number of the room.
- The 'room name' is the name of the room.
- The 'room description' is the long multi-line description of the room.
- The 'ROOM-FLAGS' describe more attributes of the room, and a listing of possible FLAGS is given at the end of the #ROOMS help section in this file.
- The 'SECTOR-TYPE' identifies the type of terrain. This affects movement cost through the room. Certain sector types (AIR and WATER) require special capabilities to enter, i.e., a flying potion or spell for air, and a boat or raft for water.
- Unlike mobiles and objects, rooms don't have any keywords associated with them. One may not manipulate a room in the same way one manipulates a mobile or object.
- The optional 'D' sections and 'E' sections come after the main data. A 'D' section contains one or more 'doors' in the range from 0 to 5. The numbers represent the 6 possible directions. These direction values are given at the end of this #ROOMS section. A 'D' command also contains a 'description' for that direction,

and 'keywords' for manipulating the door. 'Doors' include not just real doors, but ANY kind of exit from the room. The locks-number' defines whether or not the door is locked or not, and if it can be picked or not. The values for the 'locks-numbers' is given at the end of the #ROOMS section. The 'key-number' is the vnum of an object which locks and unlocks the door. IMPORTANT: If there is NO key for opening the door, either because the door is unlocked, is a simple, unhindered exit, or because you want a locked door with no key in the game, use '-1' for the 'key-number', NOT a '0'. Lastly, 'to_room-number' is the vnum of the room to which this door leads.

- Unless you want a one-way exit, you must specify two 'D' sections, one for each side of the door; i.e., one leaving one room and one in the other room the door goes to. otherwise, you will end up with a one-way exit.
- An 'E' section (extended description) contains a 'keywords' string and a 'description' string. As you might guess, looking at one of the words in the 'keywords' yields the 'description' string.
- The 'S' at the end marks the end of the room. IT IS NOT OPTIONAL.

When your #ROOMS sections is complete, to end the #ROOMS section, you must put '#0' on the line after the 'S' line of the last room in the #ROOMS section of your area.

Direction Values for the 'D' section:

NORTH	0	SOUTH	2	UP	4
EAST	1	WEST	3	DOWN	5

Locks-numbers Values for the 'D' section:

Unhindered exit	0	Hidden/Pickproof & Passproof	4	Hidden	7
Door	1	Passproof	5	Hidden/Pickproof	8
Pickproof	2	Pickproof/Passproof	6		
Hidden/Passproof	3				

Closed doors do not show in the exits command. Hidden doors MUST be searched for to be found. When using OLC it is important to be sure and flag your exits as doors if you want to use any other flag. Otherwise they will NOT work after reboot, and may cause the mud to not reboot correctly.

ROOM-FLAGS:

NOTE: Because of some vagary of OLC, OLC will save these flags in their numerical form. However, we can read the letter form without problem when loading an area.

ROOM_DARK	(A)	ROOM_IMP_ONLY	(O)
ROOM_NO_MOB	(C)	ROOM_GODS_ONLY	(P)
ROOM_INDOORS	(D)	ROOM_HEROES_ONLY	(Q)
ROOM_PRIVATE	(J)	ROOM_NEWBIES_ONLY	(R)
ROOM_SAFE	(K)	ROOM_LAW	(S)
ROOM_SOLITARY	(L)	ROOM_DONATION	(T)
ROOM_PET_SHOP	(M)	ROOM_NOTELEPORT	(U)
ROOM_NO_RECALL	(N)	ROOM_MARK	(V)

ROOM_NOMAGIC (W)

SECTOR-TYPES:

SECT_INSIDE	0	SECT_WATER_SWIM	6
SECT_CITY	1	SECT_WATER_NOSWIM	7
SECT_FIELD	2	SECT_UNUSED	8
SECT_FOREST	3	SECT_AIR	9
SECT_HILLS	4	SECT_DESERT	10
SECT_MOUNTAIN	5	SECT_MAX	11

VI. The #RESETS section:

NOTE: The resets system for Ember is in the process of being redone. The new reset system should debut sometime after 0.9.45. A design document is available from www.embermud.org. Any input you may have for a wish list for resets, is certainly appreciated.

The resets is what bring an area to live, cleans up the mess the players left, and in generally puts everthing in its place. This includes loading mobs, randomizing doors, loading objects, and equipping objects.

To reset an area, the server executes each command in the list of reset commands once. Each area is reset once when the server loads, and again periodically as it ages. An area is reset if it is at least 3 ticks old and is empty of players, or if it is 15 area-minutes old and has players in it. The only exception is the mud school area.

A tick varies between 30 and 90 seconds of real time, with an average of 60 seconds. The variation defeats area timekeepers. This number may vary depending on the speed of the server, since it is independent of the real time clock on the system.

