DIABLO II DATA FILE GUIDE

Table of Contents

actinfo.txt

Overview

This file controls global Act functionalities including item levels, monster behaviors, and waypoints

This file uses the wanderingmon.txt file for a modular list of potential wandering units to spawn

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

act - Defines the ID for the Act
town - Uses an area level (“Name field” from levels.txt) to define the Act’s town area
start - Uses an area level (“Name field” from levels.txt) to define where the player starts in the Act

maxnpcitemlevel - Controls the maximum item level for items sold by the NPC in the Act

classlevelrangestart - Uses an area level (“Name field” from levels.txt) with its MonLvl values as a global Act minimum monster level. For example, this is used to determine chest levels in an Act.
classlevelrangeend - Uses an area level (“Name field” from levels.txt) with its MonLvl values as a global Act maximum monster level. For example, this is used to determine chest levels in an Act.

wanderingnpcstart - Uses an index to determine which wandering monster class to use when populating areas (See wanderingmmon.txt for a list of possible monsters to spawn)
wanderingnpcrange - This is a modifier that gets added to the “wanderingnpcstart” value to randomly select an index

commonactcof - Specifies which “.D2” file to use as for the common Act COF file. This is used to establish the seed when initializing the Act.

waypoint1 (to waypoint9) - Uses an area level (“Name field” from levels.txt) as the designated waypoint selection in the Waypoint UI

wanderingMonsterPopulateChance - The percent chance (from 0 to 100) to spawn a wandering monster (See wanderingmmon.txt for a list of possible monsters to spawn)
wanderingMonsterRegionTotal - The maximum number of wandering monsters allowed at once
wanderingPopulateRandomChance - A secondary percent chance (from 0 to #) to determine whether to attempt populating with monsters. Only fails if random chance selects 0.

armor.txt

Overview

This file controls the functionalities for armor type items

This file is loaded together with other similar files in the following order: weapons.txt, armor.txt, misc.txt
These combined files form the items structure. Technically these files share the same fields, but some fields are exclusive for specific item types, so they are not displayed in the data files that do not need them.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

name - This is a reference field to define the item
version - Defines which game version to create this item (0 = Classic mode | 100 = Expansion mode)
compactsave - Boolean Field. If equals 1, then only the item's base stats will be stored in the character save, but not any modifiers or additional stats. If equals 0, then all of the items stats will be saved.
rarity - Determines the chance that the item will randomly spawn (1/#). The higher the value then the rarer the item will be. This field depends on the "spawnable" field being enabled, the "quest" field being disabled, and the item level being less than or equal to the area level. This value is also affected by the relative Act number that the item is dropping in, where the higher the Act number, then the more common the item will drop.
spawnable - Boolean Field. If equals 1, then this item can be randomly spawned. If equals 0, then this item will never randomly spawn.

speed - If the item type is an armor, then this will affect the Walk/Run Speed reduction when wearing the armor. If the item type is a weapon, then this will affect the Attack Speed reduction when wearing the weapon.
reqstr - Defines the amount of the Strength attribute needed to use the item
reqdex - Defines the amount of the Dexterity attribute needed to use the item

durability - Defines the base durability amount that the item will spawn with.
nodurability - Boolean Field. If equals 1, then the item will not have durability. If equals 0, then the item will have durability.

level - Controls the base item level. This is used for determining when the item is allowed to drop, such as making sure that the item level is not greater than the monster's level or the area level.
ShowLevel - Boolean Field. If equals 1, then display the item level next to the item name. If equals 0, then ignore this.
levelreq - Controls the player level requirement for being able to use the item

cost - Defines the base gold cost of the item when being sold by an NPC. This can be affected by item modifiers and the rarity of the item.
gamble cost - Defines the gambling gold cost of the item on the Gambling UI

code - Defines a unique 3 letter/number code for the item. This is used as an identifier to reference the item.
namestr - String Key that is used for the base item name

magic lvl - Defines the magic level of the item, which can affect how magical item modifiers that can appear on the item (See automagic.txt)
auto prefix - Automatically picks an item affix name from a designated "group" value from the automagic.txt file, instead of using random prefixes. This is only used when the item is Magical quality.

alternategfx - Uses a unique 3 letter/number code similar to the defined "code" fields to determine what in-game graphics to display on the player character when the item is equipped

normcode - Links to a "code" field to determine the normal version of the item
ubercode - Links to a "code" field to determine the Exceptional version of the item
ultracode - Links to a "code" field to determine the Elite version of the item

component - Determines the layer of player animation when the item is equipped. This uses a code referenced from the Composit.txt file.

Code	Description
0	Head
1	Torso
2	Legs
3	Right Arm
4	Left Arm
5	Right Hand
6	Left Hand
7	Shield
8	Special 1
9	Special 2
10	Special 3
11	Special 4
12	Special 5
13	Special 6
14	Special 7
15	Special 8
16	Do not display anything

invwidth & **invheight -** Defines the width and height of grid cells that the item occupies in the player inventory

hasinv - Boolean Field. If equals 1, then the item will have its own inventory allowing for the capability to socket gems, runes, or jewels. If equals 0, then the item cannot have sockets.
gemsockets - Controls the maximum number of sockets allowed on this item. This is limited by the item's size based on the "invwidth" and "invheight" fields. This also compares with the "MaxSock1", "MaxSock25" and "MaxSock40" fields from the ItemTypes.txt file.
gemapplytype - Determines which affect from a gem or rune will be applied when it is socketed into this item (See gems.txt)

Code	Description
0	Weapon
1	Armor or Helmet
2	Shield

flippyfile - Controls which DC6 file to use for displaying the item in the game world when it is dropped on the ground (uses the file name as the input)
invfile - Controls which DC6 file to use for displaying the item graphics in the inventory (uses the file name as the input)
uniqueinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Unique quality item (uses the file name as the input)
setinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Set quality item (uses the file name as the input)

useable - Boolean Field. If equals 1, then the item can be used with the right-click mouse button command (this only works with specific belt items or quest items). If equals 0, then ignore this.

stackable - Boolean Field. If equals 1, then the item will use a quantity field and handle stacking functionality. This can depend on if the item type is throwable, is a type of ammunition, or is some other kind of miscellaneous item. If equals 0, then the item cannot be stacked.
minstack - Controls the minimum stack count or quantity that is allowed on the item. This field depends on the “stackable” field being enabled.
maxstack - Controls the maximum stack count or quantity that is allowed on the item. This field depends on the “stackable” field being enabled.
spawnstack - Controls the stack count or quantity that the item can spawn with. This field depends on the “stackable” field being enabled.

Transmogrify - Boolean Field. If equals 1, then the item will use the transmogrify function. If equals 0, then ignore this. This field depends on the “useable” field being enabled.
TMogType - Links to a “code” field to determine which item is chosen to transmogrify this item to.
TMogMin - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the “TMogType” field, and that the transmogrify item has quantity.
TMogMax - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the “TMogType” field, and that the transmogrify item has quantity.

type - Points to an Item Type defined in the ItemTypes.txt file, which controls how the item functions
type2 - Points to a secondary Item Type defined in the ItemTypes.txt file, which controls how the item functions. This is optional but can add more functionalities and possibilities with the item.

dropsound - Points to sound defined in the sounds.txt file. Used when the item is dropped on the ground.
dropsfxframe - Defines which frame in the “flippyfile” animation to play the “dropsound” sound when the item is dropped on the ground.
usesound - Points to sound defined in the sounds.txt file. Used when the item is moved in the inventory or used.

unique - Boolean Field. If equals 1, then the item can only spawn as a Unique quality type. If equals 0, then the item can spawn as other quality types.

transparent - Boolean Field. If equals 1, then the item will be drawn transparent on the player model (similar to ethereal models). If equals 0, then the item will appear solid on the player model.
transtbl - Controls what type of transparency to use, based on the “transparent” field being enabled.

Code	Description
0	Transparency at 25%
1	Transparency at 50%
2	Transparency at 75%
3	Black Alpha Transparency
4	White Alpha Transparency
5	No Transparency
6	Dark Transparency (Unused)
7	Highlight Transparency (Used when mousing over the unit)
8	Blended

lightradius - Controls the value of the light radius that this item can apply on the monster. This only affects monsters with this item equipped, not other types of units. This is ignored if the item’s component on the monster is “lit”, “med”, or “hvy”.

belt - Controls which belt type to use for belt items only. This field determines what index entry in the belts.txt file to use.

quest - Controls what quest class is tied to the item which can enable certain item functionalities for a specific quest. Any value greater than 0 will also mean the item is flagged as a quest item, which can affect how it is displayed in tooltips, how it is traded with other players, its item rarity, and how it cannot be sold to an NPC. If equals 0, then the item will not be flagged as a quest item.

Code	Description
0	Not a quest item
1	Act 1 Prologue
2	Den of Evil
3	Sisters’ Burial Grounds
4	Tools of the Trade
5	The Search for Cain
6	The Forgotten Tower
7	Sisters to the Slaughter
8	Act 2 Prologue
9	Radament’s Lair
10	The Horadric Staff
11	The Tainted Sun
12	The Arcane Sanctuary
13	The Summoner
14	The Seven Tombs
15	Act 2 Traversed
16	Lam Esen’s Tome
17	Khalim’s Will
18	Blade of the Old Religion
19	The Golden Bird
20	The Blackened Temple
21	The Guardian
22	Act 4 Prologue
23	The Fallen Angel
24	Terror’s End
25	The Hellforge
26	Rogue Warning
27	Guard in Town Warning

28	Guard in Desert Warning
29	Dark Wanderer Seen
30	Angel Warning
31	Respec from Akara Complete Act 5 Prologue
32	Siege on Harrogath
33	Rescue on Mount Arreat
34	Prison of Ice
35	Betrayal of Harrogath
36	Rite of Passage
37	Eve of Destruction

questdiffcheck - Boolean Field. If equals 1 and the “quest” field is enabled, then the game will check the current difficulty setting and will tie that difficulty setting to the quest item. This means that the player can have more than 1 of the same quest item as long each they are obtained per difficulty mode (Normal / Nightmare / Hell). If equals 0 and the “quest” field is enabled, then the player can only have 1 count of the quest item in the inventory, regardless of difficulty.

missiletype - Points to the “Id” field from the Missiles.txt file, which determines what type of missile is used when using the throwing weapons
durwarning - Controls the threshold value for durability to display the low durability warning UI. This is only used if the item has durability.
qntwarning - Controls the threshold value for quantity to display the low quantity warning UI. This is only used if the item has stacks.

mindam - The minimum physical damage provided by the item
maxdam - The maximum physical damage provided by the item

StrBonus - The percentage multiplier that gets multiplied the player’s current Strength attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.
DexBonus - The percentage multiplier that gets multiplied the player’s current Dexterity attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.

gemoffset - Determines the starting index offset for reading the gems.txt file when determining what effects gems or runes will have the item based on the “gemapplytype” field. For example, if this value equals 9, then the game will start with index 9 (“Chipped Emerald”) and ignore the previously defined gems in the gems.txt file, which can mean that those ignored gems will not apply modifiers when socketed into the item.
bitfield1 - Controls different flags that can affect the item. Uses an integer value to check against different bit fields by using the “&” operator. For example, if the value equals 5 (binary = 101) then that returns true for both the 4 (binary = 100) and 1 (binary = 1) bit field values.

Bit Field One Bits	Binary Equivalent Value	Description
1	1	Allow the item to be capable of having Magic quality
2	10	The item is classified as metal
4	100	The item is classified as a spellcaster item (currently does nothing)
8	1000	The item is classified as a skill based item (currently does nothing)

The following fields are separated per NPC in each Act:
[NPC]Min - Minimum amount of this item type in Normal rarity that the NPC can sell at once
[NPC]Max - Maximum amount of this item type in Normal rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicMin - Minimum amount of this item type in Magical rarity that the NPC can sell at once
[NPC]MagicMax - Maximum amount of this item type in Magical rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicLvl - Maximum magic level allowed for this item type in Magical rarity
Where [NPC] is one of the following:

Charsi
Gheed
Akara
Fara
Lysander
Drognan
Hratli
Alkor
Ormus
Elzix
Asheara
Cain
Halbu
Jamella
Larzuk
Malah
Anya

Transform - Controls the color palette change of the item for the character model graphics
InvTrans - Controls the color palette change of the item for the inventory graphics

Code	Color
0	No color change
1	Grey
2	Grey 2
3	Gold
4	Brown
5	Grey Brown
6	Inventory Grey
7	Inventory Grey 2
8	Inventory Grey Brown

SkipName - Boolean Field. If equals 1 and the item is Unique rarity, then skip adding the item's base name in its title. If equals 0, then ignore this.

NightmareUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Nightmare difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.
HellUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Hell difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.

Nameable - Boolean Field. If equals 1, then the item's name can be personalized by Anya for the Act 5 Betrayal of Harrogath quest reward. If equals 0, then the item cannot be used for the personalized name reward.
PermStoreItem - Boolean Field. If equals 1, then this item will always appear on the NPC's store. If equals 0, then the item will randomly appear on the NPC's store when appropriate.
diablocloneweight - The amount of weight added to the diablo clone progress when this item is sold. When offline, selling this item will instead immediately spawn diablo clone.

The following fields are exclusive to the armor.txt file, because these fields only work with Armor type items:

minac - The minimum amount of Defense that an armor item type can have
maxac - The maximum amount of Defense that an armor item type can have

block - Controls the block percent chance that the item provides (out of 100, but caps at 75).

rArm - Controls the character's graphics and animations for the Right Arm component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"
lArm - Controls the character's graphics and animations for the Left Arm component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"
Torso - Controls the character's graphics and animations for the Torso component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"
Legs - Controls the character's graphics and animations for the Legs component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"
rSPad - Controls the character's graphics and animations for the Right Shoulder Pad component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"
lSPad - Controls the character's graphics and animations for the Left Shoulder Pad component when wearing the armor, where the value 0 = Light or "lit", 1 = Medium or "med", and 2 = Heavy or "hvy"

automagic.txt

Overview

This file controls what item affixes (groups of item modifiers) are automatically applied to items, regardless of their quality type. These item affixes will not change the quality of the item.
An example can be class based items like a paladin shield that can be Normal Quality but still have Paladin skill bonuses without changing the item's name.

This file is loaded together with other similar files in the following order: magicsuffix.txt, magicprefix.txt, automagic.txt
These combined files form the Item Mods structure.

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

Name - Defines the item affix name
version - Defines which game version to use this item affix (<100 = Classic mode | 100 = Expansion mode)

spawnable - Boolean Field. If equals 1, then this item affix is used as part of the game's randomizer for assigning item modifiers when an item spawns. If equals 0, then this item affix is never used.
rare - Boolean Field. If equals 1, then this item affix can be used when randomly assigning item modifiers when a rare item spawns. If equals 0, then this item affix is not used for rare items.

level - The minimum item level required for this item affix to spawn on the item. If the item level is below this value, then the item affix will not spawn on the item.
maxlevel - The maximum item level required for this item affix to spawn on the item. If the item level is above this value, then the item affix will not spawn on the item.
levelreq - The minimum character level required to equip an item that has this item affix

classspecific - Controls if this item affix should only be used for class specific items. This relies on the class specified in the "Class" field from ItemTypes.txt, for the specific item.

Code	Description
(empty)	Any Class
ama	Amazon only
bar	Barbarian only
pal	Paladin only
nec	Necromancer only
sor	Sorceress only
dru	Druid only
ass	Assassin only

class - Controls which character class is required for the class specific level requirement “classlevelreq” field

Code	Description
(empty)	None
ama	Amazon
bar	Barbarian
pal	Paladin
nec	Necromancer
sor	Sorceress
dru	Druid
ass	Assassin

classlevelreq - The minimum character level required for a specific class in order to equip an item that has this item affix. This relies on the class specified in the “class” field. If equals null, then the class will default to using the “levelreq” field.

frequency - Controls the probability that the affix appears on the item (a higher value means that the item affix will appear on the item more often). This value gets summed together with other “frequency” values from all possible item affixes that can spawn on the item, and then is used as a denominator value for the randomizer. Whichever item affix is randomly selected will be the one to appear on the item. The formula is calculated as the following: [Item Affix Selected] = [“frequency”] / [Total Frequency]. If the item has a magic level (from the “magic lvl” field in weapons.txt/armor.txt/misc.txt) then the magic level value is multiplied with this value. If equals 0, then this item affix will never appear on an item.
group - Assigns an item affix to a specific group number. Items cannot spawn with more than 1 item affix with the same group number. This is used to guarantee that certain item affixes do not overlap on the same item. If this field is null, then the group number will default to group 0.

mod1code (to mod3code) - Controls the item properties for the item affix (Uses the “code” field from Properties.txt)
mod1param (to mod3param) - The “parameter” value associated with the related property (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)
mod1min (to mod3min) - The “min” value to assign to the listed related (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)
mod1max (to mod3 max) - The “max” value to assign to the listed related (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)

transformcolor - Controls the color change of the item after spawning with this item affix. If empty, then the item affix will not change the item’s color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

itype1 (to itype7) - Controls what Item Types are allowed to spawn with this item affix. Uses the “code” field from ItemTypes.txt
etype1 (to etype5) - Controls what Item Types are forbidden to spawn with this item affix. Uses the “code” field from ItemTypes.txt

multiply - Multiplicative modifier for the item’s buy and sell costs, based on the item affix (Calculated in 1024ths for buy cost and 4096ths for sell cost)
add - Flat integer modification to the item’s buy and sell costs, based on the item affix

AutoMap.txt

Overview

This file controls how the Automap in game will display the discovered parts of the area level and store this progress in character save files.

The Automap is composed of many different image files depicted as small icons to convey what part of the area level is being displayed. This file will assign these image files to their related map cells, which will properly build the Automap as the player explores the area.

Not all tiles will have image files assigned to them, and in these cases, those parts of the Automap will remain blank.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

LevelName - Uses a string format system to define the Act number and name of the level type. Level types are static defined values that cannot be added. The number at the start of the string defines the Act number, and the word that follows this number defines the level type. This data should stay grouped by level.

String Code	Description
	Level Type None
1 Town	Level Type 1 Town
1 Wilderness	Level Type 1 Wilderness
1 Cave	Level Type 1 Cave
1 Crypt	Level Type 1 Crypt
1 Monestary	Level Type 1 Monestary
1 Courtyard	Level Type 1 Courtyard
1 Barracks	Level Type 1 Barracks
1 Jail	Level Type 1 Jail
1 Cathedral	Level Type 1 Cathedral
1 Catacombs	Level Type 1 Catacombs
1 Tristram	Level Type 1 Tristram
2 Town	Level Type 2 Town
2 Sewer	Level Type 2 Sewer
2 Harem	Level Type 2 Desert
2 Basement	Level Type 2 Sewer
2 Desert	Level Type 2 Desert
2 Tomb	Level Type 2 Tomb
2 Lair	Level Type 2 Lair
2 Arcane	Level Type 2 Arcane
3 Town	Level Type 3 Town
3 Jungle	Level Type 3 Jungle
3 Kurast	Level Type 3 Kurast
3 Spider	Level Type 3 Spider
3 Dungeon	Level Type 3 Dungeon
3 Sewer	Level Type 3 Sewer
4 Town	Level Type 4 Town
4 Mesa	Level Type 4 Mesa
4 Lava	Level Type 4 Hell
5 Town	Level Type 6 Town
5 Siege	Level Type 6 Siege
5 Barricade	Level Type 6 Barricade
5 Temple	Level Type 6 Temple
5 Ice	Level Type 5 Ice Caves
5 Baal	Level Type 5 Baal
5 Lava	Level Type 5 Lava

TileName - Uses defined string codes to control the tile orientations on the Automap

String Code	Description
fl	Base Floor
wl	Base Left Wall
wr	Base Right Wall
wtlr	Base Upper Top Corner Right
wtll	Base Upper Top Corner Left
wtr	Base Upper Top Corner
wbl	Base Lower Bottom Corner Left
wbr	Base Lower Bottom Corner Right
wld	Base Left Door
wrd	Base Right Door
wle	Base Left Exit
wre	Base Right Exit
co	Base Column
sh	Base Shadow
tr	Base Tree
rf	Base Roof
ld	Base Left Wall Down
rd	Base Right Wall Down
fd	Base Full Wall Down
fi	Base Front Wall Down

Style - Defines a group numeric ID for the range of cells, meaning that the game will try to use cells that match the same style value, after determining the Level Type and Tile Type. If this value is equal to 255, then the style is ignored in the “Cel#” field selection.

StartSequence - The start index value for valid “Cel#” field to choose for displaying on the Automap. If this value is equal to 255, then both the “StartSequence” and “EndSequence” are ignored in the “Cel#” field selection. If this value is equal to -1, then this field is ignored in the “Cel#” field selection.
EndSequence - The end index value for a valid “Cel#” field to choose for displaying on the Automap. If this value is equal to -1, then this field is ignored in the “Cel#” field selection.

Cel1 (to Cel4) - Determines the unique image frame to use from the MaxiMap.dc6 file that will be used to display on the Automap for that position of the level tile. There are multiple of these fields because they can be randomly chosen to give image variety in the Automap display. If the value equals -1, then this cell is not valid and will be ignored. If no cell is chosen overall, then nothing will be drawn in this area on the Automap.

belts.txt

Overview

This file controls the statistics for how belts and their various item slots work.

This file relies on the “belt” field from the armor.txt file. Each belt entry in this file defines a belt type that controls how many slots the belt item provides. Each of these belt types are a possible value that items in the armor.txt file can use in the “belt” field.

The game uses the 3rd entry is defined as the “default” belt, meaning that the player has no belt equipped, and the game will use this entry’s stats to determine how to handle the belt slots.

Data Fields

name - This is a reference field to define the belt type
numboxes - This integer field defines the number of item slots in the belt. This is used when inserting items into the belt and also for handling the removal of items when the belt item is unequipped.

box1left (to box 16left) - Specifies the belt slot left side coordinates. This is use for Server verification purposes and does not affect the local box UI in the client.
box1right (to box16right) - Specifies the belt slot right side coordinates. This is use for Server verification purposes and does not affect the local box UI in the client.
box1top (to box16top) - Specifies the belt slot left top coordinates. This is use for Server verification purposes and does not affect the local box UI in the client.
box1bottom (to box16bottom) - Specifies the belt slot bottom side coordinates. This is use for Server verification purposes and does not affect the local box UI in the client.
defaultItemTypeCol1 (to defaultItemTypeCol4) - Specifies the default item type used for the populate belt and auto-use functionality on controller.
defaultItemCodeCol1 (to defaultItemCodeCol4) - Specifies the default item code used for the populate belt and auto-use functionality on controller. Leaving this blank uses no code and instead relies entirely on the item type.

books.txt

Overview

This file controls functionalities of book items (also called Tomes). This includes how they interact with their related scroll items.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - This is a reference field to define the book
ScrollSpellCode - Uses an item’s code to define as the scroll item for the book
BookSpellCode - Uses an item’s code to define as the book item
pSpell - Defines the item spell function to use when using the book

Code	Description
0	None
1	Identify
2	Town Portal
3	Health Potion (Flat Amount with special bonus for class and Vitality attribute)
4	Health Potion 2 (Flat Amount with no special bonus)
5	Health Potion 3 (Percentage)
6	Antidote Potion (Apply a state that can override other states)
7	Transmogrify (Used to open the Horadric Cube’s UI for transmogrification)
8	Elixir (Modify a stat)
9	Herb (Apply a state using item stats)
10	Skill (Cast the Fireball skill to a target unit)
11	Skill XY (Cast the Fireball skill to a location using X and Y coordinates)

SpellIcon - Controls which DC6 file to display for the mouse cursor when using the scroll or book (Uses numeric indices to pick the DC6 file. Example: When using Identify, use icon 1 or buysell.DC6)
ScrollSkill - Defines which Skill to use for the scroll item (uses the “skill” field from skills.txt)
BookSkill - Defines which Skill to use for the book item (uses the “skill” field from skills.txt)

BaseCost - The starting gold cost to buy the book from an NPC
CostPerCharge - The additional gold cost added with the book’s “BaseCost” value, based on how many charges the book has

charstats.txt

Overview

This file controls the starting stats for each of the classes

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

class - The name of the character class (this cannot be changed)

str - Starting amount of the Strength attribute
dex - Starting amount of the Dexterity attribute
int - Starting amount of the Energy attribute
vit - Starting amount of the Vitality attribute

stamina - Starting amount of Stamina
hpadd - Bonus starting Life value (This value gets added with the "vit" field value to determine the overall starting amount of Life)
ManaRegen - Number of seconds to regain max Mana. (If this equals 0 then it will default to 300 seconds)
ToHitFactor - Starting amount of Attack Rating

WalkVelocity - Base Walk movement speed
RunVelocity - Base Run movement speed
RunDrain - Rate at which Stamina is lost while running

LifePerLevel - Amount of Life added for each level gained (Calculated in fourths and is divided by 256)
StaminaPerLevel - Amount of Stamina added for each level gained (Calculated in fourths and is divided by 256)
ManaPerLevel - Amount of Mana added for each level gained (Calculated in fourths and is divided by 256)

LifePerVitality - Amount of Life added for each point in Vitality (Calculated in fourths and is divided by 256)
StaminaPerVitality - Amount of Stamina added for each point in Vitality (Calculated in fourths and is divided by 256)

ManaPerMagic - Amount of Mana added for each point in Energy (Calculated in fourths and is divided by 256)

StatPerLevel - Amount of Attribute stat points earned for each level gained
SkillsPerLevel - Amount of Skill points earned for each level gained

LightRadius - Baseline radius size of the character's Light Radius
BlockFactor - Baseline percent chance for Blocking
MinimumCastingDelay - Global delay on all Skills after using a Skill with a Casting Delay (Calculated in Frames, where 25 Frames = 1 Second)

StartSkill - Controls what skill will be added by default to the character's starting weapon and will be slotted in the Right Skill selection (Uses the "skill" field from skills.txt)
Skill 1 (to Skill 10) - Skill that the character starts with and will always have available (Uses the "skill" field from skills.txt)

StrAllSkills - String key for displaying the item modifier bonus to all skills for the class (Ex: "+1 to Barbarian Skill Levels")
StrSkillTab1 - String key for displaying the item modifier bonus to all skills for the class's first skill tab (Ex: "+1 to Warcries")
StrSkillTab2 - String key for displaying the item modifier bonus to all skills for the class's second skill tab (Ex: "+1 to Combat Skills")
StrSkillTab3 - String key for displaying the item modifier bonus to all skills for the class's third skill tab (Ex: "+1 to Masteries")
StrClassOnly - String key for displaying on item modifier exclusive to the class or for class specific items (Ex: "Barbarian only")

HealthPotionPercent - This scales the amount of Life that a Healing potion will restore based on the class
ManaPotionPercent - This scales the amount of Mana that a Mana potion will restore based on the class

baseWClass - Base weapon class that the character will use by default when no weapon is equipped

Code	Description
hth	Hand to Hand (Default value if the value is empty)
1hs	One Handed Swing
1ht	One Handed Thrust
bow	Bow
2hs	Two Handed Swing
2ht	Two Handed Thrust
1js	Dual Wielding - Left Jab Right Swing
1jt	Dual Wielding - Left Jab Right Thrust
1ss	Dual Wielding - Left Swing Right Swing
1st	Dual Wielding - Left Swing Right Thrust
stf	Staff
xbw	Crossbow
ht1	One Hand to Hand
ht2	Two Hand to Hand

item1 (to item10) - Item that the character starts with (Uses ID pointer from Weapons.txt, Armor.txt or Misc.txt)
item1loc (to item10loc) - Location where the related item will be placed in the character's inventory

Code	Description
(empty)	Inventory grid (Default Value)
head	Head
neck	Neck
tors	Torso
rarm	Right Arm

larm	Left Arm
rrin	Right Ring
lrin	Left Ring
belt	Belt
feet	Feet
glov	Gloves

item1count (to item10count) - The amount of the related item that the character starts with
item1quality (to item10quality) - Controls the quality level of the related item

Item Quality Code	Description
0	Any Quality (Used for a random quality)
1	Low Quality (Ex: "Crude")
2	Normal Quality (Default value if the value is empty)
3	High Quality (Superior)
4	Magic Quality (Uses Magic Prefixes and Suffixes)
5	Set Item
6	Rare Quality
7	Unique (Predetermined stats)
8	Crafted
9	Tempered

cubemain.txt

Overview

This file controls the recipes for the Horadric Cube

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

description - This is a reference field to define the cube recipe
enabled - Boolean field. If equals 1, then the recipe can be used in-game. If equals 0, then the recipe cannot be used in-game.
firstLadderSeason - Integer field. The first ladder season this cube recipe can be made on (inclusive). If blank or 0 then it is available in non-ladder.
lastLadderSeason - Integer field. The last ladder season this cube recipe is ladder-only (inclusive). Must be used in conjunction with firstLadderSeason.
min diff - The minimum game difficulty to use the recipe (0 = All Game Difficulties | 1 = Nightmare and Hell Difficulty only | 2 = Hell Difficulty only)
version - Defines which game version to use this recipe (0 = Classic mode | 100 = Expansion mode)

op - Uses a function as an additional input requirement for the recipe

Op ID	Parameters	Description
(empty)		Do nothing
1	param value	Require that the current day of the month is less than “param” or greater than “value”
2	value	Require that the current day of the week does not equal “value” (0=None, 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday)
3	param value	Require that the player’s current stat (using “param” as the stat ID from ItemStatCost.txt) is greater than “value”
4	param value	Require that the player’s current stat (using “param” as the stat ID from ItemStatCost.txt) is less than “value”
5	param value	Require that the player’s current stat (using “param” as the stat ID from ItemStatCost.txt) is not equal to “value”
6	param value	Require that the player’s current stat (using “param” as the stat ID from ItemStatCost.txt) is equal to “value”
7	param value	Require that the player’s base stat (using “param” as the stat ID from ItemStatCost.txt) is greater than “value”
8	param value	Require that the player’s base stat (using “param” as the stat ID from ItemStatCost.txt) is less than “value”
9	param value	Require that the player’s base stat (using “param” as the stat ID from ItemStatCost.txt) is not equal to “value”
10	param value	Require that the player’s base stat (using “param” as the stat ID from ItemStatCost.txt) is equal to “value”
11	param value	Require that the player’s non-base stat (using “param” as the stat ID from ItemStatCost.txt) is greater than “value”
12	param value	Require that the player’s non-base stat (using “param” as the stat ID from ItemStatCost.txt) is less than “value”
13	param value	Require that the player’s non-base stat (using “param” as the stat ID from ItemStatCost.txt) is not equal to “value”
14	param value	Require that the player’s non-base stat (using “param” as the stat ID from ItemStatCost.txt) is equal to “value”
15	param value	Require that the input item’s current stat (using “param” as the stat ID from ItemStatCost.txt) is greater than “value”
16	param value	Require that the input item’s current stat (using “param” as the stat ID from ItemStatCost.txt) is less than “value”
17	param value	Require that the input item’s current stat (using “param” as the stat ID from ItemStatCost.txt) is not equal to “value”
18	param	Require that the input item’s current stat (using “param” as the stat ID from ItemStatCost.txt) is equal to “value”

	value	
19	param value	Require that the input item's base stat (using "param" as the stat ID from ItemStatCost.txt) is greater than "value"
20	param value	Require that the input item's base stat (using "param" as the stat ID from ItemStatCost.txt) is less than "value"
21	param value	Require that the input item's base stat (using "param" as the stat ID from ItemStatCost.txt) is not equal to "value"
22	param value	Require that the input item's base stat (using "param" as the stat ID from ItemStatCost.txt) is equal to "value"
23	param value	Require that the input item's non-base stat (using "param" as the stat ID from ItemStatCost.txt) is greater than "value"
24	param value	Require that the input item's non-base stat (using "param" as the stat ID from ItemStatCost.txt) is less than "value"
25	param value	Require that the input item's non-base stat (using "param" as the stat ID from ItemStatCost.txt) is not equal to "value"
26	param value	Require that the input item's non-base stat (using "param" as the stat ID from ItemStatCost.txt) is equal to "value"
27	value	Require that the item's Mod Class is not equal to "value". An item's Mod Class value can be the item's unique ID or quality type, depending on the case.
28		Checks that the item has the Quest and QuestDiffCheck flag, then ensures that the item's quest difficulty is greater than the game's difficulty level

param - Integer value used as a possible parameter for the "op" function
value - Integer value used as a possible parameter for the "op" function

class - Defines the recipe to be only usable by a defined class

Code	Description
(empty)	Any Class
ama	Amazon only
bar	Barbarian only
pal	Paladin only
nec	Necromancer only
sor	Sorceress only
dru	Druid only
ass	Assassin only

numinputs - Controls the number of items that need to be inside the cube for the recipe
input 1 (to input7) - Controls what items are required for the recipe. Uses the item's unique code. Users can also add input parameters by adding a comma "," to the input and using a code.