The #RESETS section contains a series of single lines.
The reset commands are:
* comment
M read a mobile
O read an object
P put an object in an object (gold in a safe, etc.)
G give an object to mobile
E equip object to mobile
D set state of door
R randomize room exits
S stop (END OF LIST)

Here is the format for the various reset commands:
M <unused> <mobile-vnum> <limit-number> <room-vnum>
For the 'unused' value, simply put a '0'. The mobile-vnum is the vnum of the mobile loaded. The limit-number is the limit of how many of this mobile may be present in the

room. The last number, the room-vnum, is the vnum of the room where the mobile is loaded.

O <unused> <object-vnum> <limit number> <room-vnum>
For the 'unused' value, simply put a '0'. The object-vnum is the vnum of the object loaded. If you want more than one of these objects loaded into the room at one time, make it load twice by putting it twice in resets (or more). The last number, the room-vnum, is the vnum of the room where the object is loaded.

P <unused> <object-vnum> <limit number> <in_object-vnum>
The number is the number (amount) of the first object that is placed into another object (i.e., a desk, coffer, safe, etc.). The object-vnum is the vnum of the object loaded. To make more than one of the same item load into a container, load it twice (or more) in the resets. The limit number is currently not implemented in Ember.

G <number> <object-vnum> <limit-number>
For the 'G' command, there is no fourth number. Only use three numbers. The 'G' command MUST follow an 'M' command, as it 'gives' the object to the whatever mobile in a 'M' command is above it. Note the reason for this is due to the fact that no mobile-vnum is listed in the 'G' command. The first number is the number (amount) of this particular object to be given to the mobile. The object-vnum is the vnum of the object given. The limit number is not currently implemented in Ember. The resets system is undergoing a major revision.

E <number> <object-vnum> <limit-number> <wear_loc-number> The first number is the number (amount) of an object equipped on the mobile. The object-vnum is just that. The limit number is the same as in the 'G' command (see above). The 'wear_loc-number' is the wear location that the object is equipped on the mobile's body. A listing of the wear_loc values is given at the end of this #RESETS help section.

D <unused> <room-vnum> <door-number> <state-number>
For the 'unused' value, simply put a '0'. The room-vnum is the vnum of the room that the door IS IN. The door-number is just that; i.e., 0=north, 1=west, 2=south, etc. (see the help section for #ROOMS, above). The last number, the 'state-number', is the "state" of the door (whether it is open, closed, locked, etc.). A listing for the 'state-number' values is provided at the end of the #RESETS helps section.

R <unused> <room-vnum> <last_door-number>
For the 'R' command, there are only three values, and the first value is always 0 as it is unused. The second number is the vnum of a room. The third number is a last_door-number. When this command is used, the doors from 0 to the indicated 'last_door-number' are shuffled. The room will still have the same exits leading to the same other rooms as before, the DIRECTIONS will be different at each reset. Thus, a last_door-number of 4 makes a two-dimensional maze room; a door number of 6 makes a three-dimensional maze room. Use of both the 'D' and 'R' commands on the same room will yield UNPREDICTABLE results, so take care how you utilize the 'R' command.

Any line (except an 'S' line) may have a comment at the end.

NOTES:

- IMPORTANT!: End the #RESETS section in your area with an 'S' on the line after the last reset command, much the same as you do for the #ROOMS section.
- For the 'P' command, the actual container used is the MOST RECENTLY loaded object with the right vnum; for best results, there should be only one such container in the world. The is not loaded if no container object exists, or if someone is carrying it, or if it already contains on of the to-be-loaded objects.
- For all limit numbers use 0 which will be unlimited.
- For the 'E' and 'G' command, if the most recent 'M' command succeeded (e.g. the mobile limit wasn't exceeded), the object is given to that mobile. If the most recent 'M' command FAILED (due to hitting mobile limit), then the object is not loaded.
- Also remember, EACH MOBILE in your area (not just each type) must be loaded with the 'M' command. For example, if you have 20 cityguards total wandering about your town, you must do 20 reset lines of the 'M' command; for cityguard 1, cityguard 2, cityguard 3, etc., through cityguard 20. You can see how writing the #RESETS section of your area can rapidly become tiresome, but try to take extra care with this section, as it is easy to make a small typo, and these are a pain to debug if you end up with numerous reset errors!
- IMPORTANT!!: For the 'D' command: ROOM EXITS MUST BE COHERENT! If room 1 had an exit East going to room 2, and room 2 has an exit in the reverse direction (West), that exit MUST GO BACK to room 1. This doesn't prevent one-way exits; room 2 doesn't HAVE to have an exit in the reverse direction. Ember will still load these exits but may change them in unpredictable ways.

EXAMPLE:

The following is an example of several of the above reset commands as they appear in the area Gnome village. This is NOT the complete reset section but rather a selection to show examples. The text descriptions should be left blank when editing an read by hand.

```
#RESETS
D 0 1536 1 2
D 0 1545 3 2
D 0 1546 1 2
M 0 1502 0 1503 Load a gnome man
E 0 1503 0 16 A gnarled staff is loaded wielded of a gnome man
M 0 1500 0 1509 Load a gnome child
G 0 1502 0 A toy boomerang is given to a gnome child
M 0 1501 0 1509 Load a gnome woman
M 0 1503 0 1513 Load a gnome scientist
G 0 1511 0 A wicker basket is given to a gnome scientist
P 0 1513 0 1511 Some grain put inside some grain
M 0 1503 0 1515 Load a gnome scientist
E 0 1514 0 16 A skiff is loaded wielded of a gnome scientist
R 0 1548 4 Randomize A Damp Hallway
R 0 1549 4 Randomize A Damp Hallway
M 0 1515 0 1556 Load the fire bat
G 0 1505 0 A studded shield is given to the fire bat
```

```
G 0 1512 0 Some peanuts is given to the fire bat
G 0 1516 0 A pile of coins is given to the fire bat
O 0 1517 0 1557 A pile of coins loaded to The Altar
O 0 1518 0 1557 A pile of coins loaded to The Altar
M 0 1505 0 1558 Load the chief gnome
O 0 1521 0 1573 A large club loaded to Hobgoblin Armory
M 0 1517 2 1584 Load the hobgoblin soldier
M 0 1517 2 1584 Load the hobgoblin soldier
M 0 1526 0 1587 Load the cook
M 0 1524 0 1589 Load a hermit
M 0 1520 2 1590 Load a hobgoblin miner
S
```

Wear_loc Values for the 'E' reset command:

WEAR_NONE	-1	WEAR_LEGS	7	WEAR_WRIST_R	15
WEAR_LIGHT	0	WEAR_FEET	8	WEAR_WIELD	16
WEAR_FINGER_L	1	WEAR_HANDS	9	WEAR_HOLD	17
WEAR_FINGER_R	2	WEAR_ARMS	10	WEAR_SECOND_WIELD	
WEAR_NECK_1	3	WEAR_SHIELD	11		18
WEAR_NECK_2	4	WEAR_ABOUT	12	MAX_WEAR	19
WEAR_BODY	5	WEAR_WAIST	13		
WEAR_HEAD	6	WEAR_WRIST_L	14		

VII. The #PROGS section:

EmberMUD supports object, room and mob programs. These take the place of the #SPECIALS and perform similar functions. However, unlike specials they do not add as severely to the CPU utilization of the MUD. They are not hard coded into the MUD and so can be edited without having to recompile the MUD. Ember includes an OLC front end to editing the progs as well. Ember stores all of the progs for the MUD in a single file in the area directory. The #PROGS section tells the mud what progs to load to which mobs. Here is a small sample of the #PROGS section from a builder on Tempestuous Realms. Rajċ has created a whole series of progs for use in his

#PROGS
M 15200 80 Aggy-evil double Load to: Eldamar the Gatekeeper
M 15200 56 Evil Aggy Load to: Eldamar the Gatekeeper
O 15200 39 Refreshing fountain Load to: a tinkling fountain
M 15201 80 Aggy-evildouble Load to: The elven patrol
M 15201 56 Evil Aggy Load to: The elven patrol
M 15201 92 Rachee summon Load to: The elven patrol
R 15201 29 Arborlon Prog Load to: Heart of Arborlon
O 15202 72 Carla's recall Load to: a long wooden pench
R 15203 61 Web trap Load to: Main Eastern Forest Road
O 15203 69 Bradshao recall Load to: the statue of King Elissar
O 15204 60 Close Pine gate Load to: the Pine Tree Gate
R 15205 61 Web trap Load to: Ash Gate
S

The format is #PROGS to start the section.

M, O, or R for Mob, Object, or Room prog.
The vnum of the mob, room or object
The vnum of the prog from the MudProgs.prg file
Anything else to the end of the line is a comment, olc adds the comments seen above.
S on a line by itself ends the section.

VIII. The #SHOPS section:

If you intend on having shops in your area, then you will need a #SHOPS section. This section, in fact, is the easiest of all the sections in an area to do, and takes very little time.