Code	Description
qty=#	The number (#) of this item type required for the recipe
low	Low Quality
nor	Normal Quality
hiq	High Quality (Superior)
mag	Magic Item
set	Set Item
rar	Rare Item
uni	Unique Item
crf	Crafted Item
tmp	Tempered Item
nos	Item with no sockets
sock=#	Item with sockets, where # defines the number of sockets
noe	Item that is not Ethereal
eth	Item that is Ethereal
upg	Item that allows Upgrades
bas	Basic Item
exc	Exceptional Item
eli	Elite Item
nru	Item is not a Rune Word

output - Controls the first output item. Uses the item's unique code. Users can also add output parameters by adding a comma "," to the output and using a code.

Code	Description
Cow Portal	Special code to create the Portal to the Moo Moo Farm
Pandemonium Portal	Special code to randomly create 1 of the 3 Pandemonium Portals: (The Matron's Den / The Forgotten Sands / The Furnace of Pain) (Does not create duplicate portals in the same game)
Pandemonium Finale Portal	Special code to create the Portal to Uber Tristram
Red Portal	Special code to create a permanent Red Portal to a Level ID. The Level ID is determined by the output "qty=#" code.
usetype	Use the same item type as "input 1" for the output item's type
useitem	Use the item from "input 1" as the output item
qty=#	The number (#) of this item type created
pre=#	Force the output item to have an item prefix, where # equals the ID of the prefix (see the row count on MagicPrefix.txt)
suf=#	Force the output item to have an item suffix, where # equals the ID of the suffix (see the row count on MagicSuffix.txt)
low	Low Quality Item
nor	Normal Item
hiq	High Quality Item (Superior)

mag	Magic Item
set	Set Item
rar	Rare Item
uni	Unique Item
crf	Crafted Item
tmp	Tempered Item
eth	Ethereal Item
sock	Item with sockets, where # defines the number of sockets
mod	Use the item modifiers from “input 1” as the output item’s modifiers
uns	Destroy all gems/runes/jewels in the item’s sockets
rem	Remove all gems/runes/jewels in the item’s sockets
reg	If the function has “usetype” and if the item is a Unique, then regenerate/reroll the Unique
exc	Exceptional Item
eli	Elite Item
rep	Repair the Item
rch	Recharge all of the skill charges on the Item
lvl=#	The number (#) of this item type created (same as “qty=#”)

lvl - Forces the output item level to be a specific level. If this field is used, then ignore the “plvl” and “ilvl” fields.
plvl - This is a numeric ratio that gets multiplied with the current player’s level, to add to the output item’s level requirement
ilvl - This is a numeric ratio that gets multiplied with “input 1” item’s level, to add to the output item’s level requirement
mod 1 (to mod 5) - Controls the output item properties (Uses the “code” field from Properties.txt)
mod 1 chance (to mod 5 chance) - The percent chance that the property will be assigned. If this equals 0, then the Property will always be assigned.
mod 1 param (to mod 5 param) - The “parameter” value associated with the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
mod 1 min (to mod 5 min) - The “min” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
mod 1 max (to mod 5 max) - The “max” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)

output b - Controls the second output item. Uses the item’s unique code. Users can also add output parameters by adding a comma “,” to the output and using a code. (See “output” for more details)
b lvl - Forces the output item level to be a specific level. If this field is used, then ignore the “plvl” and “ilvl” fields.
b plvl - This is a numeric ratio that gets multiplied with the current player’s level, to add to the output item’s level requirement
b ilvl - This is a numeric ratio that gets multiplied with “input 2” item’s level, to add to the output item’s level requirement
b mod 1 (to b mod 5) - Controls the output item properties (Uses the “code” field from Properties.txt)
b mod 1 chance (to b mod 5 chance) - The percent chance that the property will be assigned. If this equals 0, then the Property will always be assigned.
b mod 1 param (to b mod 5 param) - The “parameter” value associated with the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
b mod 1 min (to b mod 5 min) - The “min” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
b mod 1 max (to b mod 5 max) - The “max” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)

output c - Controls the third output item. Uses the item’s unique code. Users can also add output parameters by adding a comma “,” to the output and using a code. (See “output” for more details)
c lvl - Forces the output item level to be a specific level. If this field is used, then ignore the “plvl” and “ilvl” fields.
c plvl - This is a numeric ratio that gets multiplied with the current player’s level, to add to the output item’s level requirement
c ilvl - This is a numeric ratio that gets multiplied with “input 3” item’s level, to add to the output item’s level requirement
c mod 1 (to c mod 5) - Controls the output item properties (Uses the “code” field from Properties.txt)
c mod 1 chance (to c mod 5 chance) - The percent chance that the property will be assigned. If this equals 0, then the Property will always be assigned.
c mod 1 param (to c mod 5 param) - The “parameter” value associated with the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
c mod 1 min (to c mod 5 min) - The “min” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
c mod 1 max (to c mod 5 max) - The “max” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)

difficultylevels.txt

Overview

This file controls global parameters for game rules and how they work between each difficulty mode
Users cannot add new difficulty modes from this file

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - This is a reference field to define the difficulty mode

ResistPenalty - Defines the baseline starting point for a player character’s resistances for Expansion mode
ResistPenaltyNonExpansion - Defines the baseline starting point for a player character’s resistances for Non-Expansion mode
DeathExpPenalty - Modifies the percentage of current level experience lost when a player character dies

MonsterSkillBonus - Adds additional skill levels to skills used by monsters (defined from monstats.txt)
MonsterFreezeDivisor - Divisor that affects all Freeze Length values on monsters. The attempted Freeze Length value is divided by this divisor to determine the actual Freeze Length.
MonsterColdDivisor - Divisor that affects all Cold Length values on monsters. The attempted Cold Length value is divided by this divisor to determine the actual Cold Length.
AiCurseDivisor - Divisor that affects all durations of Curses on monsters. The attempted Curse duration is divided by this divisor to determine the actual Curse duration.

LifeStealDivisor - Divisor that affects the amount of Life Steal that player characters gain. The attempted Life Steal value is divided by this divisor to determine the actual Life Steal.
ManaStealDivisor - Divisor that affects the amount of Mana Steal that player characters gain. The attempted Mana Steal value is divided by this divisor to determine the actual Mana Steal.

UniqueDamageBonus - Percentage modifier for a Unique monster's overall Damage and Attack Rating. This is applied after calculating the monster's other modifications.
ChampionDamageBonus - Percentage modifier for a Champion monster's overall Damage and Attack Rating. This is applied after calculating the monster's other modifications.

PlayerDamagePercentVSPlayer - Percentage modifier for the total damage a player deals to another player
PlayerDamagePercentVSMercenary - Percentage modifier for the total damage a player deals to another player's mercenary
PlayerDamagePercentVSPrimeEvil - Percentage modifier for the total damage a player deals to a Prime Evil boss

PlayerHitReactBufferVSPlayer – The frame length for the amount of time a player cannot be placed into another hit react from a player (25 frames = 1 second).
PlayerHitReactBufferVSMonster - The frame length for the amount of time a player cannot be placed into another hit react from a monster (25 frames = 1 second).

MercenaryDamagePercentVSPlayer - Percentage modifier for the total damage a player's mercenary deals to another player
MercenaryDamagePercentVSMercenary - Percentage modifier for the total damage a player's mercenary deals to another player's mercenary
MercenaryDamagePercentVSBoss - Percentage modifier for the total damage a player's mercenary deals to a boss monster
MercenaryMaxStunLength - The frame length for the maximum stun length allowed on a player's mercenary (25 Frames = 1 second)

PrimeEvilDamagePercentVSPlayer - Percentage modifier applied to the total damage a Prime Evil boss deals to a player
PrimeEvilDamagePercentVSMercenary - Percentage modifier for the total damage a Prime Evil boss deals to a player's mercenary
PrimeEvilDamagePercentVSPet - Percentage modifier for the total damage a Prime Evil boss deals to a player's pet

PetDamagePercentVSPlayer - Percentage modifier for the total damage a player's pet deals to another player

MonsterCEDamagePercent - Percentage modifier that affects how much damage is dealt to a player by a Monster's version of Corpse Explosion. For example, when certain monsters die and explode on death.
MonsterFireEnchantExplosionDamagePercent - Percentage modifier that affects how much damage is dealt to a player by a Monster's Fire Enchant explosion. The Fire Enchant death explosion uses the same Corpse Explosion functionality and this value is applied after the "MonsterCEDamagePercent" value.
StaticFieldMin - Percentage modifier for capping the amount of current Life damage dealt to monsters by the Sorceress Static Field skill. This field only affects games in Expansion mode.

GambleRare - The odds to obtain a Rare item from gambling. The game rolls a random number between 0 to 100000. If that rolled number is less than this value, then the gambled item will be a Rare item.
GambleSet - The odds to obtain a Set item from gambling. The game rolls a random number between 0 to 100000. If that rolled number is less than this value, then the gambled item will be a Set item.
GambleUnique - The odds to obtain a Unique item from gambling. The game rolls a random number between 0 to 100000. If that rolled number is less than this value, then the gambled item will be a Unique item.
GambleUber - The odds to make the gambled item be an Exceptional Quality item. The game rolls a random number between 0 to 10000. This rolled number is then compared to the following formula:
([Item Level] - [Base Item Level]) * ["GambleUber"] + 1. If the rolled number is less than this value, then the item becomes an Exceptional Quality item, and the game will roll for upgrading it to Elite Quality (See "GambleUltra").
GambleUltra - The odds to make the gambled item be an Elite Quality item. The game rolls a random number between 0 to 10000. This rolled number is then compared to the following formula:
([Item Level] - [Base Item Level]) * ["GambleUltra"] + 1. If the rolled number is less than this value, then the item is upgraded to an Elite Quality item. This only happens if the item successfully rolled for Exceptional Quality.

experience.txt

Overview

This file controls the experience required for each level by each class

Data Fields

Level - This is a reference field to define the level

Amazon - Controls the experience required for each level with the Amazon class
Sorceress - Controls the experience required for each level with the Sorceress class
Necromancer - Controls the experience required for each level with the Necromancer class
Paladin - Controls the experience required for each level with the Paladin class
Barbarian - Controls the experience required for each level with the Barbarian class

Druid - Controls the experience required for each level with the Druid class
Assassin - Controls the experience required for each level with the Assassin class

ExpRatio - This multiplier affects the percentage of experienced earned, based on the level (Calculated in 1024ths by default, but this can be changed by updating the “10” value in the “Maxlvl” row)

gamble.txt

Overview

This file controls what Item Types will appear as possible items to purchase in the Gambling UI

Item Types can be added to this list as a potential option for the Gambling system

Data Fields

name - This is a reference field to describe the Item
code - This is a pointer to “code” field from weapons.txt/armor.txt/misc.txt

gems.txt

Overview

This file controls the item modifiers of gems and runes for each item type

This file is used by the following files: weapons.txt, armor.txt, misc.txt

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

name - This is a reference field to define the gem/rune name
letter - Defines the string that is concatenated together in the item tooltip when a rune is socketed into an item
transform - Controls the color change of the item after being socketed by the gem/rune. (Uses Color Codes from the reference file colors.txt)

Code	Color
0	White
1	Light Grey
2	Dark Grey
3	Black
4	Light Blue
5	Dark Blue
6	Crystal Blue
7	Light Red
8	Dark Red
9	Crystal Red
10	Light Green
11	Dark Green
12	Crystal Green
13	Light Yellow
14	Dark Yellow
15	Light Gold
16	Dark Gold
17	Light Purple
18	Dark Purple
19	Orange
20	Bright White

code - Defines the unique item code used to create the gem/rune

weaponMod1Code (to weaponMod3Code) - Controls the item properties that the gem/rune provides when socketed into an item with a “gemapplytype” value that equals 0 (Uses the “code” field from Properties.txt)
weaponMod1Param (to weaponMod3Param) - The stat’s “parameter” value associated with the listed property (weaponMod1Code). Usage depends on the property function (See the “func” field on Properties.txt)
weaponMod1Min (to weaponMod3Min) - The stat’s “min” value associated with the listed property (weaponMod1Code). Usage depends on the property function (See the “func” field on Properties.txt)
weaponMod1Max (to weaponMod3 Max) - The stat’s “max” value to assign to the listed property (weaponMod1Code). Usage depends on the property function (See the “func” field on Properties.txt)

helmMod1Code (to helmMod3Code) - Controls the item properties that the gem/rune provides when socketed into an item with a “gemapplytype” value that equals 1 (Uses the “code” field from Properties.txt)

helmMod1Param (to helmMod3Param) - The stat's "parameter" value associated with the listed property (helmMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)
helmMod1Min (to helmMod3Min) - The stat's "min" value associated with the listed property (helmMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)
helmMod1Max (to helmMod3Max) - The stat's "max" value to assign to the listed property (helmMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)

shieldMod1Code (to shieldMod3Code) - Controls the item properties that the gem/rune provides when socketed into an item with a "gemapplytype" value that equals 2 (Uses the "code" field from Properties.txt)
shieldMod1Param (to shieldMod3Param) - The stat's "parameter" value associated with the listed property (shieldMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)
shieldMod1Min (to shieldMod3Min) - The stat's "min" value associated with the listed property (shieldMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)
shieldMod1Max (to shieldMod3Max) - The stat's "max" value to assign to the listed property (shieldMod1Code). Usage depends on the property function (See the "func" field on Properties.txt)

inventory.txt

Overview

This file controls the grid sizes of the inventory slots for the game's various UI windows.

These values are measured in pixels on the screen.

Data Fields

class - This is a reference field to define the type of inventory screen

invLeft - Starting X coordinate pixel position of the inventory panel
invRight - Ending X coordinate pixel position of the inventory panel (Includes the "invLeft" value with the inventory width size)
invTop - Starting Y coordinate pixel position of the inventory panel
invBottom - Ending Y coordinate pixel position of the inventory panel (Includes the "invTop" value with the inventory height size)

gridX - Column number size of the inventory grid, measured in the number of grid boxes to use
gridY - Column row size of the inventory grid, measured in the number of grid boxes to use
gridLeft - Starting X coordinate location of the inventory's left grid side
gridRight - Ending X coordinate location of the inventory's right grid side (Includes the "gridLeft" value with the grid width size)
gridTop - Starting Y coordinate location of the inventory's top grid side
gridBottom - Ending Y coordinate location of the inventory's bottom grid side (Includes the "gridTop" value with the grid height size)
gridBoxWidth - Width size of an inventory's box cell
gridBoxHeight - Height size of an inventory's box cell

rArmLeft - Starting X coordinate location of the Right Weapon Slot
rArmRight - Ending X coordinate location of the Right Weapon Slot (Includes the "rArmLeft" value with the "rArmWidth" value)
rArmTop - Starting Y coordinate location of the Right Weapon Slot
rArmBottom - Ending Y coordinate location of the Right Weapon Slot (Includes the "rArmTop" value with the "rArmHeight" value)
rArmWidth - The pixel width of the Right Weapon Slot
rArmHeight - The pixel Height of the Right Weapon Slot

torsoLeft - Starting X coordinate location of the Body Armor Slot
torsoRight - Ending X coordinate location of the Body Armor Slot (Includes the "torsoLeft" value with the "torsoWidth" value)
torsoTop - Starting Y coordinate location of the Body Armor Slot
torsoBottom - Ending Y coordinate location of the Body Armor Slot (Includes the "torsoTop" value with the "torsoHeight" value)
torsoWidth - The pixel width of the Body Armor Slot
torsoHeight - The pixel Height of the Body Armor Slot

lArmLeft - Starting X coordinate location of the Left Weapon Slot
lArmRight - Ending X coordinate location of the Left Weapon Slot (Includes the "lArmLeft" value with the "lArmWidth" value)
lArmTop - Starting Y coordinate location of the Left Weapon Slot
lArmBottom - Ending Y coordinate location of the Left Weapon Slot (Includes the "lArmTop" value with the "lArmHeight" value)
lArmWidth - The pixel width of the Left Weapon Slot
lArmHeight - The pixel Height of the Left Weapon Slot

headLeft - Starting X coordinate location of the Helm Slot
headRight - Ending X coordinate location of the Helm Slot (Includes the "headLeft" value with the "headWidth" value)
headTop - Starting Y coordinate location of the Helm Slot
headBottom - Ending Y coordinate location of the Helm Slot (Includes the "headTop" value with the "headHeight" value)
headWidth - The pixel width of the Helm Slot
headHeight - The pixel Height of the Helm Slot

neckLeft - Starting X coordinate location of the Amulet Slot
neckRight - Ending X coordinate location of the Amulet Slot (Includes the "neckLeft" value with the "neckWidth" value)
neckTop - Starting Y coordinate location of the Amulet Slot
neckBottom - Ending Y coordinate location of the Amulet Slot (Includes the "neckTop" value with the "neckHeight" value)

neckWidth - The pixel width of the Amulet Slot
neckHeight - The pixel Height of the Amulet Slot

rHandLeft - Starting X coordinate location of the Right Ring Slot
rHandLeft - Ending X coordinate location of the Right Ring Slot (Includes the “rHandLeft” value with the “rHandWidth” value)
rHandTop - Starting Y coordinate location of the Right Ring Slot
rHandBottom - Ending Y coordinate location of the Right Ring Slot (Includes the “rHandTop” value with the “rHandHeight” value)
rHandWidth - The pixel width of the Right Ring Slot
rHandHeight - The pixel Height of the Right Ring Slot

lHandLeft - Starting X coordinate location of the Left Ring Slot
lHandRight - Ending X coordinate location of the Left Ring Slot (Includes the “lHandLeft” value with the “lHandWidth” value)
lHandTop - Starting Y coordinate location of the Left Ring Slot
lHandBottom - Ending Y coordinate location of the Left Ring Slot (Includes the “lHandTop” value with the “lHandHeight” value)
lHandWidth - The pixel width of the Left Ring Slot
lHandHeight - The pixel Height of the Left Ring Slot

beltLeft - Starting X coordinate location of the Belt Slot
beltRight - Ending X coordinate location of the Belt Slot (Includes the “beltLeft” value with the “beltWidth” value)
beltTop - Starting Y coordinate location of the Belt Slot
beltBottom - Ending Y coordinate location of the Belt Slot (Includes the “beltTop” value with the “beltHeight” value)
beltWidth - The pixel width of the Belt Slot
beltHeight - The pixel Height of the Belt Slot

feetLeft - Starting X coordinate location of the Boots Slot
feetRight - Ending X coordinate location of the Boots Slot (Includes the “feetLeft” value with the “feetWidth” value)
feetTop - Starting Y coordinate location of the Boots Slot
feetBottom - Ending Y coordinate location of the Boots Slot (Includes the “feetTop” value with the “feetHeight” value)
feetWidth - The pixel width of the Boots Slot
feetHeight - The pixel Height of the Boots Slot

glovesLeft - Starting X coordinate location of the Gloves Slot
glovesRight - Ending X coordinate location of the Gloves Slot (Includes the “glovesLeft” value with the “glovesWidth” value)
glovesTop - Starting Y coordinate location of the Gloves Slot
glovesBottom - Ending Y coordinate location of the Gloves Slot (Includes the “glovesTop” value with the “glovesHeight” value)
glovesWidth - The pixel width of the Gloves Slot
glovesHeight - The pixel Height of the Gloves Slot

itemratio.txt

Overview

This file determines the quality of items when being spawned. After the game determines what Item Type should spawn, it then uses this file to calculate the quality of that item.

These Item Quality checks are used for most item drops in the game such as monster drops and chest drops.

The following files related to these calculations: ItemTypes.txt, weapons.txt, armor.txt, misc.txt, Uniqueitems.txt, SetItems.txt, monstats.txt, TreasureClassEx.txt

Data Fields

Function - This is a reference field to define the item ratio name
Version - Defines which game version to use this item ratio (0 = Classic mode | 100 = Expansion mode)

Uber - Boolean Field. If equals 1, then the item ratio will apply to items with Exceptional or Elite Quality. If equals 0, then the item ratio will apply to Normal Quality items (This is determined by the “normcode”, “ubercode” and “ultracode” fields in weapons.txt / armor.txt)
Class Specific - Boolean Field. If equals 1, then the item ratio will apply to class-based items (This will compare to the Item Type’s “Class” field to determine if the item is class specific)

Unique - Base value for calculating the Unique Quality chance. Higher value means rarer chance. (Calculated first)
UniqueDivisor - Modifier for changing the Unique Quality chance, based on the difference between the Monster Level and the Item’s base level
UniqueMin - The minimum value of the probability denominator for Unique Quality. This is compared to the calculated Unique Quality value after Magic Find calculations and is chosen if it is greater than that value. (Calculated in 128ths)

Set - Base value for calculating the Set Quality chance. Higher value means rarer chance. (Calculated after Unique)
SetDivisor - Modifier for changing the Set Quality chance, based on the difference between the Monster Level and the Item’s base level
SetMin - The minimum value of the probability denominator for Set Quality. This is compared to the calculated Set Quality value after Magic Find calculations and is chosen if it is greater than that value. (Calculated in 128ths)

Rare - Base value for calculating the Rare Quality chance. Higher value means rarer chance. (Calculated after Set)
RareDivisor - Modifier for changing the Rare Quality chance, based on the difference between the Monster Level and the Item’s base level
RareMin - The minimum value of the probability denominator for Rare Quality. This is compared to the calculated Rare Quality value after Magic Find calculations and is chosen if it is greater than that value. (Calculated in 128ths)

Magic - Base value for calculating the Magic Quality chance. Higher value means rarer chance. (Calculated after Rare)
MagicDivisor - Modifier for changing the Magic Quality chance, based on the difference between the Monster Level and the Item's base level
MagicMin - The minimum value of the probability denominator for Magic Quality. This is compared to the calculated Magic Quality value after Magic Find calculations and is chosen if it is greater than that value. (Calculated in 128ths)

HiQuality - Base value for calculating the High Quality (Superior) chance. Higher value means rarer chance. (Calculated after Magic)
HiQualityDivisor - Modifier for changing the High Quality (Superior) chance, based on the difference between the Monster Level and the Item's base level

Normal - Base value for calculating the Normal Quality chance. Higher value means rarer chance. (Calculated after Normal, and if this does not succeed in rolling, then the item is defaulted to Low Quality)
NormalDivisor - Modifier for changing the Normal Quality chance, based on the difference between the Monster Level and the Item's base level

Calculations

Item Quality is determined by first calculating the roll chance for a specific Quality. Then the game will attempt a random roll for that Item Quality. If that roll fails, then the game will calculate for the next lower Item Quality.
The order of Item Quality calculations is the following: Unique > Set > Rare > Magic > High Quality (Superior) > Normal > Low Quality

The following information details how the game uses the Data Fields in its calculations.

Step 1: Calculate the Base Probability

The first part of the calculations is to obtain the base probability for rolling the item quality with the following formula:
Probability = (Quality - (mlvl - ilvl) / Divisor) * 128

This probability value is a ratio divisor, meaning that there is a 1 in probability chance of the game choosing that Item Quality, so the lower the probability value, then the better the chance the item will successfully roll that Item Quality. The multiplication with 128 is for decreasing rounding errors. The **Quality** value is the "Unique"/"Set"/"Rare"/"Magic"/"HiQuality"/"Normal" Data Field. The **mlvl** value is the Monster Level, which depends on the current area level and game difficulty (this can also be the level for the chest drop). The **ilvl** value is the Base Item Level (See the "level" field in the weapons.txt/armor.txt/misc.txt or the "lvl" field in UniqueItems.txt/SetItems.txt). The **Divisor** value is the "UniqueDivisor"/"SetDivisor"/"RareDivisor"/"MagicDivisor"/"HiQualityDivisor"/"NormalDivisor" Data Field.

Step 2: Calculate Magic Find

The game will then obtain the character's magic find bonus from items. If the current calculation is for Unique/Set/ Rare Quality items and the magic find item bonus exceeds 110%, then the character's magic find will be modified with diminishing returns:
MagicFind = 100 + MF * dim / (MF + dim)

The **MF** value is the character's magic find bonus percentage value plus the baseline default 100 value (See "item_magicbonus" in ItemStatCost.txt). The **dim** value is a special modifier for adding diminishing returns to the magic find bonus, which differs based on the Item Quality being calculated (Unique = 250, Set = 500, Rare = 600)

After calculating the proper magic find value, the probability value is modified with the following formula:
Probability = Probability * 100 / MagicFind

This will reduce the Probability value, giving the Item Quality a higher chance to be successfully rolled.

Step 3: Calculate Probability with Treasure Class

After calculating the baseline probability with the magic find bonus, the game will then compare this value with the minimum value for the Item Quality to cap it from reducing any further (See "UniqueMin"/"SetMin"/"RareMin"/"MagicMin" Data Fields). High Quality (Superior) and Normal Quality do not have a minimum value.

The game will then modify the probability with the value from the related Treasure Class:
Probability = Probability - Probability * TreasureClass / 1024)

The **TreasureClass** value is a modifier for this Item Quality based on the Treasure Class being used (See the "Unique"/"Set"/"Rare"/"Magic" field from the TreasureClassEx.txt file)

Step 4: Roll for the Item Quality

Finally, after calculating the overall value of the probability for the Item Quality, the game will then find a random number between 0 and the probability value. If that random value is between 0 and 128, then the item has successfully rolled that specific Item Quality. Otherwise, the calculations will move on to checking for the next lower Item Quality.

ItemStatCost.txt

Overview

This file controls the functionalities for each possible stat on a unit

These defined stats are used to form modifiers for the Properties.txt file

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

Stat - Defines the unique pointer for this stat, which is used in other files

Send Other - Boolean Field. If equals 1, then only add the stat to a new monster if the that has no state and has an item mask. If equals 0, then ignore this.
Signed - Boolean Field. If equals 1, then the stat will be treated as a signed integer, meaning that it can be either a positive or negative value. If equals 0, then stat will be treated as an unsigned integer, meaning that it can only be a positive value. This only affects stats with state bits.
Send Bits - Controls how many bits of data for the stat to send to the game client, essentially controlling the max value possible for the stat. Signed values should have less than 32 bits, otherwise they will be treated as unsigned values.
Send Param Bits - Controls how many bits of data for the stat's parameter value to send to the client for a unit. This value is always treated as a signed integer.

UpdateAnimRate - Boolean Field. If equals 1, then the stat will notify that game to handle and adjust the speed of the unit when the stat changes. If equals 0, then ignore this. This is only checked for stats with states or for specific skill server functions including 30, 61, 71.

Saved - Boolean Field. If equals 1, then this state will be inserted in the change list to be stored in the Character Save file. If equals 0, then ignore this.
CSvSigned - Boolean Field. If equals 1, then the stat will be saved as a signed integer in the Character Save file. If equals 0, then the stat will be saved as an unsigned integer in the Character Save file. This is only used if the "Saved" field is enabled.
CSvBits - Controls how many bits of data for the stat to send to save in the Character Save file. Signed values should have less than 32 bits, otherwise they will be treated as unsigned values. This is only used if the "Saved" field is enabled.
CSvParam - Controls how many bits of data for the stat's parameter value to save in the Character Save file. This value is always treated as a signed integer. This is only used if the "Saved" field is enabled.

fCallback - Boolean Field. If equals 1, then any changes to the stat will call the Callback function which will update thgecharacter's states, skills, or item events based on the changed stat value. If equals 0, then ignore this.

fMin - Boolean Field. If equals 1, then the stat will have a minimum value that cannot be reduced further than that value (See "MinAccr" field). If equals 0, then ignore this.
MinAccr - The minimum value of a stat. This is only used if the "fMin" field is enabled.