First, I'll provide an example from the midgaard.are area file of some of the shops from Midgaard's #SHOPS section:

```
#SHOPS
3000  2  3  4  10 0  105  15  0  23
3001  0  0  0  0  0  110  100 0  23
3002  1  8  13 15 19 150  40  0  23
3003  5  0  0  0  0  130  40  0  23
3004  9  0  0  0  0  100  50  0  23
3006  22 0  0  0  0  120  90  6  22
```

The syntax of the #SHOPS is as follows:
<mobile-vnum> <trade-0> <trade-1> <trade-2> <trade-3> <trade-4> <profit-buy> <profit-sell> <open-hour> <close-hour>

NOTES:

- The first value, the mobile-vnum, is the the shopkeeper. All mobiles with that vnum will be shopkeepers.
- The trade-0 through trade-4 are item types which the shopkeeper will buy. Unused slots should have a '0' in them'; for instance, a shopkeeper who doesn't buy anything would have five zeroes. The item types are the same values from the #OBJECTS section, in the
- OBJECT TYPE table. It appears again at the end of this section.
- The 'profit-buy' number is a markup for players buying the item, in percentage points. 100 is nominal price; 150 is 50% markup, and so on. The 'profit-sell' number is a markdown for players selling the item, in percentage points. 100 is nominal price, 75 is 25% markdown, and so on. The buying markup should be at least 100, generally greater, and the selling markdown should be no more than 100, generally lower.
- The 'open-hour' and 'close-hour' numbers define the hours when the shopkeeper will do business. For a 24-hour shop, these numbers would be 0 and 23. Everything beyond 'close-hour' to the end of the line is taken to be a comment.
- There is no room number for a shop. Just load the shopkeeper mobile in to the room of your choice, via that #RESETS section, and make the mobile a sentinel in the ACT-FLAGS section of the mobile in #MOBILES. Or, for a wandering shopkeeper, just make it non-sentinel.

- The objects the shopkeeper sells are exactly those loaded by the 'G' reset command in #RESETS for that shopkeeper. These items replenish automatically. If a player sells an object to a shopkeeper, the shopkeeper will keep it for resale if he, she, or it doesn't already have an identical object. The items a player sells to a shopkeeper, however, do not replenish. Also objects with a 0 value will not be sold. Objects have to have a value to be traded.

You end your #SHOPS section with a '0' on the line after the last shop.

OBJECT TYPES:

ITEM_LIGHT	1	ITEM_CLOTHING	11	ITEM_BOAT	22
ITEM_SCROLL	2	ITEM_FURNITURE	12	ITEM_CORPSE_NPC	23
ITEM_WAND	3	ITEM_TRASH	13	ITEM_CORPSE_PC	24
ITEM_STAFF	4	ITEM_CONTAINER	15	ITEM_FOUNTAIN	25
ITEM_WEAPON	5	ITEM_DRINK_CON	17	ITEM_PILL	26
ITEM_TREASURE	8	ITEM_KEY	18	ITEM_PROTECT	27
ITEM_ARMOR	9	ITEM_FOOD	19	ITEM_MAP	28
ITEM_POTION	10	ITEM_MONEY	20	ITEM_PORTAL	29

In general, when designing a mobile as a shopkeeper in the #MOBILES section, you should make him/her IMMUNE to all forms of attack by having the mobile immune to magic, disease, poison, and weapons using the IMM_BITS in the immunities section of the mob in #MOBILES. Also, it is generally a good idea to make these mobiles NOPURGE by having an 'ACT_FLAG' of ACT_NOPURGE for the mobile in the #MOBILES section. After all, you don't want your players killing the shopkeepers, or a purge-happy immortal accidentally purging the shopkeeper with a purge command. You can still purge the mob by specifically purging it, if you are of high enough level.

IX. The #FACTIONAFFS section

The #FACTIONSAFF section is extremely simple. It is as follows:

#FACTIONAFFS to open the section
<mobvnum> <factionvnum> <change amount>
0 ← 0 on a line by itself ends the section.

For example:

#FACTIONAFFS
1200 1 10
1201 1 –50
0
In the above example, killing mob 1200 would raise your faction standing by 10, and kill mob 1201 would lower it by –50, both in relationship to faction #1 from the factions.dat file.

X. The #$ section:

This section marks the end of an area file. If you concentrate several area files into one, remember to delete the terminating '#$' from all but the last file. Conversely, if you split area files, remember to terminate EACH NEW FILE with a '#$'.

The syntax of this section is:

#$

---

[i] Most of the sections of this file have been changed, however it would not be possible without building on the previous files. The credits from the original file are:

```
*** compiled/written by: Ozymandias
-+-+ MadROM muds at: telnet mad.rom.org 1536 -+-+-
*** Acknowledgement:
This file contains material and information from the Merc release
2.1 Area help files (done by Furey, Hatchet, and Kahn), material
from the Merc Diku Mud code itself, material and examples from
various area files that are found in both ROM2 and MadROM, as well
as coding material that was adapted and modified by Alander for
ROM2, and later by Madman for MadROM.
```

[ii] Parse – according to Merriam-Websters dictionary - **a :** to resolve (as a sentence) into component parts of speech and describe them grammatically **b :** to describe grammatically by stating the part of speech and explaining the inflection and syntactical relationships. In computer terms this means to break into manageable chunks.