Encode - Controls how the stat will modify an item's buy, sell, and repair costs. This field uses a code value to select a function to handle the calculations. This field relies on the "Add", "Multiply" and "ValShift" fields. The baseline Stat Value is first modified using the "ValShift" field to shift the bits. This Stat Value is then used in the calculations by one of the selected functions.

Code	Parameters	Description
0 (or empty)	Add Multiply	Buy Cost += [Stat Value] * ["Multiply"] / 1024 + ["Add"] Sell Cost += [Stat Value] * ["Multiply"] / 1024 + ["Add"] Repair Cost += [Stat Value] * ["Multiply"] / 1024 + ["Add"]
1		Use the stat's parameter value to determine the skill ID used. Use the stat's value to determine the skill level. Obtain the "cost mult" and "cost add" values from the skill linked in this stat's parameter (see skills.txt). The Stat Value is considered the skill's level. Buy Cost += [Stat Value] * ["cost mult"] / 1024 + ["cost add"] Sell Cost += [Stat Value] * ["cost mult"] / 4096 + ["cost add"] Repair Cost += [Stat Value] * ["cost mult"] / 1024 + ["cost add"]
2		Use the stat's parameter value to determine both the skill ID and skill level. Obtain the "cost mult" and "cost add" values from the determined skill ID (see skills.txt). Buy Cost += [Stat Value] * ["cost mult"] / 1024 + ["cost add"] Sell Cost += [Stat Value] * ["cost mult"] / 4096 + ["cost add"] Repair Cost += [Stat Value] * ["cost mult"] / 1024 + ["cost add"]
3		Same as function 2
4	Add Multiply	Obtains the stat's min and max values based on the By Time bit masks and uses them to calculate the average value or Baseline stat value. Buy Cost += [Baseline] * ["Multiply"] / 1024 + ["Add"] Sell Cost += [Baseline] * ["Multiply"] / 1024 + ["Add"] Repair Cost += [Baseline] * ["Multiply"] / 1024 + ["Add"]

Add - Used as a possible parameter value for the "Encode" function. Flat integer modification to the Unique item's buy, sell, and repair costs. This is added after the "Multiply" field has modified the costs.
Multiply - Used as a possible parameter value for the "Encode" function. Multiplicative modifier for the item's buy, sell, and repair costs. The way this value is used depends on the Encode function selected.

ValShift - Used to shift the stat's input value by a number of bits to obtain the actual value when performing calculations (such as for the "Encode" function).

1.09-Save Bits - Controls how many bits of data are allocated for the overall size of the stat when saving/reading an item from a Character Save. This value can be treated as a signed or unsigned integer, depending on the stat. This field is only used for items saved in a game version of Patch 1.09d or older.
1.09-Save Add - Controls how many bits of data are allocated for the stat's value when saving/reading an item from a Character Save. This value is treated as a signed integer. This field is only used for items saved in a game version of Patch 1.09d or older.
Save Bits - Controls how many bits of data are allocated for the overall size of the stat when saving/reading an item from a Character Save. This value can be treated as a signed or unsigned integer, depending on the stat.
Save Add - Controls how many bits of data are allocated for the stat's value when saving/reading an item from a Character Save. This value is treated as a signed integer.
Save Param Bits - Controls how many bits of data for the stat's parameter value to use when saving/reading an item from a Character Save. This value is always treated as an unsigned integer.

keepzero - Boolean Field. If equals 1, then this stat will remain on the stat change list, when being updated, even if that stat value is 0. If equals 0, then ignore this.

op - This is the stat operator, used for advanced stat modification when calculating the value of a stat. This can involves using this stat and its value to modify another stat's value. This use a function ID to determine what to calculate.

Code	Parameters	Description
0 (or empty)		No Operator. Just add the stat normally
1	op stat1 op stat2 op stat3	Percent Operator. Gets the value of "op stat#" and multiplies it by a percentage increase equal to this stat's value: ["op stat#"] += ["op stat#"] * value / 100
2	op param op base op stat1 op stat2 op stat3	By Level Operator. Gets value of "op stat#" and uses it as a multiplier with "op param" as the divisor: ["op stat#"] += ["op stat#"] * ["op base"] << ["op param"]
3	op param op base op stat1 op stat2 op stat3	By Level Percent Operator. Gets value of "op stat#" and uses it as a multiplier with "op param" as the divisor. Then it uses this value as a percentage increase to "op stat#": percent = ["op stat#"] * ["op base"] << ["op param"] ["op stat#"] = ["op stat#"] * percent / 100
4	op param op base op stat1 op stat2 op stat3	By Level Source Operator. Gets value of "op stat#" for the item (not the unit) and uses it as a multiplier with "op param" as the divisor: ["op stat#"] += ["op stat#"] * ["op base"] << ["op param"]
5	op param op base op stat1 op stat2 op stat3	By Level Source Percent Operator. Gets value of "op stat#" for the item (not the unit) and uses it as a multiplier with "op param" as the divisor. Then it uses this value as a percentage increase to "op stat#": percent = ["op stat#"] * ["op base"] << ["op param"] ["op stat#"] = ["op stat#"] * percent / 100
6	op stat1 op stat2 op stat3	By Time Operator. Gets the value of "op stat#" and increases it by a delta value which depends on game's time of day. The delta is calculated by using the stat's min and max as a range of increase/decrease and biasing this value with the current progress of game's time of day ["op stat#"] += ["op stat#"] * [delta]
7	op stat1 op stat2 op stat3	By Time Percent Operator. Gets the value of "op stat#" and multiplies it by a percentage. This percentage is determined by obtaining "op stat#" and a delta value which depends on game's time of day. The delta is calculated by using the stat's min and max as a range of increase/decrease and biasing this value with the current progress of game's time of day percent = ["op stat#"] * [delta] ["op stat#"] = ["op stat#"] * percent / 100
8	op stat1 op stat2 op stat3	Energy Operator. This will only apply for stats on the player. Gets the value of "op stat#" and multiplies it by the related "ManaPerMagic" field from the charstats.txt file. This is then bit shifted by the baseline Mana bit value, MANA_SHIFT = 8, with the fourths value calculation from the "ManaPerMagic" field. ["op stat#"] = ["op stat#"] * ["ManaPerMagic"] << (MANA_SHIFT - 2)
9	op stat1 op stat2 op stat3	Vitality Operator. This will only apply for stats on the player. If the stat is "maxstamina", then the operator will get the value of "op stat#" and multiply it by the related "StaminaPerVitality" field from the charstats.txt file. This is then bit shifted by the baseline Stamina bit value, STAMINA_SHIFT = 8, with the fourths value calculation from the "StaminaPerVitality" field: ["op stat#"] = ["op stat#"] * ["StaminaPerVitality"] << (STAMINA_SHIFT - 2) If the stat is not "maxstamina", then the operator will get the value of "op stat#" and multiply it by the related "LifePerVitality" field from the charstats.txt file. This is then bit shifted by the baseline Life bit value, LIFE_SHIFT = 8, with the fourths value calculation from the "LifePerVitality" field: ["op stat#"] = ["op stat#"] * ["LifePerVitality"] << (LIFE_SHIFT - 2)
10		Currently not being used. Does nothing.
11	op stat1 op stat2 op stat3	Player Percent Operator. This will only apply for stats on units. Gets the value of "op stat#" and multiplies it by a percentage increase equal to this stat's value: ["op stat#"] += ["op stat#"] * value / 100
12		Currently not being used. Does nothing.
13	op stat1 op stat2 op stat3	Item Percent Operator. This will only apply for stats on items. Gets the value of "op stat#" and multiplies it by a percentage increase equal to this stat's value: ["op stat#"] += ["op stat#"] * value / 100

op param - Used as a possible parameter value for the "op" function.
op base - Used as a possible parameter value for the "op" function.
op stat1 (to opstat3) - Used as a possible parameter value for the "op" function.

direct - Boolean Field. If equals 1, then when the stat is being updated in certain skill functions having to do with state changes, the stat will update in relation to its "maxstat" field to ensure that it never exceeds that value. If equals 0, then ignore this, and the stat will simply update in these cases. This only applies to skills that use skill server function 65, 66, 81, and 82.
maxstat - Controls which stat is associated with this stat to be treated as the maximum version of this stat. This means that 2 stats are essentially linked so that there can be a current version of the stat and a maximum version to control the cap of stat's value. This is used for Life, Mana, Stamina, and Durability. This field relies on the "direct" field to be enabled unless it is being used for the healing potion item spell.
damagerelated - Boolean Field. If equals 1, then this stat will be exclusive to the item and will not add to the unit. If equals 0, then ignore this, and the stat will always add to the unit. This is typically used for weapons and is important when dual wielding weapons so that when a unit attacks, then one weapon's stats do not stack with another weapon's stats.

itemevent1 & itemevent2 - Uses an event that will activate the specified function defined by "itemeventfunc#". This points to the ID of an event defined in the events.txt file.

event	Description
(empty)	Do nothing.
hitbymissile	Unit is hit by a missile
damagedinmelee	Unit takes damage from a melee attack
damagedbymissile	Unit takes damage from a missile
attackedinmelee	Unit is attacked by a melee attack
doactive	Unit used a skill
domeleedamage	Unit dealt damage with a melee attack
domissiledamage	Unit dealt damage with a missile
domeleeattack	Unit used a melee attack
domissileattack	Unit used a missile attack
kill	Unit killed another Unit
killed	Unit dies
absorbdamage	Unit takes damage
levelup	Unit gained a Level
death	Monster dies

itemeventfunc1 & itemeventfunc2 - Specifies the function to use after the related item event occurred. Functions are defined by a numeric ID code. This is only applied based on the “itemevent#” field definition.

Code	Description
0 (or empty)	Do nothing.
1	Applies the effects of the Sorceress Chilling Armor skill
2	Applies the effects of the Sorceress Frozen Armor skill
3	Applies the effects of the Sorceress Shiver Armor skill
4	Applies the effects of the Necromancer Iron Maiden skill, causing damage taken to be dealt to the attacker
5	Applies the effects of the Necromancer Life Tap skill on monsters. Usable only on monsters when cast by players.
6	Attacker Takes Physical Damage
7	Applies knockback on the target, moving the target backwards from the attacker and being briefly stunned during this time. Chances depend on the “small” and “large” flags from the monstats2.txt file.
8	Applies the effects Barbarian Howl skill on monsters, causing them to run away in fear. Does not work on Champion or Unique monsters.
9	Applies the effects of the Necromancer Dim Vision skill to a target. Effectiveness is reduced from missile attacks.
10	Attacker Takes Lightning Damage
11	Attacker Takes Fire Damage
12	Attacker Takes Cold Damage
13	A percentage of damage taken will also reduce the user’s mana by that amount
14	Applies the Freeze effect on the target. Effectiveness is reduced from missile attacks.
15	Applies Open Wounds damage on the target, which causes life damage over time
16	Applies Crushing Blow damage on the target. Damage depends on if the target is a Player, Mercenary, Boss monster, Unique monster, Champion monster, or normal monster.
17	Restores mana to the user that performed the kill
18	Restores life to the user that performed the kill on a Demon monster
19	• Applies the slow state on the target which reduces that target’s attack speed and movement speed • If the target is a player, Champion monster, Unique Monster, Boss, or mercenary, then the max slow value is 50 • If the target is a Super monster, then the max slow value is 75 • Otherwise, the max slow value is 90.
20	Use a skill against the target after the user is attacks or hits the enemy
21	Use a skill against the attacker after the user is hit by an attack. If there is no attacker, then the skill is cast at the user’s location
22	Applies the effects of the Necromancer Bone Armor skill, absorbing physical damage taken
23	• Transfers damage dealt by a pet as healing that is split between the pet and its owner (Used by the Necromancer Blood Golem pet) • Uses the linked skill’s “calc2” field from the skills.txt file to determine the total healing percentage • Uses the linked skill’s “calc3” field from the skills.txt file to determine the healing percentage that is split to the pet’s owner
24	• Absorb a percentage any damage taken and deals that damage to Mana instead of Life (Used by the Sorceress Energy Shield skill) • Uses the linked skill’s “calc1” and “calc2” fields from the skills.txt file to determine the mana to Damage ratio conversion
25	Apply the effects of the Druid Cyclone armor skill, absorbing Fire, Cold, and Lightning damage taken
26	• Transfers damage taken from the pet to its owner (Used by the Necromancer Blood Golem pet) • The percentage of damage transferred is defined in the linked skill’s “Param5” field from the skills.txt file
27	• Applies the “item_slow” stat which reduces the target’s attack speed and movement speed (Used by the Necromancer Clay Golem pet) • If the target is a Champion or Unique monster, then the max slow value is 50. Otherwise, the max slow value is 90.
28	Restores life to the user that performed a kill
29	Applies the “restinpeace” state which essentially the corpse of a killed monster, making it unusable
30	Cast a skill when the item event occurs, either with without a target
31	Reanimates the targeted enemy as a pet monster for the user. Only applies on units classified as monsters, and not Champions or Uniques.
32	Use a skill to deal area radius damage around the user
33	Use a skill’s linked sub-skill from the “sumskill1” in the skills.txt file

descpriority - Controls how this stat is sorted in item tooltips. This field is compared to the same field on other stats to determine how to order the stats. The higher the value means that the stat will be sorted higher than other stats. If more than 1 stat has the same “descpriority” value, then they will be listed in the order defined in this data file.

descfunc - Controls how the stat is displayed in tooltips. Uses an ID value to select a description function to format the string value.

Code	Parameters	Description
0		No display. Do nothing.
1	descval descstrpos descstrneg	Plus or Minus • If value > 0, “+[value] [descstr]” • If value < 0, “-[value] [descstr]”
2	descval descstrpos	Percent • “[value] [descstr]”

	descstrneg	
3	descval descstrpos descstrneg	String • “[value] [descstr]”
4	descval descstrpos descstrneg	Plus Percent • “+[value]% [descstr]”
5	descval descstrpos descstrneg	Percent 128 • “+[value * 100 / 128]% [descstr]”
6	descval descstrpos descstrneg descstr2	Plus or Minus Per Level • If value > 0, “+[value] [descstr] [descstr2]” • If value < 0, “-[value] [descstr] [descstr2]”
7	descval descstrpos descstrneg descstr2	Percent Per Level • “[value]% [descstr] [descstr2]”
8	descval descstrpos descstrneg descstr2	Plus Percent Per Level • “+[value]% [descstr] [descstr2]”
9	descval descstrpos descstrneg descstr2	String Per Level • “[value] [descstr] [descstr2]”
10	descval descstrpos descstrneg descstr2	Percent 128 Per Level • “[value * 100 / 128]% [descstr] [descstr2]”
11		Repair • Uses the string ModStre9t and inserts the value into this string
12	descval descstrpos descstrneg	Plus Sub One • If value > 1, then use “+[value] [descstr]” • Else, use “[value] [descstr]” or “-[value] [descstr]”
13		Add Class Skill • Uses the “StrAllSkills” from the charstats.txt file
14		Add Tab Skill • Uses “StrSkillTab#” from the charstats.txt file based on related tab being modified
15	descstrpos	Proc Skill • Gets the skill name, skill level, and chance percent to insert into the “descstrpos” string
16	descstrpos descstrneg	Aura • Gets the skill name, and uses the stat’s value for the skill level and inserts these values into the designated “descstr” string
17	descstrpos descstrneg	Plus Minus By Time • Gets the proper value based on the time of day and inserts this value into the “descstr” string • Uses the following strings for the second part of the description, depending on the time selected: ModStre9d, ModStre9e, ModStre9f, ModStre9g
18	descstrpos descstrneg	(Same as function 17)
19	descstrpos descstrneg descstr2	Sprintf Num • Uses the Sprintf string function with the designated “descstr” string and adds “descstr2” if that value is not empty
20	descval descstrpos descstrneg	Minus Percent • “[value * -1]% [descstr]”
21	descstrpos descstrneg descstr2	Minus Percent Per Level • “[value * -1]% [descstr] [descstr2]” • If “descstr2” is empty, then default to using the increaseswithplaylevelX string
22	descstrpos descstrneg	Versus Monster Percent • Uses “strplur” from the MonType.txt file based on the monster type selected, and inserts this value into the designated “descstr” string
23	descstrpos descstrneg	Reanimate • Obtains the related “NameStr” string from the monstats.txt file and inserts this string into the designated “descstr” string
24	descstrpos descstrneg	Charges • Obtains the skill, skill level, max charges, and current charges and inserts these values into the designated “descstr” string
25	descval descstrpos descstrneg	Minus • If desval equals 1, then use “+[value * -1] [descstr]” • If desval equals 2, then use “[descstr] +[value * -1]”
26	descval descstrpos descstrneg	Minus Per Level • (Same as function 25)

	descstr2	
27	descstrpos descstrneg	Single Skill • Obtains the “str name” field from skilldesc.txt file and the “StrClassOnly” field from the charstats.txt file • Uses the stat value as the skill level • Combines these values into the designated “descstr” string
28	descstrpos descstrneg	Non Class Skill • Obtains the “str name” field from skilldesc.txt file • Uses the stat value as the skill level • Combines these values into the designated “descstr” string
29	descstrpos descstrneg descstr2	Sprintf num positive • Same as function 19, except the it uses the absolute value of [value]

descval - Used as a possible parameter value for the “descfunc” function. This controls the how the value of the stat is displayed.

Code	Description
0	Do not show the value of the stat
1	Shows the value of the stat at the start of its description
2	Shows the value of the stat at the end of its description

descstrpos - Used as a possible parameter value for the “descfunc” function. This uses a string to display the item stat in a tooltip when its value is positive.
descstrneg - Used as a possible parameter value for the “descfunc” function. This uses a string to display the item stat in a tooltip when its value is negative.
descstr2 - Used as a possible parameter value for the “descfunc” function. This uses a string to append to an item stat’s string in a tooltip.

dgrp - Assigns the stat to a group ID value. If all stats with a matching “dgrp” value are applied on the unit, then instead of displaying each stat individually, the group description will be applied instead (see “dgrpfunc” field)
dgrpfunc - Controls how the shared group of stats is displayed in tooltips. Uses an ID value to select a description function to format the string value. This function IDs are exactly the same as the “descfunc” field, see that description for more details.
dgrpval - Used as a possible parameter value for the “dgrpfunc” function. This controls the how the value of the stat is displayed. (Functions the same as the “descval” field)
dgrpstrpos - Used as a possible parameter value for the “dgrpfunc” function. This uses a string to display the item stat in a tooltip when its value is positive.
dgrpstrneg - Used as a possible parameter value for the “dgrpfunc” function. This uses a string to display the item stat in a tooltip when its value is negative.
dgrpstr2 - Used as a possible parameter value for the “dgrpfunc” function. This uses a string to append to an item stat’s string in a tooltip.

stuff - Used as a bit shift value for handling the conversion of skill IDs and skill levels to bit values for the stat. Controls the numeric range of possible skill IDs and skill levels for charge based items. This value cannot be less than or equal to 0, or greater than 8, otherwise it will default to 6. The row that this value appears in the data file is unrelated, since this is a universally applied value.

advdisplay - Controls how the stat appears in the Advanced Stats UI

Code	Description
0 (or empty)	The stat will never appear on the Advanced Stats UI
1	The stat will always show on the Advanced Stats UI
2	The stat will only show on the Advanced Stats UI if the value is greater than 0

ItemTypes.txt

Overview

This file controls the general statistics for each type of item, which is then used for the item type fields in other files

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

ItemType - This is a reference field to define the Item Type name
Code - Defines the unique pointer for this Item Type, which is used by the following files: weapons.txt, armor.txt, misc.txt, cubemain.txt, skills.txt, treasureclassex.txt

Equiv1 & Equiv2 - Points to the index of another Item Type to reference as a parent. This is used to create a hierarchy for Item Types where the parents will have more universal settings shared across the related children
Repair - Boolean Field. If equals 1, then the item can be repaired by an NPC in the shop UI. If equals 0, then the item cannot be repaired.

Body - Boolean Field. If equals 1, then the item can be equipped by a character (also will require the “BodyLoc1” & “BodyLoc2” fields as parameters). If equals 0, then the item can only be carried in the inventory, stash, or Horadric Cube.
BodyLoc1 & BodyLoc2 - These are required parameters if the “Body” field is enabled. These fields specify the inventory slots where the item can be equipped.

Code	Description
(empty)	None
head	Head
neck	Neck
tors	Torso
rarm	Right Arm
larm	Left Arm

rrin	Right Ring
lrin	Left Ring
belt	Belt
feet	Feet
glov	Gloves

Shoots - Points to the index of another Item Type as the required equipped Item Type to be used as ammo
Quiver - Points to the index of another Item Type as the required equipped Item Type to be used as this ammo's weapon

Throwable - Boolean Field. If equals 1, then it determines that this item is a throwing weapon. If equals 0, then ignore this.
Reload - Boolean Field. If equals 1, then the item (considered ammo in this case) will be automatically transferred from the inventory to the required "BodyLoc" when another item runs out of that specific ammo. If equals 0, then ignore this.
ReEquip - Boolean Field. If equals 1, then the item in the inventory will replace a matching equipped item if that equipped item was destroyed. If equals 0, then ignore this.
AutoStack - Boolean Field. If equals 1, then if the player picks up a matching Item Type, then they will try to automatically stack together. If equals 0, then ignore this.

Magic - Boolean Field. If equals 1, then this item will always have the Magic quality (unless it is a Quest item). If equals 0, then ignore this.
Rare - Boolean Field. If equals 1, then this item can spawn as a Rare quality. If equals 0, then ignore this.
Normal - Boolean Field. If equals 1, then this item will always have the Normal quality. If equals 0, then ignore this.
Beltable - Boolean Field. If equals 1, then this item can be placed in the character's belt slots. If equals 0, then ignore this.

MaxSockets1 - Determines the maximum possible number of sockets that can be spawned on the item when the item level is greater than or equal to 1 and less than or equal to the "MaxSocketsLevelThreshold1" value. The number of sockets is capped by the "gemsockets" value from the weapons.txt/armor.txt/misc.txt file.
MaxSocketsLevelThreshold1 - Defines the item level threshold between using the "MaxSockets1" and "MaxSockets2" field
MaxSockets2 - Determines the maximum possible number of sockets that can be spawned on the item when the item level is greater than the "MaxSocketsLevelThreshold1" value and less than or equal to the "MaxSocketsLevelThreshold2". The number of sockets is capped by the "gemsockets" value from the weapons.txt/armor.txt/misc.txt file.
MaxSocketsLevelThreshold2 - Defines the item level threshold between using the "MaxSockets2" and "MaxSockets3" field
MaxSockets3 - Determines the maximum possible number of sockets that can be spawned on the item when the item level is greater than the "MaxSocketsLevelThreshold2" value. The number of sockets capped by the "gemsockets" value from the weapons.txt/armor.txt/misc.txt file.

TreasureClass - Boolean Field. If equals 1, then allow this Item Type to be used in default treasure classes. If equals 0, then ignore this.
Rarity - Determines the chance for the item to spawn with stats, when created as a random Weapon/Armor/Misc item. Used in the following formula: IF RANDOM(0, (["Rarity"] - [Current Act Level])) > 0, THEN spawn stats
StaffMods - Determines if the Item Type should have class specific item skill modifiers

Code	Description
(empty)	No preference
ama	Amazon skills
bar	Barbarian skills
pal	Paladin skills
nec	Necromancer skills
sor	Sorceress skills
dru	Druid skills
ass	Assassin skills

Class - Determines if this item should be useable only by a specific class

Code	Description
(empty)	Any Class
ama	Amazon only
bar	Barbarian only
pal	Paladin only
nec	Necromancer only
sor	Sorceress only
dru	Druid only
ass	Assassin only

VarInvGfx - Tracks the number of inventory graphics used for this item type. This number much match the number of "InvGfx" fields used.
InvGfx1 (to InvGfx6) - Defines a DC6 file to use for the item's inventory graphics. The amount of this fields used should match the value used in "VarInvGfx"

StorePage - Uses a code to determine which UI tab page on the NPC shop UI to display this Item Type, such as after it is sold to the NPC.

Code	Description
armo	Armor Page
weap	Weapons Page
mag	Magic Page
misc	Misc Page

dualwieldclass1 (to dualwieldclass7) - Determines if the weapon can be dual wielded by a defined class. There are 7 fields to potentially allow all 7 classes.

Code	Description
ama	Amazon
bar	Barbarian
pal	Paladin
nec	Necromancer
sor	Sorceress
dru	Druid
ass	Assassin

hireling.txt

Overview

This file controls the unit statistics for player mercenaries and their related functions

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Hireling - This is a reference field to define the Hireling name
Version - Defines which game version to use this hireling (0 = Classic mode | 100 = Expansion mode)
Id - The unique identification number to define each hireling type
Class - This refers to the “hcIdx” field in MonStats.txt, which defines the base type of unit to use for the hireling

Act - The Act that the hireling belongs to (values 1 to 5 equal Act 1 to Act 5, respectively)
Difficulty - The difficulty mode associated with the hireling (1 = Normal | 2 = Nightmare | 3 = Hell)
Level - The starting level of the unit

Seller - This refers to the “hcIdx” field in MonStats.txt, which defines the unit NPC that sells this hireling
NameFirst & NameLast - These fields define a string key which the game uses as a sequential range of string IDs from “NameFirst” to “NameLast” to randomly generate as hireling names. (Max name length is 48 characters)

Gold - The initial cost of the hireling. This is used in the following calculation to generate the full hire price: Cost = [“Gold”] * (100 + 15 * [Difference of Current Level and “Level”]) / 100
Exp/Lvl - This modifier is used in the following calculation to determine the amount of Experience need for the hireling’s next level: [Current Level] + [Current Level] * [Current Level + 1] * [“Exp/Lvl”]
HP - The starting amount of Life at base Level
HP/Lvl - The amount of Life gained per Level
Defense - The starting amount of Defense at base Level
Def/Lvl - The amount of Defense gained per Level
Str - The starting amount of Strength at base Level
Str/Lvl - The amount of Strength gained per Level (Calculated in 8ths)
Dex - The starting amount of Dexterity at base Level
Dex/Lvl - The amount of Dexterity gained per Level (Calculated in 8ths)
AR - The starting amount of Attack Rating at base Level
AR/Lvl - The amount of Attack Rating gained per Level
Dmg-Min - The starting amount of minimum Physical Damage for attacks
Dmg-Max - The starting amount of maximum Physical Damage for attacks
Dmg/Lvl - The amount of Physical Damage gained per level, to be added to “Dmg-Min” and “Dmg-Max” (Calculated in 8ths)
ResistFire - The starting amount of Fire Resistance at base Level
ResistFire/Lvl - The amount of Fire Resistance gained per Level (Calculated in 4ths)
ResistCold - The starting amount of Fire Resistance at base Level
ResistCold/Lvl - The amount of Fire Resistance gained per Level (Calculated in 4ths)
ResistLightning - The starting amount of Fire Resistance at base Level
ResistLightning/Lvl - The amount of Fire Resistance gained per Level (Calculated in 4ths)
ResistPoison - The starting amount of Fire Resistance at base Level
ResistPoison/Lvl - The amount of Fire Resistance gained per Level (Calculated in 4ths)

HireDesc - This accepts a string key, which is used to display as the special description of the hireling in the hire UI window

DefaultChance - This is the chance for the hireling to attack with his/her weapon instead of using a Skill. All Chance values are summed together as a denominator value for a random roll to determine which skill to use.
Skill1 (to Skill6) - Points to a skill from the “skill” field in the skills.txt file. This gives the hireling the Skill to use (requires “Mode#”, “Chance#”, “ChancePerLvl#”)
Mode1 (to Mode6) - Uses a monster mode to determine the hireling’s behavior when using the related Skill (Uses the numeric ID of the monster mode, not the Token)

ID	Token	Description
0	DT	Death / Reset
1	NU	Neutral
2	WL	Walk
3	GH	Get Hit
4	A1	Attack 1
5	A2	Attack 2
6	BL	Block
7	SC	Cast
8	S1	Skill 1
9	S2	Skill 2
10	S3	Skill 3
11	S4	Skill 4
12	DD	Dead
13	GH	Knockback
14	xx	Sequence
15	RN	Run

Chance1 (to Chance6) - This is the base chance for the hireling to use the related Skill. All Chance values are summed together as a denominator value for a random roll to determine which skill to use.
ChancePerLvl1 (to ChancePerLvl6) - This is the chance for the hireling to use the related Skill, affected by the difference in the hireling's current Level and the hireling's "Level" field. All Chance values are summed together as a denominator value for a random roll to determine which skill to use. Each skill Chance is calculated with the following formula: ["Chance#"] + ["ChancePerLvl#"] * [Difference of Current Level and "Level"] / 4
Level1 (to Level6) - The starting Level for the related Skill.
LvlPerLvl1 (to LvlPerLvl6) - A modifier to increase the related Skill level for every Level gained. This is used in the following calculated to determine the current skill level: [Current Skill Level] = FLOOR(["Level"] + ((["LvlPerLvl"] * [Difference of Current Level and "Level"]) / 32))

HiringMaxLevelDifference - This is used to generate a range with this value plus and minus with the player's current Level. In the hiring UI window, hirelings start with a random Level that is between this range.
resurrectcostmultiplier - A modifier used to calculate the hireling's current resurrect cost. Used in the following formula: [Resurrect Cost] = [Current Level] * [Current Level] / ["resurrectcostdivisor"] * ["resurrectcostmultiplier"]
resurrectcostdivisor - A modifier used to calculate the hireling's current resurrect cost. Used in the following formula: [Resurrect Cost] = [Current Level] * [Current Level] / ["resurrectcostdivisor"] * ["resurrectcostmultiplier"]
resurrectcostmax - This is the maximum Gold cost to resurrect this hireling
equivalentcharclass – Determines what class this hireling is treated like under the hood when calculating skill level bonuses and gear restrictions.

hirelingdesc.txt

Overview

This file controls attributes about player mercenaries that relate to the monster class but not the specific statblocks.

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

id – The id of the hireling monster class as defined in monstats.txt
alternateVoice – Boolean field. If equals 1, then the hireling will use the alternate (feminine) voice type for voice lines. If equals 0, then it will use the masculine voice type.

LevelGroups.txt

Overview

This file controls how the game groups levels together. This has currently no gameplay purpose and is mainly used for condensing level names in desecrated (terror) zone messaging.

Data Fields

Name - Defines the unique name pointer for the level group, which is used in other files
Id - Defines the unique numeric ID for the level group, which is used in other files
GroupName – String Field. Used for displaying the name of the level group, such as when all levels in a group have been desecrated.

Levels.txt

Overview

This file controls how the game controls the area levels, including how the level is built, what rules are allowed on the level, and what monsters/objects can spawn on the level.

This file uses the following files: AutoMap.txt, LevelGroups.txt, LvlMaze.txt, LvlPrest.txt, LvlSub.txt, LvlTypes.txt, LvlWarp.txt, monstats.txt, Objgroup.txt

Data Fields

Name - Defines the unique name pointer for the area level, which is used in other files
Id - Defines the unique numeric ID for the area level, which is used in other files
Pal - Defines which palette file to use for the area level. This uses index values from 0 to 4 to convey Act 1 to Act 5.
Act - Defines the Act number that the area level is a part of. This uses index values from 0 to 4 to convey Act 1 to Act 5.

QuestFlag - Controls what quest record that the player needs to have completed before being allowed to enter this area level, while playing in Classic Mode. Each quest can have multiple quest records, and this field is looking for a specific quest record from a quest.

QuestFlagEx - Controls what quest record that the player needs to have completed before being allowed to enter this area level, while playing in Expansion Mode. Each quest can have multiple quest records, and this field is looking for a specific quest record from a quest.

Code	Description
0	Act 1 Prologue Seen
1	Den of Evil Complete
2	Sisters' Burial Grounds Complete
3	Tools of the Trade Complete
4	The Search for Cain Complete
5	The Forgotten Tower Complete
6	Sisters to the Slaughter Complete
7	Act 1 Traversed
8	Act 2 Prologue Seen
9	Radament's Lair Complete
10	The Horadric Staff Complete
11	The Tainted Sun Complete
12	The Arcane Sanctuary Complete
13	The Summoner Complete
14	The Seven Tombs Complete
15	Act 2 Traversed
16	Act 3 Prologue Seen
17	Lam Esen's Tome Complete
18	Khalim's Will Complete
19	Blade of the Old Religion Complete
20	The Golden Bird Complete
21	The Blackened Temple Complete
22	The Guardian Complete
23	Act 3 Traversed
24	Act 4 Prologue Seen
25	The Fallen Angel Complete
26	Terror's End Complete
27	The Hellforge Complete
28	Act 4 Traversed
29	Rogue Warning Complete
30	Guard in Town Warning Complete
31	Guard in Desert Warning Complete
32	Dark Wanderer Seen
33	Angel Warning Complete
34	Act 5 Prologue Seen
35	Siege on Harrogath Complete
36	Rescue on Mount Arreat Complete
37	Prison of Ice Complete
38	Betrayal of Harrogath Complete
39	Rite of Passage Complete
40	Eve of Destruction Complete
41	Respec from Akara Complete

Layer - Defines a unique numeric ID that is used to identify which Automap data belongs to which area level when saving and loading data from the character save.
SizeX & SizeX(N) & SizeX(H) - Specifies the Length tile size values of an entire area level, which are used for determining how to build the level, for Normal, Nightmare, and Hell Difficulty, respectively.
SizeY & SizeY(N) & SizeY(H) - Specifies the Width tile size values of an entire area level, which are used for determining how to build the level, for Normal, Nightmare, and Hell Difficulty, respectively.
OffsetX & OffsetY - Specifies the location offset coordinates (measured in tile size) for the origin point of the area level in the world.

Depend - Assigns another level to be this area level's depended level, which controls this area level's position and how it starts building its tiles. Uses the level "Id" field. If this equals 0, then ignore this.
Teleport - Controls the functionality of the Sorceress Teleport skill and the Assassin Dragon Flight skill on the area level

Code	Description
0	Teleport is disabled on the area level
1	Teleport is enabled on the area level
2	Teleport is enabled on the area level but adheres to the collision of the rooms

Rain - Boolean Field. If equals 1, then allow rain to play its effects on the area level. If the level is part of Act 5, then it will snow on the area level, instead of rain. If equals 0, then it will never rain on the area level.
Mud - Boolean Field. If equals 1, then random bubbles will animate on the tiles that are flagged as water tiles. If equals 0, then ignore this.
NoPer - Boolean Field. If equals 1, then allow the use of display option of Perspective Mode while the player is in the level. If equals 0, then disable the option of Perspective Mode and force the player to use Orthographic Mode while the player is in the level.
LOSDraw - Boolean field. If equals 1, then the level will check the player's line of sight before drawing monsters. If equals 0, then ignore this.
FloorFilter - Boolean field. If equals 1 and if the floor's layer in the area level equals 1, then draw the floor tiles with a linear texture sampler. If equals 0, then draw the floor tiles with a nearest texture sampler.
BlankScreen - Boolean field. If equals 1, then draw the area level screen. If equals 0, then do not draw the area level screen, meaning that the level will be a blank screen.
DrawEdges - Boolean field. If equals 1, then draw the areas in levels that are not covered by floor tiles. If equals 0, then ignore this.

DrlgType - Determines the type of Dynamic Random Level Generation used for building and handling different elements of the area level. Uses a numeric code to handle which type of DRLG is used.

Code	Description

0	None
1	Maze
2	Preset
3	Outdoor

LevelType - Defines the Level Type used for this area level. Uses the Level Type's ID, which is determined by what order it is defined in the LvlType.txt file.
SubType - Controls the group of tile substitutions for the area level (see LvlSub.txt). There are defined sub types to choose from.

Code	Description
-1	None
0	Border Cliffs
1	Border Middle
2	Border Corner
3	Border General
4	Border Wild Waypoint
5	Border Wild Shrine
6	Border Wild Themes
7	Border Desert Waypoint
8	Border Desert Shrine
9	Border Desert Themes
10	Siege Dirt
11	Siege Snow
12	Barricade
13	Broken Barricade

SubTheme - Controls which theme number to use in a Level Substitution (See LvlSub.txt). The allowed values are 0 to 4, which convey which "Prob#", "Trials#", and "Max#" field to use from the LvlSub.txt file. If this equals -1, then there is no sub theme for the area level.
SubWaypoint - Controls the level substitutions for adding waypoints in the area level (see LvlSub.txt). This uses a defined sub type to choose from (See "SubType"). This will depend on the room having a waypoint tile.
SubShrine - Controls the level substitutions for adding shrines in the area level (see LvlSub.txt). This uses a defined sub type to choose from (See "SubType"). This will depend on the room allowing for a shrine to spawn.

Vis0 (to Vis7) - Defines the visibility of other area levels involved with this area level, allowing for travel functionalities between levels. This uses the "Id" field of another defined area level to link with this area level. If this equals 0, then no area level is specified.
Warp0 (to Warp7) - Uses the "Id" field from LevelWarp.txt**,** which defines which Level Warp to use when exiting the area level. This is connected with the definition of the related "Vis#" field. If this equals -1, then no Level Warp is specified which should also mean that the related "Vis#" field is not defined.

Intensity - Controls the intensity value of the area level's ambient colors. This affects brightness of the room's RGB colors. Uses a value between 0 and 128. If all these related fields equal 0, then the game ignores setting the area level's ambient colors.
Red - Controls the red value of the area level's ambient colors. Uses a value between 0 and 255.
Green - Controls the green value of the area level's ambient colors. Uses a value between 0 and 255.
Blue - Controls the blue value of the area level's ambient colors. Uses a value between 0 and 255.

Portal - Boolean Field. If equals 1, then this area level will be flagged as a portal level, which is saved in the player's information and can be used for keeping track of the player's portal functionalities. If equals 0, then ignore this.
Position - Boolean Field. If equals 1, then enable special casing for positioning the player on the area level. This can mean that the player could spawn on a different location on the area level, depending on the level room's position type. An example can be when the player spawns in a town when loading the game, or using a waypoint, or using a town portal. If equals 0, then ignore this.

SaveMonsters - Boolean Field. If equals 1, then the game will save the monsters in the area level, such as when all players leave the area level. If equals 0, then monsters will not be saved and will be removed. This is usually disabled for areas where monsters do not spawn.

Quest - Controls what quest record is attached to monsters that spawn in this area level. This is used for specific quests handling lists of monsters in the area level.

Code	Description
0	Act 1 Prologue Seen
1	Den of Evil Complete
2	Sisters' Burial Grounds Complete
3	Tools of the Trade Complete
4	The Search for Cain Complete
5	The Forgotten Tower Complete
6	Sisters to the Slaughter Complete
7	Act 1 Traversed
8	Act 2 Prologue Seen
9	Radament's Lair Complete
10	The Horadric Staff Complete
11	The Tainted Sun Complete
12	The Arcane Sanctuary Complete
13	The Summoner Complete
14	The Seven Tombs Complete
15	Act 2 Traversed
16	Act 3 Prologue Seen
17	Lam Esen's Tome Complete
18	Khalim's Will Complete
19	Blade of the Old Religion Complete
20	The Golden Bird Complete
21	The Blackened Temple Complete
22	The Guardian Complete
23	Act 3 Traversed
24	Act 4 Prologue Seen

25	The Fallen Angel Complete
26	Terror's End Complete
27	The Hellforge Complete
28	Act 4 Traversed
29	Rogue Warning Complete
30	Guard in Town Warning Complete
31	Guard in Desert Warning Complete
32	Dark Wanderer Seen
33	Angel Warning Complete
34	Act 5 Prologue Seen
35	Siege on Harrogath Complete
36	Rescue on Mount Arreat Complete
37	Prison of Ice Complete
38	Betrayal of Harrogath Complete
39	Rite of Passage Complete
40	Eve of Destruction Complete
41	Respec from Akara Complete

WarpDist - Defines the minimum pixel distance from a Level Warp that a monster is allowed to spawn near. Tile distance values are converted to game pixel distance values by multiplying the tile distance value by 160 / 32, where 160 is the width of pixels of a tile.

MonLvl & MonLvl(N) & MonLvl(H) - Controls the overall monster level for the area level for Normal, Nightmare, and Hell Difficulty, respectively. This is for Classic mode only. This can affect the highest item level allowed to drop in this area level.
MonLvlEx & MonLvlEx(N) & MonLvlEx(H) - Controls the overall monster level for the area level for Normal, Nightmare, and Hell Difficulty, respectively. This is for Expansion mode only. This can affect the highest item level allowed to drop in this area level.
MonDen & MonDen(N) & MonDen(H) - Controls the monster density on the area level for Normal, Nightmare, and Hell Difficulty, respectively. This is a random value out of 100000, which will determine whether to spawn or not spawn a monster pack in the room of the area level. If this value equals 0, then no random monsters will populate on the area level.
MonUMin & MonUMin(N) & MonUMin(H) - Defines the minimum number of Unique Monsters that can spawn in the area level for Normal, Nightmare, and Hell Difficulty, respectively. This field depends on the related "MonDen" field being defined.
MonUMax & MonUMax(N) & MonUMax(H) - Defines the maximum number of Unique Monsters that can spawn in the area level for Normal, Nightmare, and Hell Difficulty, respectively. This field depends on the related "MonDen" field being defined. Each room in the area level will attempt to spawn a Unique Monster with a 5/100 random chance, and this field's value will cap the number of successful attempts for the entire area level.
MonWndr - Boolean Field. If equals 1, then allow Wandering Monsters to spawn on this area level (see wanderingmon.txt). This field depends on the related "MonDen" field being defined. If equals 0, then ignore this.
MonSpcWalk - Defines a distance value, used to handle monster pathing AI when the level has certain pathing blockers, such as jail bars or rivers. In these cases, monsters will walk randomly until a player is located within this distance value or when the monsters find a possible path to target the player. If this equals 0, then ignore this field.

NumMon - Controls the number of different monsters randomly spawn in the area level. The maximum value is 13. This controls the number of random selections from the 25 related "mon#" and "umon#" fields or "nmmon#" fields, depending on the game difficulty.
mon1 (to mon25) - Defines which monsters can spawn on the area level for Normal Difficulty. Uses the monster "Id" field from the monstats.txt file.
rangedspawn - Boolean Field. If equals 1, then for the first monster, try to pick a ranged type. If equals 0, then ignore this.
nmon1 (to nmon25) - Defines which monsters can spawn on the area level for Nightmare Difficulty and Hell Difficulty. Uses the monster "Id" field from the monstats.txt file.
umon1 (to umon25) - Defines which monsters can spawn as Unique monsters on this area level for Normal Difficulty. Uses the monster "Id" field from the monstats.txt file.

cmon1 (to cmon4) - Defines which Critter monsters can spawn on the area level. Uses the monster "Id" field from the monstats.txt file. Critter monsters are determined by the "critter" field from the monstats2.txt file.
cpct1 (to cpct4) - Controls the percent chance (out of 100) to spawn a Critter monster on the area level.
camt1 (to camt4) - Controls the amount of Critter monsters to spawn on the area level after they succeeded their random spawn chance from the related "cpct#" field.

Themes - Controls the type of theme when building a level room. This value is a summation of possible values to build a bit mask for determining which themes to use when building a level room. For example, a value of 60 means that the area level can have the following themes: 32, 16, 8, 4.

Code	Description
0	No Theme
1	Object Empty Theme (no objects spawn)
2	Barrel Theme (create random barrel objects)
4	Shrine Theme
8	Treasure Theme (create random items)
16	Armor Stand Theme
32	Weapon Rack Theme

SoundEnv - Uses the "Index" field from SoundEnviron.txt, which controls what music is played while the player is in the area level

Waypoint - Defines the unique numeric ID for the Waypoint in the area level. If this value is greater than or equal to 255, then ignore this field.

LevelName - String Field. Used for displaying the name of the area level, such as when in the UI when the Automap is being viewed.
LevelWarp - String Field. Used displaying the entrance name of the area level on Level Warp tiles that link to this area level. For example, when the player mouse hovers over a tile to warp to the area level, then this string is displayed.
LevelEntry - String Field. Used for displaying the UI popup title string when the player enters the area level.

ObjGrp0 (to ObjGrp7) - Uses a numeric ID to define which possible Object Groups to spawn in this area level (See objgroup.txt). The game will go through each of these fields, so there can be more than 1 Object Group used in an area level. If this value equals 0, then ignore this.

ObjPrb0 (to ObjPrb7) - Determines the random chance (out of 100) for each Object Group to spawn in the area level. This field depends on the related “ObjGrp#” field being defined.

LevelGroup – Defines what group this level belongs to. Used for condensing level names in desecrated (terror) zones messaging. See LevelGroups.txt.

LvlMaze.txt

Overview

This file controls the sizes of the underground area levels. This file uses the levels from Levels.txt and specifies the sizes for each room, which can mean how many Level Presets to use to build out the entire randomly generated area.

Data Fields

Name - This is a reference field to describe the area level. Ideally this should match the name of the area level from the Levels.txt file.
Level - This refers to the “Id” field from the Levels.txt file

Rooms & Rooms(N) & Rooms(H) - Controls the total number of rooms that a Level Maze will generate when playing the game in Normal Difficulty, Nightmare Difficulty, and Hell Difficulty, respectively.

SizeX & SizeY - Controls the length and width sizes of each room (ds1 map files) that is added to the Level Maze. This is measured in tile sizes.

Merge - This value affects the probability that a room gets an adjacent room linked next to it. This is a random chance out of 1000.

LvlPrest.txt

Overview

This file controls the values for each Level Preset. A Level Preset is a static area composed of tiles that is used to construct entire area levels found in the game.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - This is a reference field to define the Level Preset
Def - Defines the unique numeric ID for the Level Preset. This is referenced in other files.
LevelId - This refers to the “Id” field from the Levels.txt file. If this value is not equal to 0, then this Level Preset is used to build that entire area level. If this value is equal to 0, then the Level Preset does not define the entire area level and is used as a part of constructing area levels.

Populate - Boolean Field. If equals 1, then units are allowed to spawn in the Level Preset. If equals 0, then units will never spawn in the Level Preset.
Logicals - Boolean Field. If equals 1, then the Level Preset allow for wall transparency to function. If equals 0, then walls will always appear solid.
Outdoors - Boolean Field. If equals 1, then the Level Preset will be classified as an outdoor area, which can mean that lighting will function differently. If equals 0, then the Level Preset will be classified as an indoor area.
Animate - Boolean Field. If equals 1, then the game will animate the tiles in the Level Preset. If equals 0, then ignore this.
KillEdge - Boolean Field. If equals 1, then the game will remove tiles that border the size of the Level Preset. If equals 0, then ignore this.
FillBlanks - Boolean Field. If equals 1, then all blank tiles in the Level Preset will be filled with unwalkable tiles. If equals 0, then ignore this.

SizeX & SizeY - Specifies the Length and Width tile size values of the Level Preset, which are used for determining how big to build area levels. These values are equal to 0 for Level Presets that are static.

AutoMap - Boolean Field. If equals 1, then this Level Preset will be automatically completely revealed on the Automap. If equals 0, then this Level Preset will be hidden on the Automap and will need to be explored.
Scan - Boolean Field. If equals 1, then this Level Preset will allow the usage of warping with waypoints (This requires that the Level Preset has a waypoint object). If equals 0, then ignore this.

Pops - Defines how many Pop tiles are defined in the Level Preset file. These Pop tiles are mainly used for controlling the roof and wall popping when a player enters a building in an area.
PopPad - Determines the size of the Pop tile area, by using an offset value. This offset value can increase or decrease the size of the Pop tile size if it has a positive or negative value.

Files - Determines the number of different versions to use for the Level Preset. This value acts as a range, which the game will use for randomly choosing one of the “File#” fields to build the Level Preset. This is how the Level Presets have variety when the area level is being built.
File1 (to File6) - Specifies the name of which ds1 file to use. The ds1 files contain data for building Level Presets. If this value equals 0, then this field will be ignored. The number of these defined fields should match the value used in the “Files” field.
Dt1Mask - This functions as a bit field mask with a size of a 32 bit value. This explains to the ds1 file which of the 32 dt1 tile files to use from a Level Type when assembling the Level Preset. Each “File#” field from LevelType.txt is assigned a bit value, up to the 32 possible bit values. (For example: File1 = 1, File2=2, File3 = 4, File4=8, File5=16….File32 = 2147483648). To build the “Dt1Mask”, you would select which “File#” fields to use from LevelTypes.txt and add their associated bit values together for a total value. This total value is the bitmask value.

LvlSub.txt

Overview

This file controls how tiles can be substituted in for other tiles. The game will divide the level into clusters and iterate through these clusters to randomly substitute tiles with different ones for more visual diversity.

Data Fields

Name - This is a reference field to describe the Level Substitution
Type - This refers to the “SubType” field from the Levels.txt file. This defines a group that multiple substitutions can share.
File - Specifies the name of which ds1 file to use. The ds1 files contain data for building Level Presets.

CheckAll - Boolean Field. If equals 1, then substitute each tile in the room. If equals 0, then substitute random tiles in the room.
BordType - This controls how often substituting tiles can work for border tiles

Code	Description
0	Single One Only. This allows substituting for 1 border in total
1	One Per Cluster. This allows substituting 1 border for each cluster in the level
Other values	Allow substituting borders for all of the level

GridSize - Controls the tile size of a cluster for substituting tiles. This evenly affects both the X and Y size values of a room.
Dt1Mask - This functions as a bit field mask with a size of a 32 bit value. This explains to the ds1 file which of the 32 dt1 tile files to use from a Level Type when assembling selecting a tile for substitution. Each “File#” field from LevelType.txt is assigned a bit value, up to the 32 possible bit values. (For example: File1 = 1, File2=2, File3 = 4, File4=8, File5=16….File32 = 2147483648). To build the “Dt1Mask”, you would select which “File#” fields to use from LevelTypes.txt and add their associated bit values together for a total value. This total value is the bitmask value.

Prob0 (to Prob4) - This value affects the probability that the tile substitution is used. This is a random chance out of 100. Which “Prob#” field that is checked depends on the “SubTheme” value from the Levels.txt file.
Trials0 (to Trials4) - Controls the number of times to randomly substitute tiles in a cluster. If this value equals -1, then the game will try to do as many tile substitutions that can be allowed based on the cluster and tile size. This field depends on the “CheckAll” field being equals to 0.
Max0 (to Max4) - The maximum number of clusters of tiles to substitute randomly. This field depends on the “CheckAll” field being equals to 0.

LvlTypes.txt

Overview

This file controls which files containing tile graphics are used for creating maps. This looks at dt1 files, which contain tile images of the environments foundin the game. Each line in this file defines a Level Type and what files it uses.

The order of each Level Type defined in this file will convey what ID value it has, which is referenced by the following files: Levels.txt, LvlPrest.txt
The order of these Level Types should not be changed

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - This is a reference field to define the Level Type

File 1 (to File 32) - Specifies the name of which dt1 file to use. The dt1 files contain the images for each area tile found in each Act. If this value equals 0, then this field will be ignored.

Act - Defines which Act is related to the Level Type. When loading an Act, the game will only use the Level Types associated with that Act number. Uses a decimal number to convey each Act number (Ex: A value of 3 means Act 3)

LvlWarp.txt

Overview

This file controls how the player is moved to different area levels, such as entrances and exits between different areas. This player transportation of between levels is defined as a Level Warp. Level Warps function as special tiles that are added to the area for controlling the location for where to transport the player.

This file is used by the Levels.txt file.

Data Fields

Name - This is a reference field to define the Level Warp
Id - Defines the numeric ID for the type of Level Warp. This ID can be shared between multiple Level Warps if those Level Warps want to use the same functionality. This is referenced in other files.

SelectX & SelectY - These values define the horizontal and vertical offsets (respectively) of the starting left corner position of the Level Warp area. This is treated as the starting position to select the interactable Level Warp area in the area level. This value is measured in pixels.
SelectDX & SelectDY - These values define the horizontal and vertical offsets (respectively) of the offset from the starting position of the Level Warp area. This is added with the “SelectX” & “SelectY” fields (respectively) to determine the overall size and position of the Level Warp in the area level. This value is measured in pixels.

ExitWalkX & ExitWalkY - These values define the horizontal and vertical positions (respectively) of the destination location where the player will walk to after exiting to this Level Warp. This value is measured with a sub-tile offset from the base position of the Level Warp. One full tile on a level is composed of a grid of 5x5 sub-tiles.

OffsetX & OffsetY - These values define the horizontal and vertical positions (respectively) of the sub-tile for the Level Warp, where the player will appear when exiting to this area level. This value is measured with a sub-tile offset from the base position of the Level Warp. One full tile on a level is composed of a grid of 5x5 sub-tiles.

LitVersion - Boolean Field. If equals 1, then Level Warp tiles will change their appearance when highlighted. If equals 0, then the Level Warp tiles will not change appearance when highlighted.
Tiles - Defines an index offset to determine which tile to use in the tile set for the highlighted version of the Level Warp. These tiles are loaded and hidden/revealed when the player mouse hovers over the Level Warp tiles. This relies on “LitVersion” being enabled.
NoInteract - Boolean Field. If equals 1, then the Level War cannot be directly interacted by the player. If equals 0, then the player can interact with the Level Warp.

Direction - Defines the orientation of the Level Warp. Uses a specific string code.

Code	Description
l	Left. If this is selected, then the tile type direction should match this.
r	Right. If this is selected, then the tile type direction should match this.
b	Both. This can mean that the Level Warp can be reassigned its direction.

UniqueId - Defines the unique numeric ID for the Level Warp. Each Level Warp should have a unique ID so that the game can handle loading that specific Level Warp’s related files.

MagicPrefix.txt

Overview

This file controls what item affixes (groups of item modifiers) are applied as the prefix for an item
These item affixes will appear at the start of an item’s name

This file is loaded together with other similar files in the following order: magicsuffix.txt, magicprefix.txt, automagic.txt
These combined files form the Item Mods structure.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - Defines the item affix name
version - Defines which game version to use this item affix (<100 = Classic mode | 100 = Expansion mode)

spawnable - Boolean Field. If equals 1, then this item affix is used as part of the game’s randomizer for assigning item modifiers when an item spawns. If equals 0, then this item affix is never used.
rare - Boolean Field. If equals 1, then this item affix can be used when randomly assigning item modifiers when a rare item spawns. If equals 0, then this item affix is not used for rare items.

level - The minimum item level required for this item affix to spawn on the item. If the item level is below this value, then the item affix will not spawn on the item.
maxlevel - The maximum item level required for this item affix to spawn on the item. If the item level is above this value, then the item affix will not spawn on the item.
levelreq - The minimum character level required to equip an item that has this item affix

classspecific - Controls if this item affix should only be used for class specific items. This relies on the class specified in the “Class” field from ItemTypes.txt, for the specific item.

Code	Description
(empty)	Any Class
ama	Amazon only
bar	Barbarian only
pal	Paladin only
nec	Necromancer only
sor	Sorceress only
dru	Druid only
ass	Assassin only

class - Controls which character class is required for the class specific level requirement “classlevelreq” field

Code	Description
(empty)	None
ama	Amazon
bar	Barbarian
pal	Paladin
nec	Necromancer
sor	Sorceress
dru	Druid
ass	Assassin

classlevelreq - The minimum character level required for a specific class in order to equip an item that has this item affix. This relies on the class specified in the “class” field. If equals null, then the class will default to using the “levelreq” field.

frequency - Controls the probability that the affix appears on the item (a higher value means that the item affix will appear on the item more often). This value gets summed together with other “frequency” values from all possible item affixes that can spawn on the item, and then is used as a denominator value for the randomizer. Whichever item affix is randomly selected will be the one to appear on the item. The formula is calculated as the following: [Item Affix Selected] = [“frequency”] / [Total Frequency]. If the item has a magic level (from the “magic lvl” field in weapons.txt/armor.txt/misc.txt) then the magic level value is multiplied with this value. If equals 0, then this item affix will never appear on an item.
group - Assigns an item affix to a specific group number. Items cannot spawn with more than 1 item affix with the same group number. This is used to guarantee that certain item affixes do not overlap on the same item. If this field is null, then the group number will default to group 0.

mod1code (to mod3code) - Controls the item properties for the item affix (Uses the “code” field from Properties.txt)
mod1param (to mod3param) - The “parameter” value associated with the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
mod1min (to mod3min) - The “min” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)
mod1max (to mod3 max) - The “max” value to assign to the listed property (mod). Usage depends on the property function (See the “func” field on Properties.txt)

transformcolor - Controls the color change of the item after spawning with this item affix. If empty, then the item affix will not change the item’s color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

itype1 (to itype7) - Controls what Item Types are allowed to spawn with this item affix. Uses the “code” field from ItemTypes.txt
etype1 (to etype5) - Controls what Item Types are excluded to spawn with this item affix. Uses the “code” field from ItemTypes.txt

multiply - Multiplicative modifier for the item’s buy and sell costs, based on the item affix (Calculated in 1024ths for buy cost and 4096ths for sell cost)
add - Flat integer modification to the item’s buy and sell costs, based on the item affix

MagicSuffix.txt

Overview

This file controls what item affixes (groups of item modifiers) are applied as the suffix for an item
These item affixes will appear at the end of the item’s name

This file is loaded together with other similar files in the following order: magicsuffix.txt, magicprefix.txt, automagic.txt
These combined files form the Item Mods structure.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - Defines the item affix name
version - Defines which game version to use this item affix (<100 = Classic mode | 100 = Expansion mode)

spawnable - Boolean Field. If equals 1, then this item affix is used as part of the game's randomizer for assigning item modifiers when an item spawns. If equals 0, then this item affix is never used.
rare - Boolean Field. If equals 1, then this item affix can be used when randomly assigning item modifiers when a rare item spawns. If equals 0, then this item affix is not used for rare items.

level - The minimum item level required for this item affix to spawn on the item. If the item level is below this value, then the item affix will not spawn on the item.
maxlevel - The maximum item level required for this item affix to spawn on the item. If the item level is above this value, then the item affix will not spawn on the item.
levelreq - The minimum character level required to equip an item that has this item affix

classspecific - Controls if this item affix should only be used for class specific items. This relies on the class specified in the "Class" field from ItemTypes.txt, for the specific item.

Code	Description
(empty)	Any Class
ama	Amazon only
bar	Barbarian only
pal	Paladin only
nec	Necromancer only
sor	Sorceress only
dru	Druid only
ass	Assassin only

class - Controls which character class is required for the class specific level requirement "classlevelreq" field

Code	Description
(empty)	None
ama	Amazon
bar	Barbarian
pal	Paladin
nec	Necromancer
sor	Sorceress
dru	Druid
ass	Assassin

classlevelreq - The minimum character level required for a specific class in order to equip an item that has this item affix. This relies on the class specified in the "class" field. If equals null, then the class will default to using the "levelreq" field.

frequency - Controls the probability that the affix appears on the item (a higher value means that the item affix will appear on the item more often). This value gets summed together with other "frequency" values from all possible item affixes that can spawn on the item, and then is used as a denominator value for the randomizer. Whichever item affix is randomly selected will be the one to appear on the item. The formula is calculated as the following: [Item Affix Selected] = ["frequency"] / [Total Frequency]. If the item has a magic level (from the "magic lvl" field in weapons.txt/armor.txt/misc.txt) then the magic level value is multiplied with this value. If equals 0, then this item affix will never appear on an item.
group - Assigns an item affix to a specific group number. Items cannot spawn with more than 1 item affix with the same group number. This is used to guarantee that certain item affixes do not overlap on the same item. If this field is null, then the group number will default to group 0.

mod1code (to mod3code) - Controls the item properties for the item affix. (Uses the "code" field from Properties.txt)
mod1param (to mod3param) - The "parameter" value associated with the listed property (mod). Usage depends on the property function (See the "func" field on Properties.txt)
mod1min (to mod3min) - The "min" value to assign to the listed property (mod). Usage depends on the property function (See the "func" field on Properties.txt)
mod1max (to mod3 max) - The "max" value to assign to the listed property (mod). Usage depends on the property function (See the "func" field on Properties.txt)

transformcolor - Controls the color change of the item after spawning with this item affix. If empty, then the item affix will not change the item's color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

itype1 (to itype7) - Controls what Item Types are allowed to spawn with this item affix. Uses the “code” field from ItemTypes.txt
etype1 (to etype5) - Controls what Item Types are excluded to spawn with this item affix. Uses the “code” field from ItemTypes.txt

multiply - Multiplicative modifier for the item’s buy and sell costs, based on the item affix (Calculated in 1024ths for buy cost and 4096ths for sell cost)
add - Flat integer modification to the item’s buy and sell costs, based on the item affix

Missiles.txt

Overview

This file controls the different functions for all missiles and their statistics. Missiles are projectiles used throughout the game for attacks, skills, and special effects.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Missile - Defines the unique name ID for the missile, which is how other files can reference the missile. The order of defined missiles will determine their ID numbers, so they should not be reordered.

pCltDoFunc - Uses an ID value to select a function for the missile’s behavior while active every frame on the client side. This is more about handling the local graphics while the missile is moving.

Code	Parameters	Description
0 (or empty)		Do nothing
1		ProcessMissileMode - Perform the standard client missile move function. This is called in most of the other functions
2		ProcessBlood - Kill the missile if it goes off screen. Otherwise, set the number of duration frames for the missile to equal 128 and perform the standard missile move function.
3	CltCalc1 CltSubMissile1	PoisonJavelin - Create a sub missile each frame that the current missile is moving where “CltCalc1” controls the number of subloops for the created sub missile.
4	CltParam1 CltParam2 CltParam3 CltSubMissile1	PoisonCloud - Create multiple sub missiles randomly in a area where the parameters control the spawn rate, number of sub missiles to spawn per rate, and the spawn radius size, respectively.
5	SubStart SubStop	Firewall - Create an animation sequence for the missile where the difference of the “SubStart” and “SubStop” fields control the length of the sequence within the animation, and the game will randomly choose a frame within this sequence to loop back to.
6	CltParam1 CltSubMissile1 CltSubMissile2 CltSubMissile3	FirewallMaker - Randomly create 1 of the 3 possible sub missiles, where the parameter controls the random chance that the sub missile spawns with no light.
7	CltParam1	ProcessGuidedArrow - Try to retarget the missile on a possible target unit where the parameter controls the rate at which the missile should attempt to retarget again.
8	CltParam1 CltSubMissile1 InitSteps	LightningMaker - Attempt to create a sub missile where the parameter controls the number of subloops for the sub missile (minimum value equals 1). The sub missile is only created if the current missile’s has existed for greater than or equal to the “InitSteps” value.
9	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2 ProgSound	ProcessMeteorCenter - Create each sub missile where the parameters control the number of frames, the missile fall rate (Used to calculate the starting height), and the missile slide rate (used to calculated the starting position offset), respectively. Also attempt to play the “ProgSound” sound at 2 frames before the “CltParam1” frame count.
10	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2	ProcessMonBliz - Randomly create 1 of the 2 sub missiles in an area radius, where the parameters control the baseline radius size (affected by the missile level), the spawn frequency (minimum equals 3 frames, affected by the missile level), and the level divisor (affects the difference in value for the other fields per missile level). The “CltParam1” and “CltParam2” parameters also can control the missile fall distance and fall rate.
11		ProcessHoldLast - At the end of the missile’s animation, set it to dead and destroy its light. Otherwise, run the ProcessMissileMode function.
12	CltParam1 CltParam2	ProcessScreenShake - Run the ProcessMissileMode function and also call the camera shaking function. The “CltParam1” field controls the camera shake magnitude, and the “CltParam2” field controls the camera shake duration.
13	CltParam1 CltParam2 CltSubMissile1 CltSubMissile2	ProcessBlizzard - Randomly create 1 of the 2 missiles in a calculated area every duration. The radius and spawn frequency are controlled by the missile’s linked skill’s “calc1” and “calc2” fields. The “CltParam1” and “CltParam2” parameters control the sub missile fall distance and fall rate.
14	CltParam3 CltParam4 CltParam5	FingerMageSpider2 - Shoot a missile that can be retargeted on a unit. The parameters control the retarget frequency, the retarget distance range, and the retarget position offset (minimum value equals 1), respectively.
15	CltParam1 CltParam2 CltParam3 CltParam4 CltParam5 CltSubMissile1	FingerMageSpider - Shoot a missile that can be retargeted on a unit and can spawn a sub missile. The parameters control the number of frames to wait before spawning the sub missile (minimum value equals 1), the number of frames to wait before processing the missile, the retarget frequency, the retarget distance range, and the retarget position offset (minimum value equals 1), respectively.

16	CltSubMissile1	DiabWallMaker - Shoot a missile using the ProcessMissileMode function where every frame while moving it can create a sub missile that will have a random chance to spawn with no light
17	CltParam1 CltSubMissile1	ProcessCurse - Shoot a missile that will create a disc of sub missiles every 3 frames after a specified frame duration controlled by the parameter. The radius and density of the disc of sub missile is controlled by the current missile's radius value, which is given to the missile by the skill function.
18	CltParam1 CltSubMissile1 InitSteps	SimpleTrailMaker - Shoot a missile that will create a sub missile after the "InitSteps" frame count, where the parameter controls the sub missile's subloops. The sub missile will follow the same direction, offset, and path of the current missile.
19	CltParam1 CltParam2 CltSubMissile1	ProcessFrozenOrb - Shoot a missile that will create sub missiles at a spawn rate and different directions. The "CltParam1" value controls the rate to spawn sub missiles, and the "CltParam2" value controls the direction index increment for looping through which direction the next created sub missile should fire towards.
20	CltParam1 CltParam2	ProcessFrozenOrbNova - Shoot a missile that will have a delay until processing and then will process its path at a specified rate. The "CltParam1" value controls the activation frame, and the "CltParam2" value controls the periodic rate to process the missile path
21	CltParam1 CltSubMissile1	BRDeathControl - Create a sub missile after a periodic delay that can spawn in a random location in a radius controlled by the parameter. The periodic delay can be 25, 10, 3, or 15 frames, depending how many frames the current missile has lasted.
22	CltParam1 CltSubMissile1	BRDeathLightning - Shoot a missile that will retarget to a randomized direction every periodic frame delay and will create sub missiles while moving.
23		ProcessDOELight - Run the ProcessMissileMode function and ensure that the missile lasts 500 frames if its frame count is less than 100
24	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2	ProcessCairnStones - Create multiple "CltSubMissile1" sub missiles in a radius controlled by "CltParam2" and a spawn frequency controlled by "CltParam3", lasting a duration controlled by "CltParam1". The "CltParam1" field also controls the start time for spawning more "CltSubMissile1" sub missiles and "CltSubMissile2" sub missiles every 8 frames if there are any of the "StoneAlpha" to "StoneTheta" objects in the room.
25	CltParam1 CltParam2 CltParam3 CltSubMissile1	ProcessTowerMist - Shoot a missile that will spawn a sub missile randomly in a defined radius after a periodic delay
26	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2	SmithDeathControl - Create "CltSubMissile1" sub missile in a random position in a radius controlled by "CltParam3" where "CltParam1" controls its spawn rate. Create "CltSubMissile2" sub missile as a lobbing missile in a random position in the same radius where "CltParam2" controls its spawn rate. Also the "CltSubMissile2" sub missile's level is controlled by the following function: 1 + RANDOM(0, 5)
27	CltParam1 CltSubMissile1	SmithFirewallMaker - Create the sub missile every frame while moving. The parameter controls the delay until the current missile needs to retarget in a new direction
28	CltParam1 CltSubMissile1	SmithDoNotDraw - Create 4 sub missiles in different directions after a delay that is controlled by the parameter.
29	CltParam1 CltParam2 CltSubMissile1 CltSubMissile2 CltSubMissile3 ProgSound	ProcessAndyControl0 - Create "CltSubMissile1" after 10 Frames which subsequently will make its own "CltSubMissile1" where its "CltParam1", "CltParam2" and "CltParam3" fields control the Z offset, Z Velocity Max, and Z Acceleration, respectively. Create "CltSubMissile2" randomly in a radius equal to 20 and at a periodic rate controlled by "CltParam1". "CltParam2" controls the duration of the camera shake, which starts after 90 frames. Between frame 115 and 315, create "CltSubMissile3" randomly in a radius equal to 6 where its "CltParam1" and "CltParam2" values control the Z offset and Z velocity.
30	CltParam1 CltParam2 CltParam3 CltSubMissile1	ProcessTowerChestSpawner - Start creating the sub missile periodically after an initial delay controlled by "CltParam1". "CltParam2" controls the periodic rate to spawn the sub missile and "CltParam3" controls the radius size to spawn the sub missile randomly in an area.
31		ProcessHoradricStaff - This function does multiple hardcoded features. Shake the screen after 150 frames. Create 7 "horadriclight" missiles. After 165 frames, create the "horadriclightning" missile at a specific position, direction, and velocity. After 150 frames, create the "dust" missile every other frame.
32		ProcessRadDeath - Create a "radamenthandofgod" missile at an increasing rate based on the total frame count of the current missile
33		ProcessTaintedSun - Create a "taintedsunflash" missile at an increasing rate and at random ranges, based on the total frame count of the current missile. Also add a "horadric_light" overlay on the altar object in the level room.
34		ProcessTaintedSunBall - The missile will change its path and velocities throughout its life cycle using different mods, to follow a crafted pattern
35		ProcessQueenDeath - Every 4 frames, create a "queendeathglob" using a lobbing function in a random position in a defined radius of size 12.
36		ProcessDurielDeath - Create a "explodingarrowexp" missile at an increasing rate and at random ranges, based on the total frame count of the current missile. Also create a "durieldeathrock" missile at an increasing rate and at random ranges, based on the total frame count of the current missile. Run a camera shake function.
37		ProcessDiabloAppears - Run a camera shake function and request to play the "monster_diablo_taunt_1" sound
38		ProcessHellForge - Create multiple lobbing "hffragment1" missiles at an increasing rate in random directions, based on the total frame count of the current missile. Run a camera shake function.
39		ProcessHFragment1 - If the missile has a source unit, then set the missile frame to 0 and run the ProcessMissileMode function
40		ProcessHFragment2 - Periodically create a "hfspirit1" missile after a randomized periodic delay.
41		ProcessSoul - Adjust the animation rate of the missile after certain key frames during the missile's duration.

42	Param1 Param2	ProcessIzualDeath - Create the “izual mist loop” missile every frame in a random radius of size 10. Create a “izual lightning” missile every frame between the value of the “Param1” and “Param2” fields.
43		ProcessAttached - Attach the missile follow its source unit’s position. Kill the missile if the unit is dead.
44	CltSubMissile1	ProcessDistraction - Attach the missile follow its source unit’s position. Create a sub missile while moving.
45	CltParam1 CltParam2 CltParam3 CltSubMissile1	ProcessDistractionFog - Create a number of sub missiles in an area at a defined rate. “CltParam1” controls the spawn rate, “CltParam2” controls the number of missiles to spawn per rate, and “CltParam3” controls the radius to randomly spawn the sub missiles.
46	CltParam1 CltSubMissile1	ProcessTrailJav - Create 2 sub missiles per frame while moving with perpendicular directions and a defined number of subloops controlled by the parameter
47	CltParam1 CltSubMissile1 CltSubMissile2 CltSubMissile3 ProgSound	ProcessMoltenBoulder - Play the “ProgSound” sound if the missile has a bounce value. Run the FirewallMaker function (Code = 6).
48	CltSubMissile1 CltSubMissile2	ProcessEruption - Uses the linked skill’s “calc1” and “calc2” fields to get the spawn radius and spawn frequency for creating the 2 sub missiles. For “CltSubMissile1”, the missile is automatically set to dead mode when it is created.
49	CltParam1 CltSubMissile1	ProcessVines - Periodically spawn the sub missile in the same direction as the current missile where the parameter controls the spawn rate
50	CltParam1 CltParam2 CltParam3 CltParam4 CltParam5 CltSubMissile1	ProcessVolcano - Spawn a sub missile with the lob function at a defined spawn rate at certain frames within the current missile’s duration. “CltParam1” controls the periodic spawn rate, “CltParam2” controls the radius to spawn the sub missile, “CltParam3” controls the starting frame to begin spawning the sub missile, “CltParam4” controls the end frame to stop spawning the sub missile, and “CltParam5” controls the lob start value.
51	CltParam1 CltParam2 CltParam3 CltParam4 CltSubMissile1 CltSubMissile2 ProgSound	ProcessRecycleDelay - Create “CltSubMissile1” at a frame controlled by “CltParam1” in a radius controlled by “CltParam4” and a spawn count controlled by “CltParam3”. Create “CltSubMissile2” sub missile at a frame controlled by “CltParam2” and also request to play the “ProgSound” at the same time.
52	CltSubMissile1	ProcessMakePerpMissiles - Create 2 of the same sub missile every frame while moving that face at perpendicular directions
53	CltParam1 CltSubMissile1	ProcessTigerFury - Create a sub missile every frame while moving and then run the ProcessGuidedArrow function (Code = 7).
54		ProcessAnyaCenter - Create a “anya icemagic” missile every frame until frame 110. Create a decreasing number of “anyasteam1” missiles in a random position and velocity, every frame until frame 110. Perform a camera shake function. Create a “anyasteamvent” missile and a “anyasteam” missile randomly in a radius every 7 frames until frame 200.
55		ProcessAncientDeath - Create a “ancient death cloud” missile every 3 frames randomly in an area in a random direction
56	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2 CltSubMissile3	ProcessBaalTaunt - Randomly choose to spawn one of the 3 sub missiles where each of the parameters control the spawn rate for each of the sub missiles.
57	CltSubMissile1	ProcessBladeShieldCenter - Attach the current missile to the source unit and after a certain delay, create a sub missile every frame that moves in a missile spiral path
58	Param1	ProcessChaosIce - Randomly decide to change the path of the missile to a different direction
59		ProcessWorldstoneChip - If the current Z offset is too low or too high, then stop the missile
60		ProcessHurricane - Every 2 Frames change the missile path to a different direction
61		ProcessOverseerCtrl - Randomly create either the “catapult cold ball” missile or “catapult meteor ball” missile at an increasing spawn rate in a random position in a radius.
62		ProcessNihlathak - This function handles the missile visuals for Nihlathak’s death. • Every frame has a random chance to create 2 of the following missiles in a random position: “nehlithakswoosh” “nehlithakdebris1”, “nehlithakdebris2”, “nehlithakdebris3”, nehlithakdebris4”. • After frame 60, every 20 frames create a “brdeathlightningbolt” missile in a random direction. • At frame 25, create a “nehlithakhole” and “nehlithakholelight” at the missile’s location.
63		ProcessNihlathakHurr - Update the path of the missile at every frame
64		ProcessBaalControl - This function handles the missile visuals for Baal’s death. • Randomly spawn either a “baalfx spirit 1” missile or “baalfx spirit 2” missile at an increasing rate and in a random direction. • At frame 450, create a “baalfx baal head appear” missile • At frame 425, create a “baalfx baal head 1” missile • At frame 375, create a “baalfx baal head 2” missile • At frame 325, create a “baalfx baal head 3” missile • After some time and when the “tyrael3” unit is found within the level, then randomly create either the “baalfx tyreal debris 1”, “baalfx tyreal debris 2”, or “baalfx tyreal debris 3” every frame for a specified duration
65		ProcessBaalSpirit - The missile will follow different modes that can change its path and direction, which are controlled by the ProcessBaalControl function (Code = 64)

66		ProcessWorldstoneShake - Attach the missile to the source unit. Call the camera shake function at random periodic delays. There is a small chance to randomly create either the “baalfx tyreal debris 1”, “baalfx tyreal debris 2”, or “baalfx tyreal debris 3” at periodic durations.
67	CltParam1 CltParam2 CltParam3 CltSubMissile1 CltSubMissile2 CltSubMissile3	ProcessMissileDelayed - Create a sub missile at a specified frame at the source unit’s location. Each parameter controls the specific frame to spawn one of the sub missiles.
68	CltParam1 CltSubMissile1	ProcessSucFireBall - Create a sub missile every frame while the current missile is moving where the number of sub loops for the sub missile is controlled by the parameter

pCltHitFunc - Uses an ID value to select a specialized function for the missile’s behavior when hitting something on the client side. This is more about handling the local graphics at the moment of missile collision.

Code	Parameters	Description
0 (or empty)		Do nothing
1	cHitPar1 cHitPar2 CltHitSubMissile1	HitExplodingArrow - Create a disc of sub missiles with a defined radius and missile count
2	cHitPar1 cHitPar2 cHitPar3 CltHitSubMissile1 Param1 Param2	HitPlagueJavelin - Create an inner and outer disc of sub missiles with a specified density for each ring. Each sub missile will use its “Param1” and “Param2” fields to define their velocities
3	CltHitSubMissile1 CltHitSubMissile2 CltHitSubMissile3	HitOilPotion - Create “CltHitSubMissile1” and then randomly create either “CltHitSubMissile2” or “CltHitSubMissile3”
4	cHitPar1 CltHitSubMissile1	HitDoNova - Create a ring of sub missiles with a defined count
5		Do nothing
6		Do nothing
7		Do nothing
8		Do nothing
9	cHitPar1 cHitPar2 cHitPar3 CltHitSubMissile1 ProgOverlay	HitHolyBolt - Determine whether to impact allies, how that damage is modified by the “dParam1” field based on the unit type hit, and if the missile should be killed on hitting an allied unit. If impacting a unit then create a sub missile. When hitting an ally, create an overlay.
10	ProgOverlay	HitLightningOverlay - Add an overlay on the target unit
11		Do nothing
12	cHitPar1 cHitPar2 CltHitSubMissile1 CltHitSubMissile2 CltHitSubMissile3 SHitCalc1	HitImmolationArrow - Create a ring of sub missiles with a defined radius and density count. Use “SHitCalc1” to control the range (duration) of the sub missile
13	Param2	HitGuidedArrow - Control the missile flags to either mark the target, go to the target, or run the missile retarget function
14	CltHitSubMissile1 CltHitSubMissile2	HitFreezingArrow - Create the “CltHitSubMissile1” sub missile normally, and create “CltHitSubMissile2” sub missile in a random rotation
15		Do nothing
16	cHitPar1 ProgOverlay	HitChainLightning - Create duplicate of this missile if there is a valid unit in range and there are still enough chain hits. Add an overlay on the target unit.
17		Do nothing
18	cHitPar1 cHitPar2 cHitPar3 CltHitSubMissile1 CltHitSubMissile2 CltHitSubMissile3 CltHitSubMissile4	HitMeteorCenter - Create a ring of “CltHitSubMissile1” sub missiles where “cHitPar1” controls the density. Create a “CltHitSubMissile2” sub missile where the range is controlled by the linked skill’s “Param3” and “Param4” values from the skills.txt file, and also set the missile’s light radius value to 12. Create a ring of “CltHitSubMissile3” sub missiles where “cHitPar2” controls the density. Create a ring of “CltHitSubMissile4” sub missiles where “cHitPar3” controls the density.
19	CltHitSubMissile1 CltHitSubMissile2	HitMonBliz - Randomly choose between creating one of the sub missiles
20		Do nothing
21		Do nothing
22		Do nothing
23		Do nothing
24	CltHitSubMissile1	HitBoneSpear - Create a sub missile at the target location
25	cHitPar1 cHitPar2 CltHitSubMissile1	HitLightningFury - Create a sub missile per enemy found in an area, where the radius and the maximum number of possible missiles to spawn are controlled by the parameters
26	cHitPar1 cHitPar2 CltHitSubMissile1 HitSubMissile1	HitFistOfHeavensDelay - If there is no “HitSubMissile1” sub missile, then do nothing. Otherwise, create a sub missile per enemy found in an area, where the radius and the maximum number of possible missiles to spawn are controlled by the parameters
27		nullptr
28	cHitPar1 CltHitSubMissile1	HitMonsterRancidGasPotion - Create a disc of sub missiles with a specified density. Each sub missile will use its “Param1” field to define its velocity.

	Param1	
29	cHitPar1 CltHitSubMissile1	HitGrimWard - Create a sub missile with a specified duration and force its direction to be the same as the missile that created it
30	cHitPar1 CltHitSubMissile1	HitFrozenOrb - Create a disc of sub missiles with a specified density.
31		HitIceBreak - Create a missile with a forced animation rate value of 1, depending on the missile class used. • If “Missile” equals “icebreaksmall” then create “icebreaksmallmelt” • If “Missile” equals “icebreakmedium” then create “icebreaklargemelt” • If “Missile” equals “icebreaklarge” then create “icebreaklargemelt” • If “Missile” equals “catapult cold explosion” then create “icebreaklargemelt”
32	cHitPar1 cHitPar2 CltHitSubMissile1 CltHitSubMissile2 ProgOverlay	HitFirehead - Create the “CltHitSubMissile1” sub missile at the target location and create a ring of “CltHitSubMissile2” sub missiles where the parameters control the ring radius and density. Also add an overlay on the source unit.
33	cHitPar1 CltHitSubMissile1	HitFlyingRocks - Create a random number of sub missiles in a defined area radius controlled by “cHitPar1”
34		HitSmithDoNotDraw - Make the source unit invisibile
35		Do nothing
36	Param1	HitHellMeteor - Do nothing is colliding with a wall. Otherwise, create a disc of randomly selected missiles. The random missiles chosen are either “firewall”, “firesmall”, or “firemedium”. The missile’s “Param1” field controls the disc radius. The duration of each missile created is calculated with the following: Range = 25 + RANDOM(0, 25) - 12
37		HitRadHandOfGod - Create a random number of “radamentholybolt” missiles in random positions in a radius
38		HitTaintedSunFlash - Create a “taintedsunball” missile
39		HitQueenDeathGlob - Randomly create 1 of the next 2 missile’s defined in the missile.txt file after the missile that uses this function. For example, if “Missile” equals “queendeathglob” (ID = 354) and it uses this function, then randomly choose to create “queendeathsplat1” (ID = 355) or “queendeathsplat2” (ID = 356).
40		HitHealingBolt - Determine that the target is an allied unit
41		HitDurielDeathRock - Create 3 “durieldeathdebris” missiles randomly in a radius value of 6. Create a “durieldeathsmoke” missile at the previous missile’s location.
42		HitSoulStoneFragment1 - Create a “hffragment2” missile on the target unit
43		HitSoulStoneFragment2 - Create a “hffragment3” missile on the target unit, with 0 velocity, and a random delay between 10 to 45 frames.
44	CltHitSubMissile1	HitCreateNextMissile - Create a sub missile and set its direction to match the old missile’s path
45		Do nothing
46	cHitPar1 cHitPar2 HitSubMissile1	HitCatapultChargedBall - Create a disc of sub missiles where the number of missiles created is controlled by the following formula: [“cHitPar1”] + [“cHitPar2”] * ([Missile current level] - 1)
47	cHitPar1 cHitPar2 CltHitSubMissile1 CltHitSubMissile2	HitCatapultSpikeBag - Create “CltHitSubMissile1” sub missiles using the lob function where “cHitPar1” and “cHitPar2” are used to calculate the number of missiles in the following formula: MissileCount = [“cHitPar1”] + [“cHitPar2”] 8 ([Missile current level] - 1). The radius is this missile lob function is determined by doing MissileCount / 4. Also create “CltHitSubMissile2” at the location of the old missile.
48		HitCatapultCold - Create a “freezingarrowexp1” (ID = 88) missile in the center of the previous missile location. Also create 8 different “freezingarrowexp2” (ID = 89) missiles ejected in 8 different directions. Also create 8 different “catapult cold explosion” (ID = 417) missiles that have a randomized range and velocity and use the lob function to launch these missiles which an initial Z offset equal to 30.
49		Do nothing
50	cHitPar1 cHitPar2 CltHitSubMissile1 CltHitSubMissile2	HitCatapultMeteor - Run the spray rock function to spawn different missiles. First create multiple “moltenboulder-flyingrocks” (ID = 456) missiles where the “cHitPar1” field controls the number of missiles to spawn and the targeted radius distance to spawn these missiles is equal to this value multiplied by 2. The “cHitPar2” field controls these missiles’ velocity. The “CltHitSubMissile1” sub missile is never spawned, but these “moltenboulder-flyingrocks” will not spawn unless this field has a value. Next create a ring of 18 “CltHitSubMissile2” sub missiles.
51	cHitPar1 cHitPar2 CltHitSubMissile1	HitLightJav - Create a disc of the sub missiles where “cHitPar1” controls the number of sub missiles to spawn and “cHitPar2” acts as a Boolean Field where if enabled, it will cause these spawned sub missiles to have a randomized path.
52	cHitPar1 cHitPar2 CltHitSubMissile1 CltHitSubMissile2 CltHitSubMissile3	HitMoltenBoulder - If hitting the target or a monster with the “large” flag enabled, then create “CltHitSubMissile1” sub missile. Also run the spray rock function to spawn different missiles. First create multiple “moltenboulder-flyingrocks” (ID = 456) missiles where the “cHitPar1” field controls the number of missiles to spawn and the targeted radius distance to spawn these missiles is equal to this value multiplied by 2. The “cHitPar2” field controls these missiles’ velocity. The “CltHitSubMissile2” sub missile is never spawned, but these “moltenboulder-flyingrocks” will not spawn unless this field has a value. Next create a ring of 18 “CltHitSubMissile3” sub missiles.
53	cHitPar1 cHitPar2 cHitPar3 CltHitSubMissile1	CreateRollingBoulder - Create the sub missile that bounces, where “cHitPar1” controls the number of bounces, “cHitPar2” controls the bounce dampening percentage, and “cHitPar3” controls the number of steps, or the lifetime remaining in ticks (Minimum value equal to 1).
54	CltHitSubMissile1	HitVineTrail - Create the sub missile and force it to get the same direction as the previous missile
55	CltHitSubMissile1	HitDebris - Create all 3 sub missiles at the source unit location

	CltHitSubMissile2 CltHitSubMissile3	
56	CltHitSubMissile1	HitRecycleVine - Create the sub missile at the source unit location and force it to get the same direction as the previous missile
57	cHitPar1 cHitPar2 CltHitSubMissile1	HitBaalSpawn - Create a disc ring of sub missiles where the parameters control the radius and density of the ring, respectively
58		Do nothing
59		Do nothing
60	cHitPar1 CltHitSubMissile1 CltHitSubMissile2	HitNihlathak1 - Create the “CltHitSubMissile1” sub missile at the location of the previous missile. Create the “CltHitSubMissile2” sub missile at the same position, but “cHitPar1” controls the z offset of the sub missile.
61		HitWorldstoneShake - Stop the camera shake
62	cHitPar1 CltHitSubMissile1	HitBaalRandomBolts - Create the sub missile where the range is randomized by the parameter (minimum value equals 1) and the target location is also randomized by this range value
63	cHitPar1 CltHitSubMissile1 Param1 Param2	HitBaalTauntPoison - Create a disc of sub missiles where “cHitPar1” controls the ring density (minimum value equals 1). Each sub missile will use its “Param1” and “Param2” fields to define their velocities
64	cHitPar1 CltHitSubMissile1	HitBladeFury - Create a disc of sub missiles with the parameter controlling the ring density (minimum value equals 1)

pSrvDoFunc - Uses an ID value to select a specialized function for the missile’s behavior while active every frame on the server side

Code	Parameters	Description
0 (or empty)		Do nothing
1		MissileDoArrow - Perform the standard missile move function. This is called in most of the other functions
2	SrvCalc1 SubMissile1	MissileDoPoisonJavelin - Create a sub missile with a specified number of subloops
3		StillMissileKludge - Check the missile velocity to determine whether or not to set the collision mask to Missile
4		Do nothing
5	SubStart SubStop	Firewall - Create an animation sequence for the missile, set the collision mask to Missile
6	SubMissile1	MissileMakeFirewall - Create sub missiles based on the count that was passed to this missile
7	Param1	MissileGuidedArrow - Use the parameter to control the retarget time on the unit
8	Param1 Param2 Param3 SubMissile1	MissileMonBliz - Create multiple sub missiles, using the parameters as the spawn radius, spawn frequency, and level divisor
9	SubMissile1	MissileMakeBatLightning - Create a sub missile based on the missile’s path moved
10	SubMissile1	MissileBlizzard - Create multiple sub missiles, use the linked skill’s “calc1” field for the radius, and the “calc2” field for the sub missile spawn frequency
11	Param1 Param2 Param3	FingerMageSpider - Shoot a missile that can be retargeted. Uses the parameters to control the periodic delay between retargeting, the maximum distance to retarget, and the distance delta for where to retarget.
12	SubMissile1	MissileDiabWallMaker - Shoot a missile and have that missile create sub missiles as it is moving
13		MissileBoneWallMaker - Shoot a missile and have that missile create pet summons as it is moving, using the linked skill’s “pettype” field from skills.txt
14	Param1 Param2	MissileDoGrimWard - Shoot a missile where every periodic delay, do a function from the “srvdofunc” field in the skills.txt file. “Param1” controls the periodic frame delay, and “Param2” controls which function code number to use.
15	Param1 Param2 SubMissile1	MissileFrozenOrb - Shoot a missile and have that missile shoot sub missiles in different directions as it is moving. “Param1” controls the sub missile spawn rate and “Param2” controls the change in direction per sub missile spawn.
16	Param1 Param2	MissileFrozenOrbNova - Shoot a missile in a set direction. “Param1” controls the delay until the missile moves and “Param2” controls the periodic frame delay for updating the missile’s target path
17	Param1 Param2 Param3 Param4 Param5 SubMissile1	MissileDoCairnStones - Shoot sub missiles in a radius and then after a delay, create a portal to another level. Uses parameters to control the delays between creating sub missiles, the radius to create the sub missiles, which level ID to link the portal to, and the delay before creating the portal.
18	Param1 Param2 Param3	DoTowerChestSpawner - Open the chest object to spawn items, and create random gold piles in an area
19		DoRadamentDeath - Use the Paladin Redemption skill function on nearby corpses in an area
20		MissileAttachUntilDead - Shoot the missile and keep it attached to the source until. If the source unit dies, then kill the missile.
21	SubMissile1	MissileDoDistraction - Create a sub missile and run the MissileAttachUntilDead function
22	Param1 SubMissile1	ProcessTrailJav - Shoot the missile and have it create 2 sub missiles every frame while it is moving
23	Param1 SubMissile1	ProcessSucFireBall - Create a sub missile every frame while the missile is moving, where the parameter controls the sub missile’s subloops
24	Param1 SubMissile1	Duplicate of function 23
25	SubMissile1	MissileEruption - Shoot the missile and have it create sub missiles in a radius at a periodic rate, which is controlled by the link skill’s “calc1” and “calc2” fields.

26	Param1 SubMissile1	ProcessVines - Shoot the missile and have it create sub missiles at a periodic rate, which is controlled by the parameter
27	Param1 Param2	MissileTornado - Shoot the missile and have it deal damage at a periodic rate in a radius, which is controlled by the parameters or by the linked skill's "calc4" and "aurarangecalc" fields
28	Param1 Param2 Param3 Param4 Param5 SubMissile1	ProcessVolcano - Shoot the missile and have it periodically create sub missiles in a lobbing pattern. Use parameters to control the periodic frame delay for spawning sub missiles, the radius to spawn the sub missiles, the start and stop frame count for when to spawn and stop spawning sub missiles, and the lob start value for the sub missiles
29	Param1	ProcessRecycleDelay - Shoot the missile and after a certain delay, process any life steal or mana steal, based on the linked skill's "calc1" and "calc2" fields
30	Param1 Param2 SubMissile1	MissileRabiesPlague - Shoot the missile, have it attached to the source unit, and have it periodically spawn sub missiles in a radius
31	SubMissile1	ProcessMakePerpMissiles - Shoot the missile and have it create 2 sub missiles that are fired in perpendicular directions
32	SubMissile1	MissileTigerFuryPath - Shoot the missile using the MissileGuidedArrow function and have it create a sub missile
33	Param1 Param2 SubMissile1	ProcessRecycleManaDelay - Shoot the missile and after a certain delay, process any mana steal, based on the linked skill's "calc1" field, and create a sub missile after another certain delay
34	Param1 Param2 Param3 Param4 SubMissile1 SubMissile2	ProcessBaalTaunt - Randomly choose one of the 4 parameters to select a delay and randomly spawn one of the sub missiles
35	Param1	MissileDoChaosIce - Shoot the missile and have it repath in a perpendicular direction after a periodic delay
36		MissileDoBaalDeathControl - Shoot the missile and spawn the Tyrael unit
37	SubMissile1 SubMissile2 SubMissile3	ProcessMissileDelayed - Shoot the missile and have it create 1 of each sub missile

pSrvHitFunc - Uses an ID value to select a specialized function for the missile's behavior when hitting something on the server side

Code	Parameters	Description
0 (or empty)		Do nothing
1	sHitPar1	RadialFireDamage - Deal elemental damage in an area where the parameter controls the damage radius
2	sHitPar1 sHitPar2 HitSubMissile1	HitPlagueJavelin - Kill the missile while dealing its damage, and also deal radial poison damage in an area by creating sub missiles in a defined radius for a specified number of loops
3	sHitPar1	NoTargetRadialDamage - Deal the missile damage in a defined area radius
4	sHitPar1 HitSubMissile1 HitSubMissile2 HitSubMissile3 HitSubMissile4	HitCreateMissile - Determine whether to kill this missile on hit or not, and then create 1 of each hit sub missile
5	sHitPar1 HitSubMissile1	HitDoNova - Create a certain number of sub missiles and shoot them outwards in an equalized distance from the location of this missile
6	sHitPar1 sHitPar2	HitSummon - Create a monster in a specific starting mode at the missile's location. The "sHitPar1" field controls the monster ID to use, and the "sHitPar2" field controls which mode to set on the monster.
7	sHitPar1 sHitPar2 sHitPar3 dParam1	HitHolyBolt - Determine whether to impact allies, how that damage is modified by the "dParam1" field based on the unit type hit, and if the missile should be killed on hitting an allowed unit
8		HitBlaze - The missile will deal damage if the source unit has the "blaze" state
9	sHitPar1 sHitCalc1 HitSubMissile1	HitImmolationArrow - Deal radius damage and create sub missiles with specified range values in a defined radius. If there is a freeze length or cold length from the elemental damage, then set it to 0.
10		HitGuidedArrow - Control if the missile should be redirected on a target when possible, or if it should be destroyed
11	sHitPar1 HitSubMissile1 HitSubMissile2 HitSubMissile3 HitSubMissile4	HitCreateMissileNoDmg - Kill this missile on hit and create 1 of each hit sub missile
12	sHitPar1	HitChainLightning - Create duplicate of this missile if there is a valid unit in range and there are still enough chain hits
13	sHitPar1 sHitPar2	HitGlacialSpike - Deal elemental damage in a specified radius with a specified elemental duration, if applicable
14	sHitPar1 sHitPar2 HitSubMissile1	HitMeteorCenter - Deal damage in a specified radius and create a ring of hit sub missiles (controlled by a count) and define the duration of these sub missiles based on the linked skill's "Param3" and "Param4" values
15	HitSubMissile1	HitSpiderLay - Kill the missile and create a sub missile at the location
16		HitSpiderGoo - Apply the state from the linked skill's "auratargetstate" field with a duration defined by the linked skill's "calc4" field

17		HitHowl - Use the linked skill's "auratargetstate", "calc2", "calc3", and "Param2" fields to apply a state (with a defined level and duration) to units in a defined radius. Also call the AIScare function.
18		HitShout - Kill the missile and call the AddShout function on hit allied units, which applies the "aurastate" state defined in the linked skill
19		HitFingerMageSpiderHit - Kill the missile and apply the "auratargetstate" defined in the linked skill
20	sHitPar1 sHitPar2 HitSubMissile1	HitLightningFury - Kill the missile and create a specified number of sub missiles in a radius, per unit targeted in that radius
21		HitBattleCry - Kill the missile and apply the "auratargetstate" state on the target unit, defined in the linked skill
22	sHitPar1 sHitPar2 HitSubMissile1	HitFistOfHeavensDelay - Kill the missile and create a specified number of sub missiles in a defined radius, per filtered unit targeted in that radius
23	sHitPar1	HitMissileSkill - Kill the missile and do a function from the "srvdofunc" field in the skills.txt file, where the "sHitPar1" field controls the function ID to use
24	sHitPar1	RadialMissileDamage - Kill the missile and damage in an area controlled by a defined radius
25	sHitPar1 HitSubMissile1	HitMonsterRancidGasPotion - Kill the missile and deal radial poison damage in an area by creating sub missiles in a defined radius for a specified number of loops
26	sHitPar1 HitSubMissile1	HitGrimWardStart - Kill the missile and create a sub missile with a specified duration
27		HitKillMissile - Kill the missile
28		HitGrimWardScare - Kill the missile and apply the AiScare function to all monsters in a radius controlled by the "calc2" field from the linked skill with a distance and duration value controlled by the "Param5" and "Param6" values from the linked skill.
29	sHitPar1 HitSubMissile1	HitFrozenOrb - Kill the missile and create specified number sub missiles in a circular outwards direction from that location
30	sHitPar1 sHitPar2	RadialStunDamage - Kill the missile and do stun damage in a defined radius with a defined stun length
31		HitFirehead - Kill the missile and deal elemental damage directly to the target unit's life, ignoring resistances
32	Param4	MissileHitCairnStones - Create a portal linked to a specified level area ID
33		HitTowerChest - Set the dungeon room to untile
34		Do nothing
35		OrbMistHit - Kill the missile. Set the missile's tracked object's dungeon room to untile and set that object's mode from Neutral to Operating.
36	HitSubMissile1	HitCreateNextMissile - Kill the missile and create a sub missile with parameters copied over
37		HitBladeCreeper - If this hits a target unit then deal damage
38	sHitPar1 sHitPar2 HitSubMissile1	HitCatapultChargedBall - Kill the missile and create a specified number of sub missiles in a ring
39		HitImpSpawnMonsters - Kill the missile and spawn an appropriate monster, based on the allowed monsters in the area level
40	sHitPar1 sHitPar2 HitSubMissile1	HitCatapultSpikeBag - Kill the missile and create a specified number of sub missiles that are lobbed outwards in a radius
41		Do nothing
42		Do nothing
43		HitHealing - Use the linked skill's physical damage as the amount of healing done to the target unit. Determine whether to kill the missile or not, based on the "CollideKill" field
44	sHitPar1	RadialDamage - Kill the missile and deal damage in a specified radius
45	sHitPar1 sHitPar2 HitSubMissile1	HitLightJav - Kill the missile and create a number of sub missiles in a disc. Determine whether these sub missiles created should use a random path or not.
46		Do nothing
47	sHitPar1 sHitPar2 HitSubMissile1	BoulderExplode - Kill the missile and create an explosion with a defined radius and a ring of sub missiles with a defined count of missiles
48	HitSubMissile1	CreateRollingBoulder - Kill the missile and create a sub missile
49		Do nothing
50	sHitPar1	HitPlagueVines - If the range of the missile minus the "sHitPar1" parameter is greater than the missile's current frame, then deal damage
51	HitSubMissile1 HitSubMissile2 HitSubMissile3	HitDebris - Kill the missile and create the 3 sub missiles, if possible
52	sHitPar1 HitSubMissile1	HitBladeFury - Kill the missile and spawn sub missiles in multiple directions depending on the "sHitPar1" value
53		HitRabiesContagion - Get the elemental duration from the linked skill and compare that with the duration of the missile to determine to kill the missile or create a new one, depending on the target having the linked skill's "auratargetstate" state
54		HitBaalSpawnMonsters - Kill the missile and spawn a monster in Neutral mode
55	sHitPar1	HitBaalInferno - Drain a percentage of the target's mana (from 0 to 100) and deal damage
56	sHitPar1 HitSubMissile1	HitArmageddon - Kill the missile, deal damage in an area with a defined radius, and create a sub missile
57		MissileHitBaalDeathControl - Create the Tyrael unit
58	sHitPar1 HitSubMissile1	HitBaalRandomBolts - Kill the missile and create a sub missile targeting a random randomized location range controlled by "sHitPar1"
59	sHitPar1 HitSubMissile1	HitBaalTauntPoison - Kill the missile and create an inner and outer disc of sub missiles with a specified count (using "sHitPar1") of missiles and with incremental specified velocities (the sub

	Param1 Param2	missile will use its “Param1” and “Param2” fields)

pSrvDmgFunc - Uses an ID value to select a specialized function that gets called before damaging a unit on the server side

Code	Parameters	Description
0 (or empty)		Default
1	DmgCalc1	DamageFireArrow - Converts a percentage of physical damage to elemental damage per level
2	dParam1	DamageIceArrow - Converts the Cold Length value to Freeze Length. Uses the parameter value as a percentage increase the Freeze Length
3	dParam1	DamageFireWall - Uses the parameter as a random chance to use an attack result Soft Hit flag
4		DamageIceBlast - Converts the Cold Length value to the Freeze Length value
5	dParam1 dParam2	DamageBlessedHammerNew - Uses “dParam1” as a damage percent multiplier against Undead type monsters. Uses “dParam2” as a damage percent multiplier against Demon type monsters.
6		DamageNoItem - If the source unit is not a Mercenary, then set the target unit to be unable to drop items when it dies.
7	dParam1	DamageWarCry - Uses the parameter as the Stun Length. If the parameter equals 0, then use the source unit’s related skill’s “Calc4” field as the Stun Length value. If the skill is null, then use the missile’s default skill’s Param1 & Param2 values.
8	ProgOverlay	DamageEruption - Create an overlay on the target
9	dParam1	DamageTwister - Uses the parameter as the Stun Length, and sets the damage Hit Class layer to Stun Layer. If the parameter equals 0, then use the source unit’s related skill’s “Calc2” field as the Stun Length.
10		DamageFreeze - Checks that the missile has stats and then sets the Freeze Length value to equal the Cold Length value.
11		DoRabiesDamage - Checks the remaining poison duration on the target unit and if it is less than 10, it will use the linked skill’s Elemental damage and duration length calculation (See skills.txt) to apply a new poison.
12		sDamageLightningJavelin
13	dParam1 dParam2	DamageBlessedHammerOld - Uses the source units physical damage percent stat as a percentage modifier for the missile’s damage, and then calls the “DamageBlessedHammerNew” function (See function code 5)
14	dParam1 dParam2	DamageMoltenBoulder - Uses “dParam2” as the percent chance to knockback the target unit. Uses “dParam1” to control how this percent chance is modified. Also relies on the “small” and “large” fields from the monstats2.txt file. • If the target unit is a player and “dParam1” is greater than or equal to 1, then set the knockback chance to “dParam2” • If “dParam1” is less than 1 and the monster is a small size, then set the knockback chance to “dParam2” • If “dParam1” equals 1 and the monster is a small size, then set the knockback chance to “dParam2” * 2 • If “dParam1” equals 1 and the monster is a large size, then set the knockback chance to “dParam2” • If “dParam1” is greater than 1 and the monster is a small size, then set the knockback chance to “dParam2” * 3 • If “dParam1” is greater than 1 and the monster is a large size, then set the knockback chance to “dParam2” * 2 • If “dParam1” is greater than 1 and the monster is not a small or large size, then set the knockback chance to “dParam2”
15	sHitPar2 dParam1	DamageHolyBolt - Uses “dParam1” as a percent damage multiplier for the total elemental damage, depending on the use case of “sHitPar2” • If the target unit is a monster ○ If “sHitPar2” equals 0, then do not modify the damage ○ If “sHitPar2” equals 1, then only modify the damage if the monster is an Undead type ○ If “sHitPar2” equals 2, then only modify the damage if the monster is a Demon type ○ If “sHitPar2” equals 3, then only modify the damage if the monster is an Undead or Demon type • If the target unit is a player ○ If “sHitPar2” equals 0, then modify the damage ○ If “sHitPar2” equals 0, then do not modify the damage

SrvCalc1 - Numeric calculation field. Used as a parameter for the “pSrvDoFunc” field.
Param1 (to Param5) - Integer field. Used as a parameter for the “pSrvDoFunc” field.
CltCalc1 - Numeric calculation field. Used as a parameter for the “pCltDoFunc” field.
CltParam1 (to CltParam5) - Integer field. Used as a parameter for the “pCltDoFunc” field.
SHitCalc1 - Numeric calculation field. Used as a parameter for the “pSrvHitFunc” field.
sHitPar1 (to sHitPar3) - Integer field. Used as a parameter for the “pSrvHitFunc” field.
CHitCalc1 - Numeric calculation field. Used as a parameter for the “pCltHitFunc” field.
cHitPar1 (to cHitPar3) - Integer field. Used as a parameter for the “pCltHitFunc” field.
DmgCalc1 - Numeric calculation field. Used as a parameter for the “pSrvDmgFunc” field.
dParam1 & dParam2 - Integer field. Used as a parameter for the “pSrvDmgFunc” field.

Vel - The baseline velocity of the missile, which is the speed at which the missile moves in the game world. This is measured by distance in pixels traveled per frame.

MaxVel - The maximum velocity of the missile. If the missile's current velocity increases (based on other fields), then this field controls how high the velocity is allowed to go.
VelLev - Adds extra velocity based on the caster unit's level. Each level gained beyond level 1 will add this value to the baseline "Vel" field.
Accel - Controls the acceleration of the missile's movement. A positive value will increase the missile's velocity per frame. A negative value will decrease the missile's velocity per frame. The bigger positive or negative values will cause the velocity to change faster per frame.

Range - Controls the baseline duration that the missile will exist for after it is created. This is measured in frames where 25 Frames = 1 second.
LevRange - Adds extra duration based on the caster unit's level. Each level gained beyond level 1 will add this value to the baseline "Range" field.

Light - Controls the missile's Light Radius size (measured in grid sub-tiles)
Flicker - If greater than 0, then every 4th frame while the missile is active, the Light Radius will randomly change in size between base size to its base size plus this value (measured in grid sub-tiles)
Red - Controls the red color value of the missile's Light Radius (Uses a value from 0 to 255)
Green - Controls the green color value of the monster's Light Radius (Uses a value from 0 to 255)
Blue - Controls the blue color value of the monster's Light Radius (Uses a value from 0 to 255)

InitSteps - The number of frames the missile needs to be alive until it becomes visible on the game client. If the missile's current duration in frame count is less than this value, then the missile will appear invisible.
Activate - The number of frames the missile needs to be alive until it becomes active. If the missile's current duration in frame count is less than this value, then the missile will not collide.
LoopAnim - Boolean Field. If equals 1, then the missile's animation will repeat once the previous animation finishes. If equals 0, then the missile's animation will only play once, which can cause the missile to appear invisible at the end of the animation, but it will still be alive.
CelFile - Defines which DCC missile file to use for the visual graphics of the missile
animrate - Controls the visual speed of the missile's animation graphics. The overall missile animation rate is calculated as the following: 256 * ["animrate"] / 1024
AnimLen - Defines the length of the missile's animation in frames where 25 Frames = 1 second. This field can sometimes be used to calculate the missile animation rate, depending on the missile function used.
AnimSpeed - Controls the visual speed of the missile's animation graphics on the client side (Measured in 16ths, where 16 equals 1 frame per second). This can be overridden by certain missile functions.
RandStart - If this value is greater than 0, then the missile will start at a random frame between 0 and this value when it begins its animation.
SubLoop - Boolean Field. If equals 1, then the missile will use a specific sequence of its animation while it is alive, depending on its creation. If equals 0, then the missile will not use a sequenced animation.
SubStart - The starting frame of the sequence animation. This requires that the "SubLoop" field is enabled.
SubStop - The ending frame of the sequence animation. After reaching this frame, then the sequenced animation will loop back to the "SubStart" frame. This requires that the "SubLoop" field is enabled.

CollideType - Defines the missile's collision type, which controls what units, objects, or parts of the environment that the missile can impact

Code	Description
0	No collision
1	Player unit collision
2	Monster unit collision
3	Player and Monster unit collision
4	No collision
5	Monster unit collision
6	Barrier collision, such as doors
7	Missile collision
8	Player, Monster, Barrier, and Wall collision

CollideKill - Boolean Field. If equals 1, then the missile will be destroyed when it collides with something. If equals 0, then the missile will not be destroyed when it collides with something.
CollideFriend - Boolean Field. If equals 1, then the missile can collide with friendly units, including the caster. If equals 0, then the missile will ignore friendly units.
LastCollide - Boolean Field. If equals 1, then the missile will track the last unit that it collided with, which is useful for making sure the missile does not hit the same unit twice. If equals 0, then ignore this.
Collision - Boolean Field. If equals 1, then the missile will have a missile type path placement collision mask when it is initialized or moved. If equals 0, then the missile will have no placement collision mask when it is created or moved.
ClientCol - Boolean Field. If equals 1, then the missile will check collision on the client, depending on the missile's "CollideType" field. If equals 0, then ignore this.
ClientSend - Boolean Field. If equals 1, then the server will create the missile on the client. This can be used when reloading area levels or transitioning units between areas. If equals 0, then ignore this.
NextHit - Boolean Field. If equals 1, then the missile will use the next delay. If equals 0, then ignore this.
NextDelay - Controls the delay in frame length until the missile is allowed to hit the same unit again. This field relies on the "NextHit" field being enabled.

xoffset & yoffset & zoffset - Specifies the X, Y, and Z location coordinates (measured in pixels) to offset to visually draw the missile based on its actual location. This will only offset the visual graphics of the missile, not the missile itself. The Z axis controls the visual height of the missile.
Size - Defines the diameter in sub-tiles (for both the X and Y axis) that the missile will occupy. This affects how the missile will collide with something or how the game will handle placement for the missile.

SrcTown - Boolean Field. If equals 1, then the missile will be destroyed if the caster unit is located in an act town. If equals 0, then ignore this.
CltSrcTown - If this value is greater than 0 and the "LoopAnim" field is disabled, then this field will control which frame to set the missile's animation when the player is in town. This value gets subtracted from the "AnimLen" value to determine the frame to set the missile's animation.
CanDestroy - Boolean Field. If equals 1, then the missile can be attacked and destroyed. If equals 0, then the missile cannot be attacked.

ToHit - Boolean Field. If equals 1, then this missile will use the caster's Attack Rating stat to determine if the missile should hit its target. If equals 0, then the missile will always hit its target.
AlwaysExplode - Boolean Field. If equals 1, then the missile will always process an explosion when it is killed, which can use a server hit function (See "pSrvHitFunc") and can use the "HitSound" and "ExplosionMissile" fields on the client side. If equals 0, then the missile will only rely on proper collision hits to process an explosion.
Explosion - Boolean Field. If equals 1, then the missile will be classified as an explosion which will make it use different handlers for finding nearby units and dealing damage. If equals 0, then ignore this.

Town - Boolean Field. If equals 1, then the missile is allowed to be alive when in a town area. If equals 0, then the missile will be immediately destroyed when located within a town area.
NoUniqueMod - Boolean Field. If equals 1, then the missile will not receive bonuses from Unique monster modifiers. If equals 0, then the missile will receive bonuses from Unique monster modifiers.
NoMultiShot - Boolean Field. If equals 1, then the missile will not be affected by the Multi-Shot monster modifier. If equals 0, then the missile will be affected by the Multi-Shot monster modifier.
Holy - Controls a bit field flag where each value is a code to allow the missile to damage a certain type of monster

Code	Description
0	Damage all units
1	Only damage Undead
2	Only damage Demons
4	Only damage Beasts

CanSlow - Boolean Field. If equals 1, then the missile can be affected by the “slowmissiles” state (See states.txt). If equals 0, then the missile will ignore the “slowmissiles” state.
ReturnFire - Boolean Field. If equals 1, then missile can trigger the Sorceress Chilling Armor event function. If equals 0, then this missile will not trigger that function.
GetHit - Boolean Field. If equals 1, then the missile will cause the target unit to enter the Get Hit mode (GH), which acts as the hit recovery mode. If equals 0, then ignore this.
SoftHit - Boolean Field. If equals 1, then the missile will cause a soft hit on the unit, which can trigger a blood splatter effect, hit flash, and/or a hit sound. If equals 0, then ignore this.
KnockBack - Controls the percentage chance (out of 100) that the target unit will be knocked back when hit by the missile
Trans - Controls the alpha mode for how the missile is displayed, which can affect transparency and blending

Code	Description
0	No Transparency
1	Black Alpha Transparency
2	White Alpha Transparency

Pierce - Boolean Field. If equals 1, then allow the Pierce modifier function to work with this missile. If equals 0, then do not allow Pierce to work with this missile.

MissileSkill - Boolean Field. If equals 1, then the missile will look up the skill that created it and use that skill’s damage instead of the missile damage. If equals 0, then ignore this.
Skill - Links to the “skill” field from the skills.txt file. This will look up the specified skill’s damage and use it for the missile instead of using the missile’s defined damage.

ResultFlags - Controls different flags that can affect how the target reacts after being hit by the missile. Uses an integer value to check against different bit fields by using the “&” operator. For example, if the value equals 5 (binary = 101) then that returns true for both the 4 (binary = 100) and 1 (binary = 1) bit field values.

Bit Field Value	Binary Equivalent Value	Description
1	0000000000000001	Hit
2	0000000000000010	Death
4	0000000000000100	Get Hit
8	0000000000001000	Knockback
16	0000000000010000	Block
32	0000000000100000	No Passive
128	0000000010000000	Dodge
256	0000000100000000	Avoid
512	0000001000000000	Evade
4096	0001000000000000	Ignore Friendly
8192	0010000000000000	Double Damage
16384	0100000000000000	Soft Hit
32768	1000000000000000	Two Hand-to-Hand Block

HitFlags - Controls different flags that can affect the damage dealt when the target is hit by the missile. Uses an integer value to check against different bit fields by using the “&” operator. For example, if the value equals 6 (binary = 110) then that returns true for both the 4 (binary = 100) and 2 (binary = 10) bit field values.

Bit Field Value	Binary Equivalent Value	Description
1	00000000001	Do not add physical damage
2	00000000010	Do not add any damage
4	00000000100	No Life Steal
8	00000001000	No Mana Steal
16	00000010000	No Stamina Steal
32	00000100000	Use Source Damage
128	00010000000	No Triggered Events
256	00100000000	Bypass Undead
512	01000000000	Bypass Demons
1024	10000000000	Bypass Beasts

HitShift - Controls the percentage modifier for the missile’s damage. This value cannot be less than 0 or greater than 8. This is calculated in 256ths, where 8=256/256, 7=128/256, 6=64/256, 5=32/256, 4=16/256, 3=8/256, 2=4/256, 1=2/256, and 0=1/256.
ApplyMastery - Boolean Field. If equals 1, then apply the caster’s elemental mastery bonus modifiers to the missile’s elemental damage. If equals 0, then ignore this.
SrcDamage - Controls how much of the source unit’s damage should be added to the missile’s damage. This is calculated in 128ths and acts as a percentage modifier for the source unit’s damage that added to the missile. If equals -1 or 0, then the source damage is not included.
Half2HSrc - Boolean Field. If equals 1 and the source unit is currently wielding a 2-Handed weapon, then the source damage (see “SrcDamage”) is reduced by 50%. If equals 0, then ignore this.
SrcMissDmg - If the missile was created by another missile, then this controls how much of the source missile’s damage should be added to this missile’s damage. This is calculated in 128ths and acts as a percentage modifier for the source missile’s damage that added to this missile. If equals 0, then the source damage is not included.

MinDamage - Minimum baseline physical damage dealt by the missile

MinLevDam1 (to MinLevDam5) - Controls the additional minimum physical damage dealt by the missile, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field.
MaxDamage - Maximum baseline physical damage dealt by the missile
MaxLevDam1 (to MaxLevDam5) - Controls the additional maximum physical damage dealt by the missile, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field.
DmgSymPerCalc - Calculation Field. Determines the percentage increase to the physical damage dealt by the missile based on specified skill levels.

EType - Defines the type of elemental damage dealt by the missile. If this field is empty, then the related elemental fields below will not be used.

Code	Description
(empty)	None
fire	Fire
ltng	Lightning
mag	Magic
cold	Cold (Uses "ELen")
pois	Poison (Uses "ELen")
life	Life Drain
mana	Mana Drain
stam	Stamina Drain
stun	Stun (Uses "ELen")
rand	Randomly select between Fire, Lightning, Magic, Cold, or Poison
burn	Burning (Uses "ELen")
frze	Freeze (Uses "ELen")

EMin - Minimum baseline elemental damage dealt by the missile
MinELev1 (to MinELev5) - Controls the additional minimum elemental damage dealt by the missile, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field number.
EMax - Maximum baseline elemental damage dealt by the missile
MaxELev1 (MaxELev5) - Controls the additional maximum elemental damage dealt by the missile, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field.
EDmgSymPerCalc - Calculation Field. Determines the percentage increase to the elemental damage dealt by the missile based on specified skill levels.
ELen - The baseline elemental duration dealt by the missile. This is calculated in frame lengths where 25 Frames = 1 second. These fields only apply to appropriate elemental types with a duration.
ELevLen1 (to ELevLen3) - Controls the additional elemental duration added by the missile, calculated using the leveling formula between 3 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17 and beyond. These 3 level thresholds correlate to each field. These fields only apply to appropriate elemental types with a duration.

HitClass - Defines the missile's own hit class into the damage routines, mainly used for determining hit sound effects and overlays. This field only handles the hit class layers, so values beyond these defined bits are ignored. Uses an integer value to check against different bit fields by using the "&" operator. For example, if the value equals 6 (binary = 110) then that returns true for both the 4 (binary = 100) and 2 (binary = 10) bit field values.

Bit Field Value	Binary Equivalent Value	Description
16	00010000	Double Layer
32	00010100	Fire Layer
48	00011110	Cold Layer
64	01000000	Lightning Layer
80	01010000	Poison Layer
96	01100000	Stun Layer
112	01110000	Bash Layer
128	10000000	Thorns Layer
144	10010000	Sanctuary Layer
160	10100000	Silent Voice Layer
176	10110000	Goo Layer

NumDirections - The number of directions allowed by the missile, based on the DCC file used (see "CelFile"). This value should be within the power of 2, with a minimum value of 1 or up to a maximum value of 64.
LocalBlood - Boolean Field. If equals 1, then change the color of blood missiles to green. If equals 0, then keep the blood missiles colored the default red.

DamageRate - Controls the "damage_framerate" stat (Calculated in 1024ths), which acts as a percentage multiplier for the physical damage reduction and magic damage reduction stat modifiers, when performing damage resistance calculations. This is only enabled if the value is greater than 0.

TravelSound - Points to a "Sound" field defined in the sounds.txt file. Used when the missile is created and while it is alive.
HitSound - Points to a "Sound" field defined in the sounds.txt file. Used when the collides with a target.

ProgSound - Points to a "Sound" field defined in the sounds.txt file. Used for a programmed special event based on the client function.
ProgOverlay - Points to the "overlay" field defined in the Overlay.txt file. Used for a programmed special event based on the server or client function.

ExplosionMissile - Points to the "Missile" field for another missile. Used for the missile created on the client when this missile explodes.

SubMissile1 (to SubMissile3) - Points to the "Missile" field for another missile. Used for creating a new missile based on the server function used.
HitSubMissile1 (to HitSubMissile4) - Points to the "Missile" field for another missile. Used for a new missile after a collision, based on the server function used.
CltSubMissile1 (to CltSubMissile3) - Points to the "Missile" field for another missile. Used for creating a new missile based on the client function used.
CltHitSubMissile1 (to CltHitSubMissile4) - Points to the "Missile" field for another missile. Used for a new missile after a collision, based on the client function used.

misc.txt

Overview

This file controls the functionalities for miscellaneous type items, such as the non-weapons and non-armor items.

This file is loaded together with other similar files in the following order: weapons.txt, armor.txt, misc.txt
These combined files form the items structure. Technically these files share the same fields, but some fields are exclusive for specific item types, so they are not displayed in the data files that do not need them.

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

name - This is a reference field to define the item
version - Defines which game version to create this item (0 = Classic mode | 100 = Expansion mode)
compactsave - Boolean Field. If equals 1, then only the item's base stats will be stored in the character save, but not any modifiers or additional stats. If equals 0, then all of the items stats will be saved.
rarity - Determines the chance that the item will randomly spawn (1/#). The higher the value then the rarer the item will be. This field depends on the "spawnable" field being enabled, the "quest" field being disabled, and the item level being less than or equal to the area level. This value is also affected by the relative Act number that the item is dropping in, where the higher the Act number, then the more common the item will drop.
spawnable - Boolean Field. If equals 1, then this item can be randomly spawned. If equals 0, then this item will never randomly spawn.

speed - If the item type is an armor, then this will affect the Walk/Run Speed reduction when wearing the armor. If the item type is a weapon, then this will affect the Attack Speed reduction when wearing the weapon.
reqstr - Defines the amount of the Strength attribute needed to use the item
reqdex - Defines the amount of the Dexterity attribute needed to use the item

durability - Defines the base durability amount that the item will spawn with.
nodurability - Boolean Field. If equals 1, then the item will not have durability. If equals 0, then the item will have durability.

level - Controls the base item level. This is used for determining when the item is allowed to drop, such as making sure that the item level is not greater than the monster's level or the area level.
ShowLevel - Boolean Field. If equals 1, then display the item level next to the item name. If equals 0, then ignore this.
levelreq - Controls the player level requirement for being able to use the item

cost - Defines the base gold cost of the item when being sold by an NPC. This can be affected by item modifiers and the rarity of the item.
gamble cost - Defines the gambling gold cost of the item on the Gambling UI

code - Defines a unique 3 letter/number code for the item. This is used as an identifier to reference the item.
namestr - String Key that is used for the base item name

magic lvl - Defines the magic level of the item, which can affect how magical item modifiers that can appear on the item (See automagic.txt)
auto prefix - Automatically picks an item affix name from a designated "group" value from the automagic.txt file, instead of using random prefixes. This is only used when the item is Magical quality.

alternategfx - Uses a unique 3 letter/number code similar to the defined "code" fields to determine what in-game graphics to display on the player character when the item is equipped

normcode - Links to a "code" field to determine the normal version of the item
ubercode - Links to a "code" field to determine the Exceptional version of the item
ultracode - Links to a "code" field to determine the Elite version of the item

component - Determines the layer of player animation when the item is equipped. This uses a code referenced from the Composit.txt file.

Code	Description
0	Head
1	Torso
2	Legs
3	Right Arm
4	Left Arm
5	Right Hand
6	Left Hand
7	Shield
8	Special 1
9	Special 2
10	Special 3
11	Special 4
12	Special 5
13	Special 6
14	Special 7
15	Special 8
16	Do not display anything

invwidth & **invheight -** Defines the width and height of grid cells that the item occupies in the player inventory

hasinv - Boolean Field. If equals 1, then the item will have its own inventory allowing for the capability to socket gems, runes, or jewels. If equals 0, then the item cannot have sockets.
gemsockets - Controls the maximum number of sockets allowed on this item. This is limited by the item's size based on the "invwidth" and "invheight" fields. This also compares with the "MaxSock1", "MaxSock25" and "MaxSock40" fields from the ItemTypes.txt file.

gemapplytype - Determines which affect from a gem or rune will be applied when it is socketed into this item (See gems.txt)

Code	Description
0	Weapon
1	Armor or Helmet
2	Shield

flippyfile - Controls which DC6 file to use for displaying the item in the game world when it is dropped on the ground (uses the file name as the input)
invfile - Controls which DC6 file to use for displaying the item graphics in the inventory (uses the file name as the input)
uniqueinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Unique quality item (uses the file name as the input)
setinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Set quality item (uses the file name as the input)

useable - Boolean Field. If equals 1, then the item can be used with the right-click mouse button command (this only works with specific belt items or quest items). If equals 0, then ignore this.

stackable - Boolean Field. If equals 1, then the item will use a quantity field and handle stacking functionality. This can depend on if the item type is throwable, is a type of ammunition, or is some other kind of miscellaneous item. If equals 0, then the item cannot be stacked.
minstack - Controls the minimum stack count or quantity that is allowed on the item. This field depends on the “stackable” field being enabled.
maxstack - Controls the maximum stack count or quantity that is allowed on the item. This field depends on the “stackable” field being enabled.
spawnstack - Controls the stack count or quantity that the item can spawn with. This field depends on the “stackable” field being enabled.

Transmogrify - Boolean Field. If equals 1, then the item will use the transmogrify function. If equals 0, then ignore this. This field depends on the “useable” field being enabled.
TMogType - Links to a “code” field to determine which item is chosen to transmogrify this item to.
TMogMin - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the “TMogType” field, and that the transmogrify item has quantity.
TMogMax - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the “TMogType” field, and that the transmogrify item has quantity.

type - Points to an Item Type defined in the ItemTypes.txt file, which controls how the item functions
type2 - Points to a secondary Item Type defined in the ItemTypes.txt file, which controls how the item functions. This is optional but can add more functionalities and possibilities with the item.

dropsound - Points to a “Sound” field defined in the sounds.txt file. Used when the item is dropped on the ground.
dropsfxframe - Defines which frame in the “flippyfile” animation to play the “dropsound” sound when the item is dropped on the ground.
usesound - Points to a “Sound” field defined in the sounds.txt file. Used when the item is moved in the inventory or used.

unique - Boolean Field. If equals 1, then the item can only spawn as a Unique quality type. If equals 0, then the item can spawn as other quality types.

transparent - Boolean Field. If equals 1, then the item will be drawn transparent on the player model (similar to ethereal models). If equals 0, then the item will appear solid on the player model.
transtbl - Controls what type of transparency to use, based on the “transparent” field being enabled.

Code	Description
0	Transparency at 25%
1	Transparency at 50%
2	Transparency at 75%
3	Black Alpha Transparency
4	White Alpha Transparency
5	No Transparency
6	Dark Transparency (Unused)
7	Highlight Transparency (Used when mousing over the unit)
8	Blended

lightradius - Controls the value of the light radius that this item can apply on the monster. This only affects monsters with this item equipped, not other types of units. This is ignored if the item’s component on the monster is “lit”, “med”, or “hvy”.

belt - Controls which belt type to use for belt items only. This field determines what index entry in the belts.txt file to use.

quest - Controls what quest class is tied to the item which can enable certain item functionalities for a specific quest. Any value greater than 0 will also mean the item is flagged as a quest item, which can affect how it is displayed in tooltips, how it is traded with other players, its item rarity, and how it cannot be sold to an NPC. If equals 0, then the item will not be flagged as a quest item.

Code	Description
0	Not a quest item
1	Act 1 Prologue
2	Den of Evil
3	Sisters’ Burial Grounds
4	Tools of the Trade
5	The Search for Cain
6	The Forgotten Tower
7	Sisters to the Slaughter
8	Act 2 Prologue
9	Radament’s Lair
10	The Horadric Staff
11	The Tainted Sun
12	The Arcane Sanctuary
13	The Summoner
14	The Seven Tombs

15	Act 2 Traversed
16	Lam Esen's Tome
17	Khalim's Will
18	Blade of the Old Religion
19	The Golden Bird
20	The Blackened Temple
21	The Guardian
22	Act 4 Prologue
23	The Fallen Angel
24	Terror's End
25	The Hellforge
26	Rogue Warning
27	Guard in Town Warning
28	Guard in Desert Warning
29	Dark Wanderer Seen
30	Angel Warning
31	Respec from Akara Complete Act 5 Prologue
32	Siege on Harrogath
33	Rescue on Mount Arreat
34	Prison of Ice
35	Betrayal of Harrogath
36	Rite of Passage
37	Eve of Destruction

questdiffcheck - Boolean Field. If equals 1 and the "quest" field is enabled, then the game will check the current difficulty setting and will tie that difficulty setting to the quest item. This means that the player can have more than 1 of the same quest item as long each they are obtained per difficulty mode (Normal / Nightmare / Hell). If equals 0 and the "quest" field is enabled, then the player can only have 1 count of the quest item in the inventory, regardless of difficulty.

missiletype - Points to the "Id" field from the Missiles.txt file, which determines what type of missile is used when using the throwing weapons
durwarning - Controls the threshold value for durability to display the low durability warning UI. This is only used if the item has durability.
qntwarning - Controls the threshold value for quantity to display the low quantity warning UI. This is only used if the item has stacks.

mindam - The minimum physical damage provided by the item
maxdam - The maximum physical damage provided by the item

StrBonus - The percentage multiplier that gets multiplied the player's current Strength attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.
DexBonus - The percentage multiplier that gets multiplied the player's current Dexterity attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.

gemoffset - Determines the starting index offset for reading the gems.txt file when determining what effects gems or runes will have the item based on the "gemapplytype" field. For example, if this value equals 9, then the game will start with index 9 ("Chipped Emerald") and ignore the previously defined gems in the gems.txt file, which can mean that those ignored gems will not apply modifiers when socketed into the item.
bitfield1 - Controls different flags that can affect the item. Uses an integer value to check against different bit fields by using the "&" operator. For example, if the value equals 5 (binary = 101) then that returns true for both the 4 (binary = 100) and 1 (binary = 1) bit field values.

Bit Field Value	Binary Value	Description
1	1	Allow the item to be capable of having Magic quality
2	10	The item is classified as metal
4	100	The item is classified as a spellcaster item (currently does nothing)
8	1000	The item is classified as a skill based item (currently does nothing)

The following fields are separated per NPC in each Act:
[NPC]Min - Minimum amount of this item type in Normal rarity that the NPC can sell at once
[NPC]Max - Maximum amount of this item type in Normal rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicMin - Minimum amount of this item type in Magical rarity that the NPC can sell at once
[NPC]MagicMax - Maximum amount of this item type in Magical rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicLvl - Maximum magic level allowed for this item type in Magical rarity
Where [NPC] is one of the following:

Charsi
Gheed
Akara
Fara
Lysander
Drognan
Hratli
Alkor
Ormus
Elzix
Asheara
Cain
Halbu
Jamella
Larzuk
Malah
Anya

Transform - Controls the color palette change of the item for the character model graphics
InvTrans - Controls the color palette change of the item for the inventory graphics

Code	Color
0	No color change
1	Grey
2	Grey 2
3	Gold
4	Brown
5	Grey Brown
6	Inventory Grey
7	Inventory Grey 2
8	Inventory Grey Brown

SkipName - Boolean Field. If equals 1 and the item is Unique rarity, then skip adding the item's base name in its title. If equals 0, then ignore this.

NightmareUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Nightmare difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.
HellUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Hell difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.

Nameable - Boolean Field. If equals 1, then the item's name can be personalized by Anya for the Act 5 Betrayal of Harrogath quest reward. If equals 0, then the item cannot be used for the personalized name reward.
PermStoreItem - Boolean Field. If equals 1, then this item will always appear on the NPC's store. If equals 0, then the item will randomly appear on the NPC's store when appropriate.
diablocloneweight - The amount of weight added to the diablo clone progress when this item is sold. When offline, selling this item will instead immediately spawn diablo clone.

The following fields are exclusive to the misc.txt file because these fields only used with Miscellaneous type items:

autobelt - Boolean Field. If equals 1, then the item will automatically be placed is a free slot in the belt when picked up, if possible. If equals 0, then ignore this.

bettergem - Links to another item's "code" field. Used by the function 18 in the "Code" field from the shrines.txt file to know what a selected gem's upgrade will be when the player uses the gem shrine.

multibuy - Boolean Field. If equals 1, then use the multi-buy transaction function when holding the shift key and buying this item from an NPC store. This multi-buy function will automatically purchase enough of the item to fill up to a full quantity stack or fill the available belt slots if the item is has the "autobelt" field enabled. If equals 0, then ignore this.

spellicon - Determines the icon asset for displaying the item's spell. This uses an ID value based on the global skillicon file. If this value equals -1, then the item's spell will not display an icon. Used as a parameter for a "pspell" function.
pspell - Uses an ID value to select a spell function when the item is used. This depends on the item type.

Code	Parameters	Description
0		Do nothing
1	spellicon	SkillItemIdentify - Sets the spell icon. Identifies an item.
2		SkillItemTownPortal - The player creates a town portal
3	state stat1, calc1 stat2, calc2 stat3, calc3 len	SkillItemHealPotion 1. Applies a "state" on the player that is controlled by the "len" field 2. This function requires that the stat parameters be either "hitpoints", "hpregen", "mana", or "manarecovery" 3. Calculates a flat amount of these stats to restore to the player, based on the class and Vitality/Energy attribute for Life/Mana stats
4	state stat1, calc1 stat2, calc2 stat3, calc3 len	SkillItemHealPotion2 1. Applies a "state" on the player that is controlled by the "len" field 2. This function requires that the stat parameters be either "hitpoints", "hpregen", "mana", or "manarecovery" 3. Calculates a flat amount of these stats to restore to the player
5	stat1, calc1 stat2, calc2 stat3, calc3	SkillItemHealPotion3 - Adds a percentage of the stat's "maxstat" value (see ItemStatCost.txt) to the current stat. This percentage is determined by the related calculated value.
6	state cstate1 cstate2 len	SkillItemPotionAntidote - Clears the "cstate1" and "cstate2" states on the user and applies the "state" state with its duration controlled by the "len" field.
7		SkillItemTransmogrify - Opens the Horadric Cube UI
8		SkillItemElixir - Get a stat from item's mod class and set it to the it's "value" stat
9	state stat1, calc1 stat2, calc2 stat3, calc3 len	SkillItemHerb 1. Applies a "state" on the player that is controlled by the "len" field 2. Calculates a flat amount of the stats to set on the player 3. Has a special case where if the stat equals "staminarecoverybonus" then also set the current "stamina" stat to be equals to the "maxstamina" stat
10		SkillItemSkill - Cast a level 1 Sorceress Fire Ball skill at a targeted enemy or targeted location
11		SkillItemSkillXY - Cast a level 1 Sorceress Fire Ball skill at a targeted location

state - Links to a “state” field defined in the states.txt file. It signifies what state will be applied to the player when the item is used. Used as a parameter for a “pspell” function.
cstate1 & cstate2 - Links to a “state” field defined in the states.txt file. It signifies what state will be removed from the player when the item is used. Used as a parameter for a “pspell” function.
len - Calculates the frame length of a state. Used as a parameter for a “pspell” function.
stat1 (to stat3) - Controls the stat modifier when the item is used (Uses the “code” field from Properties.txt). Used as a parameter for a “pspell” function.
calc1 (to calc3) - Calculates the value of the relative “stat#” field. Used as a parameter for a “pspell” function.

spelldesc - Uses an ID value to select a function to format a string and add this string to the item’s tooltip

Code	Parameters	Description
0 (or empty)		Do nothing
1	spelldescstr spelldescstr2	1. Add the “spelldescstr” string to the item tooltip 2. If “spelldescstr2” is not null, then use this string instead of “spelldescstr” string when playing in controller mode
2	spelldescstr spelldesccalc stat1	1. Evaluate the “spelldesccalc” field 2. If the “stat1” value equals “hitpoints” or “hpregen” then adjust the calculated value based on the relative “HealthPotionPercent” value from charstats.txt file. 3. If the “stat1” value equals “mana” or “manarecovery” then adjust the calculated value based on the relative “ManaPotionPercent” value from charstats.txt file. 4. Insert the calculated value into the “spelldescstr” string and add the string to the item tooltip
3	spelldescstr spelldesccalc	1. Evaluate the “spelldesccalc” field 2. Add the “spelldescstr” string to the item tooltip 3. Append a space string and the calculated value after the “spelldescstr” string to the item tooltip
4	spelldescstr spelldesccalc	1. Evaluate the “spelldesccalc” field 2. Insert the calculated value into the “spelldescstr” string and add the string to the item tooltip

spelldescstr & spelldescstr2 - String Key. Used as a parameter for the “spelldesc” function.
spelldesccalc - Calculates the numeric equation. Used as a parameter for the “spelldesc” function.
spelldesccolor - Uses a code number to change the color of the string used in the “spelldesc” function.

Code	Description
0	White (R=255, G=255, B=255)
1	Red (R=255, G=77, B=77)
2	Green (R=0, G=255, B=0)
3	Blue (R=105, G=105, B=255)
4	Light Gold (R=199, G=179, B=119)
5	Grey (R=105, G=105, B=105)
6	Black (R=0, G=0, B=0)
7	Dark Gold (R=208, G=194, B=125)
8	Orange (R=255, G=168, B=0)
9	Yellow (R=255, G=255, B=100)
10	Dark Green (R=0, G=128, B=0)
11	Purple (R=174, G=0, B=255)
12	Medium Green (R=0, G=200, B=0)

monequip.txt

Overview

This file controls how a monster may be created with specific type of inventory equipment items
These equipment items are randomly generated on the monster, based on the specified parameters in this file
This data relies on the “inventory” field being enabled for the listed monster, found in the monstats.txt file

Data Fields

monster - Defines the monster that should be equipped. Points to the matching “Id” value in the monstats.txt file. If the monster has multiple defined equipment possibilities, then they should always be grouped together. The game will go through the list in order to match what is best to use for the monster.
oninit - Defines if the monster equipment is added on initialization during the monster’s creation, depending how the monster is spawned. Monsters created by a skill have this value set to 0. Monsters created by a level have this value set to 1.
level - Defines the level requirement for the monster in order to gain this equipment. The game will prefer the highest level allowed, so the order of these equipment should be from highest level to lowest level.

item1 (to item3) - Item that the be equipped on the monster (Uses ID pointer from Weapons.txt, Armor.txt or Misc.txt)
loc1 (to loc3) - Specifies the inventory slot where the item will be equipped. Once an item is equipped on that body location, then the game will skip any duplicate calls to equipping the same body location. This is another reason why the equipment should be ordered from highest level to lowest level.

Code	Description
(empty)	None
head	Head
neck	Neck
tors	Torso
rarm	Right Arm

larm	Left Arm
rrin	Right Ring
lrin	Left Ring
belt	Belt
feet	Feet
glov	Gloves

mod1 (to mod3) - Controls the quality level of the related item

Item Quality Code	Description
0	Any Quality (Used for a random quality)
1	Low Quality (Ex: "Crude")
2	Normal Quality (Default value if the value is empty)
3	High Quality (Superior)
4	Magic Quality (Uses Magic Prefixes and Suffixes)
5	Set Item
6	Rare Quality
7	Unique (Predetermined stats)

MonLvl.txt

Overview

This file controls how monster statistics increase per level based on the current game type and difficulty

The “(N)” text in field names signifies to use that specific value for games in Nightmare difficulty
The “(H)” text in field names signifies to use that specific value for games in Hell difficulty

This file is used by monstats.txt

Data Fields

Level - An integer value to determine how to scale the monster’s statistics when at a specific level

The following are used for the following game type: Non-Ladder Battle.net / Singleplayer / Open Battle.net / TCP
AC & AC(N) & AC(H) - Percentage multiplier for increasing the Monster’s Defense (multiplies with the “AC” field in monstats.txt)
TH & TH(N) & TH(H) - Percentage multiplier for increasing the Monster’s Attack Rating (multiplies with the “A1TH” and “A2TH” fields in monstats.txt)
HP & HP(N) & HP(H) - Percentage multiplier for increasing the Monster’s Life (multiplies with the “MinHP” and “MaxHP” fields in monstats.txt)
DM & DM(N) & DM(H) - Percentage multiplier for increasing the Monster’s Damage (multiplies with the “A1MinD”, “A1MaxD”, “A2MinD”, “A2MaxD”, “El1MinD”, “El1MaxD”, “El2MinD”, “El2MaxD”, “El3MinD”, and “El3MaxD” fields in monstats.txt)
XP & XP(N) & XP(H) - Percentage multiplier for increasing the Experience provided to the player when killing the Monster (multiplies with the “Exp” fields in monstats.txt)

The following are used only for the Ladder Battle.net game type
L-AC & L-AC(N) & L-AC(H) - Percentage multiplier for increasing the Monster’s Defense (multiplies with the “AC” field in monstats.txt)
L-TH & L-TH(N) & L-TH(H) - Percentage multiplier for increasing the Monster’s Life (multiplies with the “A1TH” and “A2TH” fields in monstats.txt)
L-HP & L-HP(N) & L-HP(H) - Percentage multiplier for increasing the Monster’s Life (multiplies with the “MinHP” and “MaxHP” fields in monstats.txt)
L-DM & L-DM(N) & L-DM(H) - Percentage multiplier for increasing the Monster’s Damage (multiplies with the “A1MinD”, “A1MaxD”, “A2MinD”, “A2MaxD”, “El1MinD”, “El1MaxD”, “El2MinD”, “El2MaxD”, “El3MinD”, and “El3MaxD” fields in monstats.txt)
L-XP & L-XP(N) & L-XP(H) - Percentage multiplier for increasing the Experience provided to the player when killing the Monster (multiplies with the “Exp” fields in monstats.txt)

MonPreset.txt

Overview

This file controls which monsters are preloaded in a preset, based on the Act number

Data Fields

Act - Defines the Act number used for each Monster Preset. Uses values between 1 to 5.
Place - Defines either a Super Unique monster from superuniques.txt, or a monster from monstats.txt, or a place from monplace.txt. This defines the Monster Preset which is used for preloading, such as during level transitions

MonProp.txt

Overview

This file controls special properties that can be added to a monster, based on the game difficulty

The “(N)” text in field names signifies to use that specific value for games in Nightmare difficulty
The “(H)” text in field names signifies to use that specific value for games in Hell difficulty

This file is used by the monstats.txt file

Data Fields

Id - Defines the monster that should gain the Property. Points to the matching “Id” value in the monstats.txt file.

prop1 (to prop6) - Defines with Property to apply to the monster (Uses the “code field from Properties.txt)
chance1 (to chance6) - The percent chance that the related property (prop#) will be assigned. If this value equals 0, then the Property will always be applied.
par1 (to par6) - The “parameter” value associated with the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
min1 (to min6) - The “min” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
max1 (to max6) - The “max” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)

monseq.txt

Overview

This file controls the sequence table used for specifying events during certain animation frames, such as when using skills.
A sequence is divided into multiple lines in this file to define each frame in the animation and to determine which event to use during a specific frame.

Data Fields

sequence - Establishes the Monster Sequence index. An entire monster sequence can be composed of multiple sequence lines, which means that each line needs to have matching “sequence” fields and must be in contiguous order.
mode - Defines which monster mode animation to use for the sequence

Code	Description
DT	Death
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
xx	Sequence
RN	Run

frame - The in-game frame number for the animation. For the first line in the sequence, this value will establish where the starting frame for the animation. These values should be in contiguous order for the sequence.
dir - Defines the numeric animation direction that the frame use. Most animations have between 8 to 64 maximum directions.
event - Defines what type of event will be used when the frame triggers

Code	Description
0	No event, do nothing
1	Do Melee attack
2	Do Missile attack
3	Play a sound
4	Use a Skill

monstats.txt

Overview

This file controls the main functionalities and statistics for every monster in the game. This includes enemy monsters, pets, and NPC units.

This file is connected to the monstats2.txt file, so additional functionalities can be found in that file. This means that this file’s number and order of entries should be identical with the monstats2.txt file.

The “(N)” text in field names signifies to use that specific value for games in Nightmare difficulty
The “(H)” text in field names signifies to use that specific value for games in Hell difficulty

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Id - Controls the unique name ID to define the monster. This must match the same value in the monstats2.txt file.
BaseId - Points to the “Id” of another monster to define the monster’s base type. This is to create groups of monsters which are considered the same type.
NextInClass - Points to the “Id” of another monster to signify the next monster in the group of this monster’s type. This is to continue the groups of monsters which are considered the same type. The order should be contiguous.

TransLvl - Defines the color transform level to use for this monster, which affects what color palette that the monster will use

Code	Description
0	Cold
1	Poison
2	Level 0
3	Level 1
4	Level 2
5	Level 3
6	Level 4
7	Level Miscellaneous

NameStr - String Key. Used to define the monster’s name, such as in the Life bar UI when it is being targeted.
MonStatsEx - Controls a pointer to the “Id” of a monster to define which entry to use in the monstats2.txt file. This should always match the same “Id” value for the monster in this file.
MonProp - Points to the “Id” field from the MonProp.txt file. Used to add special modifiers to the monster.
MonType - Points to the “type” field from the MonType.txt file. Used to handle the monster’s classification.
AI - Points to a type of AI script to use for the monster (See monai.txt).
DescStr - String Key. Used to add a string to appear as an additional description below the Life bar UI when the monster is being targeted.
Code - Controls the token used for choosing the proper cells to display the monster’s graphics
enabled - Boolean Field. If equals 1, then this monster is allowed to spawn in the game. If equals 0, then this monster will never spawn in the game.
rangedtype - Boolean Field. If equals 1, then the monster will be classified as a ranged type. If equals 0, then the monster will be classified as a melee type.

placespawn - Boolean Field. If equals 1, then this monster will be treated as a spawner, so monsters that spawn can be initially placed within this monster. If equals 0, then ignore this.
spawn - Points to the “Id” of another monster to control what kind of monster is spawned from this monster. This is only used if the “placespawn” field is enabled.
spawnx & spawny - Controls the X & Y offsets for where another monster is displaced when being spawned by this monster.
spawnmode - Defines the animation mode that the spawned monsters will be initiated with

Token	Description
DT	Death / Reset
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
xx	Sequence
RN	Run

minion1 & minion2 - Points to the “Id” of another monster to control what kind of monster is spawned with this monster when it is spawned, like a monster pack. The “minion1” field is also used for spawning a monster when this monster is killed while it has the “SplEndDeath” field enabled.
SetBoss - Boolean Field. If equals 1, then set the monster AI to use the Boss AI type, which can affect the monster’s behaviors. If equals 0, then ignore this.
BossXfer - Boolean Field. If equals 1, then the monster’s AI will transfer its boss recognition to another monster, which can affect the minion monster behaviors after this boss is killed. If equals 0, then ignore no boss AI will transfer and minion monsters will behave differently after the boss is killed. This field relies on the “SetBoss” field being enabled.
PartyMin - The minimum number of minions that can spawn with this monster. Uses the “minion1” and “minion2” fields. The actual number is a random value chosen between the “PartyMin” and “PartyMax” field values.
PartyMax - The maximum number of minions that can spawn with this monster. Uses the “minion1” and “minion2” fields. The actual number is a random value chosen between the “PartyMin” and “PartyMax” field values.
MinGrp - The minimum number of duplicates of this monster that can spawn together. The actual number is a random value chosen between the “MinGrp” and “MaxGrp” field values.
MaxGrp - The maximum number of duplicates of this monster that can spawn together. The actual number is a random value chosen between the “MinGrp” and “MaxGrp” field values.
sparsePopulate - If this value is greater than 0, then it controls the percent chance that this monster does not spawn, and another monster will spawn in its place. (Out of 100)

Velocity - Determines the movement velocity of the monster, which can be the monster’s baseline walk speed.
Run - Determines the run speed of the monster as opposed to walk speed. This is only used if the monster has a Run mode.
Rarity - Modifies the chance that this monster will be chosen to spawn in the area level. The higher the value is, then the more likely this monster will be chosen. This value acts as a numerator and a denominator. All “Rarity” values of possible monsters get summed together to give a total denominator, used for the random roll. For example, if there are 3 possible monsters that can spawn, and their “Rarity” values are 1, 2, 2, then their chances to be chosen are 1/5, 2/5, and 2/5 respectively. If this value equals 0, then this monster is never randomly selected to spawn in an area level.
Level - Determines the monster’s level. This value for Nightmare and Hell difficulty can be overridden by the area level’s “MonLvl#” or “MonLvl#Ex” value (See Levels.txt), unless the monster’s “boss” and “noRatio” fields are enabled.

MonSound - Points to the “Id” field of a monster sound from the monsounds.txt file. This is used to control the monsters assigned sounds, when the monster is spawned as a Normal monster.
UMonSound - Points to the “Id” field of a monster sound from the monsounds.txt file. This is used to control the monsters assigned sounds, when the monster is spawned as a Unique or Champion monster.

threat - Controls the AI threat value of the monster which can affect the targeting priorities of enemy Ais for this monster. The higher this value is, then the more likely that enemy AI will target this monster.
aidel - Controls the delay in frame length for how often the monster’s AI will update its commands. A lower delay means that the monster will perform commands more often without as long of a pause in between.
aidist - Controls the maximum distance (measured in tiles) between the monster and an enemy until the monster’s AI becomes aggressive. If equals 0, then default to 35.
aip1 (to aip8) - Defines numeric parameters used to control various functions of the monster’s AI. These fields depend on which AI script is being used (See monai.txt, and the “AI” field in monstats.txt)

MissA1 & MissA2 - Points to the “Missile” field from Missiles.txt to determine which missile to use when the monster is in Attack 1 & Attack 2 mode
MissS1 (to MissS4) - Points to the “Missile” field from Missiles.txt to determine which missile to use when the monster is in Skill 1 (to Skill 4) mode
MissC - Points to the “Missile” field from Missiles.txt to determine which missile to use when the monster is in Cast mode
MissSQ - Points to the “Missile” field from Missiles.txt to determine which missile to use when the monster is in Sequence mode

Align - Controls the monster’s alignment, which determines if the monster will be an enemy, ally, or neutral party to the player.

Code	Description
0	Evil Alignment - The monster will attack the player
1	Good Alignment - The monster will fight with the player and will be in the player’s party
2	Neutral Alignment - The monster will not attack the player and will not be in the player’s party

isSpawn - Boolean Field. If equals 1, then the monster is allowed to spawn in an area level. If equals 0, then the monster will not be spawned automatically in an area level.
isMelee - Boolean Field. If equals 1, then the monster is classified as a melee only type, which can affect its AI behaviors and what monster modifiers are allowed on the monster. If equals 0, then ignore this.
npc - Boolean Field. If equals 1, then the monster is classified as an NPC (Non-Playable Character), which can affect its AI behaviors and how the player treats this monster. If equals 0, then ignore this.
interact - Boolean Field. If equals 1, then the monster is interactable, meaning that the player can click on the monster to perform an interact command instead of attacking. If equals 0, then ignore this.
inventory - Boolean Field. If equals 1, then monster will have an inventory with randomly generated items, such as an NPC with shop items (if the “interact” field is enabled) or a summoned unit with random equipped items (also see monequip.txt). If equals 0, then ignore this.
inTown - Boolean Field. If equals 1, then the monster is allowed to be in town. If equals 0, then the monster is not allowed to be in town, which can affect or disable their AI or collision from entering towns.
lUndead - Boolean Field. If equals 1, then the monster is treated as a Low Undead, meaning that the monster is classified as an Undead type and can be resurrected by certain AI. If equals 0, then ignore this.
hUndead - Boolean Field. If equals 1, then the monster is treated as a High Undead, meaning that the monster is classified as an Undead type but cannot be resurrected by certain AI. If equals 0, then ignore this.
demon - Boolean Field. If equals 1, then the monster is classified as a Demon type. If equals 0, then ignore this.
flying - Boolean Field. If equals 1, then the monster is flagged as a flying type, which can affect its collision with the area level and how it is spawned. If equals 0, then ignore this.
opendoors - Boolean Field. If equals 1, then the monster will use its AI to open doors if necessary. If equals 0, then the monster cannot open doors and will treat doors as another type of collision.
boss - Boolean Field. If equals 1, then the monster is classified as a Boss type, which can affect boss related AI and functions. If equals 0, then ignore this.
primeevil - Boolean Field. If equals 1, then the monster is classified as a Prime Evil type, or an Act End boss, which can affect various skills, AI, and damage related functions. If equals 0, then ignore this.
killable - Boolean Field. If equals 1, then the monster can be killed, damage, and be put in a Death or Dead mode. If equals 0, then the monster cannot be damaged or killed.
switchai - Boolean Field. If equals 1, then monster’s AI can switched, such as by the Assassin’s Mind Blast ability. If equals 0, then the monster AI cannot be switched.
noAura - Boolean Field. If equals 1, then the monster cannot be affected by friendly auras. If equals 0, then the monster can be affected by friendly auras.
nomultishot - Boolean Field. If equals 1, then the monster is not allowed to spawn with the Multi-Shot unique monster modifier (See monumod.txt). If equals 0, then ignore this.
neverCount - Boolean Field. If equals 1, then the monster is not counted on the list of the active monsters in the area, which affects spawning and saving functions. If equals 0, then the monster will be accounted for, and can be part of the active or inactive list functions.
petIgnore - Boolean Field. If equals 1, then pet AI scripts will ignore this monster (See monai.txt). If equals 0, then pet AI will attack this monster.
deathDmg - Boolean Field. If equals 1, then the monster will explode on death. This has special cases for the “bonefetish1” and “siegebeast1” monster classes, otherwise the monster will use a general death damage function to damage nearby units based on the monster’s health percentage. If equals 0, then ignore this.
genericSpawn - Boolean Field. If equals 1, the monster is flagged as a possible selection for the AI generic spawner function. There are defaults for using the If equals 0, then ignore this.
zoo - Boolean Field. If equals 1, then the monster will be flagged as a zoo type monster, which will give it the AI zoo behavior. If equals 0, then ignore this.
CannotDesecrate - Boolean Field. If equals 1, then the monster will not be able to be desecrated when inside a desecrated level. If equals 0, then ignore this.

rightArmItemType - Determines what type of items the monster is allowed to hold in its right arm (see ItemTypes.txt). A blank value means it can hold any item.
leftArmItemType – Determines what type of items the monster is allowed to hold in its left arm (see ItemTypes.txt). A blank value means it can hold any item.
canNotUseTwoHandedItems – Boolean Field. If equals 1, then the monster can not items marked as two handed (see weapons.txt)

SendSkills - Determines which of the monster’s skill’s level should be sent to the client. Uses a byte value, where the code tests each bit to determine which of the monster’s skills to check.
Skill1 (to Skill8) - Points to a skill from the “skill” field in the skills.txt file. This gives the monster the skill to use (requires “Sk#mode”)
Sk1mode (to Sk8mode) - Determines the monster’s animation mode when using the related skill. Outside of the standard animation mode inputs, the field can also point to a “sequence” defined in the monseq.txt file, which handle a specific set of frames to place a sequence animation.

Token	Description

DT	Death / Reset
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
RN	Run

Sk1lvl (to Sk8lvl) - Controls the base skill level of the related skill on the monster

Drain - Controls the monster's overall Life and Mana steal percentage. This can also be affected by the "LifeStealDivisor" and "ManaStealDivisor" fields from the difficultylevels.txt file. If equals 0, then the monster will not have Life or Mana steal.
coldeffect - Sets the percentage change in movement speed and attack rate when the monster if chilled by a cold effect. If this equals 0, then the monster will be immune to the cold effect.

ResDm - Sets the monster's Physical Damage Resistance stat
ResMa - Sets the monster's Magic Resistance stat
ResFi - Sets the monster's Fire Resistance stat
ResLi - Sets the monster's Lightning Resistance stat
ResCo - Sets the monster's Cold Resistance stat
ResPo - Sets the monster's Poison Resistance stat

DamageRegen - Controls the monster's Life regeneration per frame. This is calculated based on the monster's maximum life: Regeneration Rate = (Life * "DamageRegen") / 16
SkillDamage - Points to a skill from the "skill" field in the skills.txt file. This changes the monster's min physical damage, max physical damage, and Attack Rating to be based off the values from the linked skill and its current level from the monster's owner (usually the player who summoned the monster).

noRatio - Boolean Field. If equals 1, then use this file's fields to determine the monster's baseline stats ("minHP", "maxHP", "AC", "Exp", "A1MinD", "A1MaxD", "A1TH", "A2MinD", "A2MaxD", "A2TH", "S1MinD", "S1MaxD", "S1TH"). If equals 0, then use the MonLvl.txt file to determine the monster's baseline stats.
ShieldBlockOverride - If equals 1, then the monster can block without a shield (the block chance stat will take effect even without a shield equipped). If equals 2, then the monster cannot block at all, even with a shield equipped. If equals 0, then ignore this.
ToBlock - The monster's percent chance to block an attack
Crit - The percent chance for the monster to score a critical hit when attacking an enemy, which causes the attack to deal double damage
minHP - The monster's minimum amount of Life when spawned
maxHP - The monster's maximum amount of Life when spawned
AC - The monster's Defense value
Exp - The amount of Experience that is rewarded to the player when the monster is killed
A1MinD - The minimum damage dealt by the monster when it is using the Attack 1 (A1) animation mode
A1MaxD - The maximum damage dealt by the monster when it is using the Attack 1 (A1) animation mode
A1TH - The monster's Attack Rating when it is using the Attack 1 (A1) animation mode
A2MinD - The minimum damage dealt by the monster when it is using the Attack 2 (A2) animation mode
A2MaxD - The maximum damage dealt by the monster when it is using the Attack 2 (A2) animation mode
A2TH - The monster's Attack Rating when it is using the Attack 2 (A2) animation mode
S1MinD - The minimum damage dealt by the monster when it is using the Skill 1 (S1) animation mode
S1MaxD - The maximum damage dealt by the monster when it is using the Skill 1 (S1) animation mode
S1TH - The monster's Attack Rating when it is using the Skill 1 (S1) animation mode

El1Mode (to El3Mode) - Determines which animation mode will trigger an additional elemental damage type when used

Token	Description
DT	Death / Reset
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
xx	Sequence
RN	Run

El1Type (to El3Type) - Defines the type of elemental damage. This field is used when El#Mode is not null.

Code	Description
fire	Fire
ltng	Lightning
mag	Magic

cold	Cold
pois	Poison
life	Life Drain
mana	Mana Drain
stam	Stamina Drain
stun	Stun
rand	Randomly select between Fire, Lightning, Magic, Cold, or Poison If the related “El#Dur” field equals 0, then default the value to 25
burn	Burning
frze	Freeze

El1Pct (to El3Pct) - Controls the random percent chance (out of 100) that the monster will append the element damage to the attack. This field is used when El#Mode is not null.
El1MinD (to El3MinD) - The minimum element damage applied to the attack. This field is used when El#Mode is not null.
El1MaxD (to El3MaxD) - The maximum element damage applied to the attack. This field is used when El#Mode is not null.
El1Dur (to El3Dur) - Controls the duration of the related element mode in frame lengths (25 Frames = 1 Second). This is only applicable for the Cold, Poison, Stun, Burning, Freeze elements. There are special cases when evaluating the elements, where Poison min and max damage are multiplied by 10, and Poison duration is multiplied by 2. This field is used when El#Mode is not null.

TreasureClass - Defines which Treasure Class is used by the monster when it is killed. Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for normal monster types.
TreasureClassChamp - Defines which Treasure Class is used by the monster when it is killed. Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for Champion monster types.
TreasureClassUnique - Defines which Treasure Class is used by the monster when it is killed. Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for Unique monster types.
TreasureClassQuest - Defines which Treasure Class is used by the monster when it is killed. Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for quest related monster drops (See “TCQuestId” and “TCQuestCP”).
TreasureClassDesecrated - Defines which Treasure Class is used by the monster when it is killed while desecrated (Terrorized). Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for normal monster types.
TreasureClassDesecratedChamp - Defines which Treasure Class is used by the monster when it is killed while desecrated (Terrorized). Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for Champion monster types.
TreasureClassDesecratedUnique - Defines which Treasure Class is used by the monster when it is killed while desecrated (Terrorized). Points to the “Treasure Class” field from the TreasureClassEx.txt file. Used for Unique monster types.

TCQuestId - Checks to see if the player has does not have a quest flag progress. If not, then use the “TreasureClass4” field, based on the game’s current difficulty.

Code	Quest Progress
0	Act 1 Prologue Seen
1	Den of Evil Completed
2	Sisters’ Burial Grounds Completed
3	Tools of the Trade Completed
4	The Search for Cain Completed
5	The Forgotten Tower Completed
6	Sisters to the Slaughter Completed
7	Act 1 Traversed
8	Act 2 Prologue Seen
9	Radament’s Lair Completed
10	The Horadric Staff Completed
11	Tainted Sun Completed
12	Arcane Sanctuary Completed
13	The Summoner Completed
14	The Seven Tombs Completed
15	Act 2 Traversed
16	Act 3 Prologue Seen
17	Lam Esen's Tome Completed
18	Khalim's Will Completed
19	Blade of the Old Religion Completed
20	The Golden Bird Completed
21	The Blackened Temple Completed
22	The Guardian Completed
23	Act 3 Traversed
24	Act 4 Prologue Seen
25	The Fallen Angel Completed
26	Terror's End Completed
27	Hell's Forge Completed
28	Act 4 Traversed
29	Rogue Warning Complete
30	Guard in Town Warning Complete
31	Guard in Desert Warning Complete
32	Dark Wanderer Seen
33	Angel Warning Complete
34	Act 5 Prologue Seen
35	Siege on Harrogath Completed
36	Rescue on Mount Arreat Completed
37	Prison of Ice Completed
38	Betrayal of Harrogath Completed

39	Rite of Passage Completed
40	Eve of Destruction Completed
41	Respecialization from Akara is Completed

TCQuestCP - Controls which Quest Checkpoint, or current progress within a quest (based on the “TCQuestId” value), is needed to use the “TreasureClass4” field, based on the game’s current difficulty

Code	Description
0	History Success
1	History Earned Reward
2	Checkpoint 1
3	Checkpoint 2
4	Checkpoint 3
5	Checkpoint 4
6	Checkpoint 5
7	Checkpoint 6
8	Checkpoint 7
9	Checkpoint 8
10	Checkpoint 9
11	Checkpoint 10
12	Complete Quest Log
13	Current Game Success
14	Current Game Failure
15	Previous Game

SplEndDeath - Controls a special case death handler for the monster that is ran on the server side

Code	Description
null	Do nothing
1	Spawn the monster type from the “minion1” field after this monster dies
2	Kill the source unit that is related to this monster. Typically this is a mount type unit that the monster is riding when it dies

SplGetModeChart - Boolean Field. If equals 1, then check special case handlers of certain monsters with specific “BaseId” fields while they are using certain a mode and perform a function. If equals 0, then ignore this.

Index	“BaseId”	Description
243 333 705	“diablo” “diablo clone” “uberdiablo”	If current mode equals Skill 3 (S3) or Skill 4 (S4), then do a generic attack function
403	“trappedsoul1”	If current mode equals Attack 1 (A1), Attack 2 (A2), Skill 1 (S1), or Skill 2 (S2), then do a generic attack function and end it with setting the monster to start the Skill 1 mode and skip the AI pause
543	“baalthrone”	If the current mode equals Skill 3 (S3), then tell the monster to do its Cast mode (SC)
544 570 709	“baalcrab” “baalclone” “uberbaal”	If the current mode equals Skill 3 (S3), then tell the monster to do its Cast mode (SC)
417 418	“shadowwarrior” “shadowmaster”	If the current mode equals Skill 4 (S4), then tell the monster to do a generic attack function

SplEndGeneric - Boolean Field. If equals 1, then check special case handlers of monsters with specific “BaseId” fields while they are ending certain modes and perform a function. If equals 0, then ignore this.

Index	“BaseId”	Mode that is ending	Description
110	“vulture1”	Skill 1 (S1)	Process the event Run AI
118	“willowwisp1”	Walk (WL)	Process the event Run AI
136	“batdemon1”	Skill 3 (S3) or Skill 4 (S4)	Process the event Run AI
230 231	“firebeast” “iceglobe”	Any mode	Process the event Run AI
247	“frogdemon1”	Sequence (xx)	Process the event Run AI
403	“trappedsoul1”	Any mode	Process the event Run AI

SplClientEnd - Boolean Field. If equals 1, then on the client side, check special case handlers of monsters with specific “BaseId” fields while they are ending certain modes and perform a function. If equals 0, then ignore this.

Index	“BaseId”	Mode that is ending	Description
110	“vulture1”	Skill 1 (S1)	Ignore setting the monster back to Neutral (NU)
403 404	“trappedsoul1” “trappedsoul2”	Skill 1 (S1) or Skill 2 (S2) or Attack 1 (A1) or Attack 2 (A2)	Set the mode to Skill 1 (S1)
136	“batdemon1”	Skill 3 (S3)	Set the mode to Skill 4 (S4)
136	“batdemon1”	Skill 4 (S4)	Ignore setting the monster back to Neutral (NU)
118 231	“willowwisp1” “iceglobe”	Walk (WL)	Ignore setting the monster back to Neutral (NU)
497	“catapult1”	Attack 1 (A1)	Remove the siege missile and add a new one
247	“frogdemon1”	Sequence (xx)	Ignore setting the monster back to Neutral (NU)
284	“maggotqueen1”	Dead (DD)	Ignore setting the monster back to Neutral (NU)

monstats2.txt

Overview

This file controls additional functionalities and statistics for every monster in the game.

This file is treated as a continuation of the monstats.txt file, and therefore its amount of entries should be identical with the monstats.txt file.

Data Fields

Id - Controls the unique name ID to define the monster. This must match the same value in the monstats.txt file.

Height - Determines the height of the monster. This has 2 purposes. The first purpose is to act as an index value for selecting which icebreak missile to use when the monster dies while frozen. The second purpose is to select a code which affects what attack animation the player characters will use when attacking the monster (Attack1 and or Attack2). See each code description types below.

Code	Description	Code	Description
1	icebreaksmall	1	Low Height
2	icebreakmedium	2	High Height
3	icebreaklarge	3	Both types of Heights
4	icebreaksmoke		

OverlayHeight - Determines the height value of overlays (See Overlay.txt) for the monster
pixHeight - Determines the pixel height value for the damage bar when the monster is selected

SizeX & SizeY - Determines the tile grid size of the monster which is use for handling placement when the monster spawns or uses movement skills
spawnCol - Controls the method for spawning the monster based on the collisions in the environment

Code	Description
0	Normal - Basic Monster Pathing
1	Water - Handle water terrain where units cannot normally walk, but can fly over
2	Preset - Placement handler which considers walls, no pathable areas, objects, doors, items, and other monsters
3	Force - Override any collision

MeleeRng - Controls the range of the monster's melee attack, which can affect also affect certain AI pathing. If this value equals 255, then refer to the monster's weapon class ("BaseW").
BaseW - Defines the monster's base weapon class, which can affect how the monster attacks

Code	Description
hth	Hand to Hand (Default value if the value is empty)
1hs	One Handed Swing
1ht	One Handed Thrust
bow	Bow
2hs	Two Handed Swing
2ht	Two Handed Thrust
1js	Dual Wielding - Left Jab Right Swing
1jt	Dual Wielding - Left Jab Right Thrust
1ss	Dual Wielding - Left Swing Right Swing
1st	Dual Wielding - Left Swing Right Thrust
stf	Staff
xbw	Crossbow
ht1	One Hand to Hand
ht2	Two Hand to Hand

HitClass - Defines the specific class of an attack when the monster successfully hits with an attack. This can affect the sound and overlay display of the attack hit.

Code	Description
0	None
1	Hand to Hand
2	One Handed Swing Small
3	One Handed Swing Large
4	Two Handed Swing Small
5	Two Handed Swing Large
6	One Handed Thrust
7	Two Handed Thrust
8	Club
9	Staff
10	Bow
11	Crossbow
12	Claw
13	Do Overlay

The following are formula fields that define the types of visual graphics to use for the specific component field. Users can add input parameters by adding a comma "," to the input and using a code. For a list of possible component inputs, see the compcode.txt file.
HDv - Head visual
TRv - Torso visual
LGv - Legs visual
RAv - Right Arm visual
LAv - Left Arm visual
RHv - Right Hand visual
LHv - Left Hand visual
SHv - Shield visual

S1v (to S8v) - Special 1 to Special 8 visual

The following fields are Boolean fields, which control which specific component piece is enabled for the monster. If equals 1, then the component is enabled. If equals 0, then the monster does not have that component.
HD - Head
TR - Torso
LG - Legs
RA - Right Arm
LA - Left Arm
RH - Right Hand
LH - Left Hand
SH - Shield
S1 (to S8) - Special 1 to Special 8

TotalPieces - Defines the total amount of component pieces that the monster uses. This value should match the number of enabled Boolean fields listed above.

mDT - Boolean Field. If equals 1, then enable the Death Mode for the monster. If equals 0, then this mode is disabled.
mNU - Boolean Field. If equals 1, then enable the Neutral Mode for the monster. If equals 0, then this mode is disabled.
mWL - Boolean Field. If equals 1, then enable the Walk Mode for the monster. If equals 0, then this mode is disabled.
mGH - Boolean Field. If equals 1, then enable the Get Hit Mode for the monster. If equals 0, then this mode is disabled.
mA1 & mA2- Boolean Field. If equals 1, then enable the Attack 1 (and Attack 2) Mode for the monster. If equals 0, then this mode is disabled.
mBL - Boolean Field. If equals 1, then enable the Block Mode for the monster. If equals 0, then this mode is disabled.
mSC - Boolean Field. If equals 1, then enable the Cast Mode for the monster. If equals 0, then this mode is disabled.
mS1 (to mS4) - Boolean Field. If equals 1, then enable the Skill 1 (to Skill4) Mode for the monster. If equals 0, then this mode is disabled.
mDD - Boolean Field. If equals 1, then enable the Dead Mode for the monster. If equals 0, then this mode is disabled.
mKB - Boolean Field. If equals 1, then enable the Knockback Mode for the monster. If equals 0, then this mode is disabled.
mSQ - Boolean Field. If equals 1, then enable the Sequence Mode for the monster. If equals 0, then this mode is disabled.
mRN - Boolean Field. If equals 1, then enable the Run Mode for the monster. If equals 0, then this mode is disabled.

dDT - Defines the number of directions that the monster can face during Death Mode
dNU - Defines the number of directions that the monster can face during Neutral Mode
dWL - Defines the number of directions that the monster can face during Walk Mode
dGH - Defines the number of directions that the monster can face during Get Hit Mode
dA1 & dA2 - Defines the number of directions that the monster can face during Attack 1 (and Attack 2) Mode
dBL - Defines the number of directions that the monster can face during Block Mode
dSC - Defines the number of directions that the monster can face during Cast Mode
dS1 (to dS4) - Defines the number of directions that the monster can face during Skill 1 (to Skill 4) Mode
dDD - Defines the number of directions that the monster can face during Dead Mode
dKB - Defines the number of directions that the monster can face during Knockback Mode
dSQ - Defines the number of directions that the monster can face during Sequence Mode
dRN - Defines the number of directions that the monster can face during Run Mode

A1mv & A2mv - Boolean Field. If equals 1, then enable the Attack 1 (and Attack 2) Mode while the monster is moving with the Walk mode or Run mode. If equals 0, then this mode is disabled while the monster is moving.
SCmv - Boolean Field. If equals 1, then enable the Cast Mode while the monster is moving with the Walk mode or Run mode. If equals 0, then this mode is disabled while the monster is moving.
S1mv (to S4mv) - Boolean Field. If equals 1, then enable the Skill 1 (to Skill 4) Mode while the monster is moving with the Walk mode or Run mode. If equals 0, then this mode is disabled while the monster is moving.

noGfxHitTest - Boolean Field. If equals 1, then enable the mouse selection bounding box functionality around the monster. If equals 0, then the monster cannot be selected by the mouse.
htTop - Define the pixel top offset around the monster for the mouse selection bounding box functionality. This field relies on the “noGfxHitTest” field being enabled.
htLeft - Define the pixel left offset around the monster for the mouse selection bounding box functionality. This field relies on the “noGfxHitTest” field being enabled.
htWidth - Define the pixel right offset around the monster for the mouse selection bounding box functionality. This field relies on the “noGfxHitTest” field being enabled.
htHeight - Define the pixel bottom offset around the monster for the mouse selection bounding box functionality. This field relies on the “noGfxHitTest” field being enabled.

restore - Determines if the monster should be placed on the inactive list, to be saved when the level unloads. If equals 0, then do not save the monster. If equals 1, then rely on other checks to determine to save the monster. If equals 2, then force save the monster.

automapCel - Controls what index of the Automap tiles to use to display this monster on the Automap. This field relies on the “noMap” field being disabled.
noMap - Boolean Field. If equals 1, then the monster will not appear on the Automap. If equals 0, then the monster will normally appear on the Automap.

noOvly - Boolean Field. If equals 1, then no looping overlays will be drawn on the monster. If equals 0, then overlays will be drawn on the monster. (See Overlay.txt)

isSel - Boolean Field. If equals 1, then the monster is selectable and can be targeted. If equals 0, then the monster cannot be selected.
alSel - Boolean Field. If equals 1, then the player can always select the monster, regardless of being an ally or enemy. If equals 0, then ignore this.
noSel - Boolean Field. If equals 1, then the player can never select the monster. If equals 0, then ignore this.
shiftSel - Boolean Field. If equals 1, then the player can target this monster when holding the Shift key and clicking to use a skill. If equals 0, then the monster cannot be targeted while the player is holding the Shift key.
corpseSel - Boolean Field. If equals 1, then the monster’s corpse can be with the mouse cursor. If equals 0, then the monster’s corpse cannot be selected with the mouse cursor.

isAtt - Boolean Field. If equals 1, then the monster can be attacked. If equals 0, then the monster cannot be attacked.
revive - Boolean Field. If equals 1, then the monster is allowed to be revived by the Necromancer Revive skill. If equals 0, then the monster cannot be revived by the Necromancer Revive skill.
limitCorpses - Boolean Field. If equals 1, then the monster's corpse will be placed into a pool with all other corpses with this field checked. Once that pool reaches the max (50), the corpses at the beginning of the pool start getting removed.
critter - Boolean Field. If equals 1, then the monster will be flagged as a critter, which gives some special case handling such as not creating impact sounds and differently handling its spawn placement in presets. If equals 0, then ignore this.
small - Boolean Field. If equals 1, then the monster will be classified as a small type, which can affect what types of missiles can be used on the monster (Example: Barbarian Grim Ward size) or how the monster is knocked back. If equals 0, then ignore this. If this field is enabled, then the "large" field should be disabled, to avoid confusion.
large - Boolean Field. If equals 1, then the monster will be classified as a large type, which can affect what types of missiles can be used on the monster (Example: Barbarian Grim Ward size) or how the monster is knocked back. If equals 0, then ignore this. If this field is enabled, then the "small" field should be disabled, to avoid confusion.
soft - Boolean Field. If equals 1, then the monster's corpse is classified as soft bodied, meaning that its corpse can be used by certain corpse skills such as Barbarian Find Potion, Find Item, or Grim Ward. If equals 0, then the monster's corpse cannot be used for these skills.
inert - Boolean Field. If equals 1, then the monster will never attack its enemies. If equals 0, then ignore this.

objCol - Boolean Field. If equals 1 and the monster class is "barricadedoor", "barricadedoor2", or "evilhut", then the monster will place an invisible object with collision. If equals 0, then ignore this.
deadCol - Boolean Field. If equals 1, then the monster's corpse will have collision with other units. If equals 0, then the monster's corpse will not have collision.

unflatDead - Boolean Field. If equals 1, then ignore the corpse draw order for rendering the sprite on top of others, while the monster is dead. If equals 0, then the monster's corpse will have a normal corpse draw order.
Shadow - Boolean Field. If equals 1, then the monster will project a shadow on the ground. If equals 0, then the monster will not project a shadow.
noUniqueShift - Boolean Field. If equals 1 and the monster is a Unique monster, then the monster will not have random color palette transform shifts. If equals 0, then the non-Unique monster will have random color palette transform shifts.
compositeDeath - Boolean Field. If equals 1, then the monster's Death Mode and Dead mode will make use of its component system. If equals 0, then the monster will default to using the Hand-To-Hand weapon class and no component system.

localBlood - Controls the color of the monster's blood based on the region locale. If equals 0, then do not change the blood to green and keep it red. If equals 1, then change the monster's special components to use the green blood locale. If equals 2, then change the blood to green.
Bleed - Controls if the monster will create blood missiles. If equals 0, then the monster will never bleed. If equals 1, then the monster will randomly create the "blood1" or "blood2" missiles when hit. If equals 2, then the monster will randomly create the "blood1", "blood2", "bigblood1", or "bigblood2" missiles when hit.

Light - Controls the monster's minimum Light Radius size (measured in grid sub-tiles)
light-r - Controls the red color value of the monster's Light Radius (Uses a value from 0 to 255)
light-g - Controls the green color value of the monster's Light Radius (Uses a value from 0 to 255)
light-b - Controls the blue color value of the monster's Light Radius (Uses a value from 0 to 255)

Utrans & Utrans(N) & Utrans(H) - Modifies the color palette transform for the monster respectively in Normal, Nightmare, and Hell difficulty.

Code	Description
0	Cold
1	Poison
2	Level 0
3	Level 1
4	Level 2
5	Level 3
6	Level 4
7	Miscellaneous
255	Special case handler. If hostile, then select the Cold transform. If not hostile, then select the Poison transform.

InfernoLen - The frame length to hold the channel cast time of the inferno skill. This is used for when the monster has the "inferno" state, or for Diablo when he is using the "DiabLight" skill.
InfernoAnim - The exact frame in the channel animation to loop back and start at again
InfernoRollback - The exact frame in the channel animation to determine when to roll back to the "InfernoAnim" frame

ResurrectMode - Controls which monster mode to set on the monster when it is resurrected

Code	Description
DT	Death
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
xx	Sequence
RN	Run

ResurrectSkill - Controls what skill should the monster use when it is resurrected (See skills.txt).
SpawnUniqueMod - Controls what unique modifier the monster should always spawn with (See monumod.txt).

MonType.txt

Overview

This file handles the classification, naming conventions and element of monsters

This is used by the monstats.txt data file

Data Fields

type - Defines the unique monster type ID
equiv1 (to equiv3) - Points to the index of another Monster Type to reference as a parent. This is used to create a hierarchy for Monster Types where the parents will have more universal settings shared across the related children
strplur - Uses a string for the plural form of the monster type. This is used for the **“**descfunc” code 22 function from the ItemStatCost.txt file, based on the monster type selected.
element - Defines the monster’s element type. This can be used for the Necromancer’s Raise Skeletal Mage skill for determining what elemental type a Skeletal Mage should be based on the monster it was raised from (If the monster has no element, then the skeletal mage element will be randomly selected).

Code	Description
(empty)	Any Element
pois	Poison
cold	Cold
fire	Fire
ltng	Lightning

monumod.txt

Overview

This file controls the different monster modifiers for special monsters, including Unique and Champion monsters

Any column field name starting with “*” is considered a comment field and is not used by the game

The “(N)” text in field names signifies to use that specific value for games in Nightmare difficulty
The “(H)” text in field names signifies to use that specific value for games in Hell difficulty

Data Fields

uniquemod - This is a reference field to define the monster modifier
id - Defines the unique numeric ID for the monster modifier. Used as a reference in other data files.
enabled - Boolean Field. If equals 1, then this monster modifier will be an available option for monsters to spawn with. If equals 0, then this monster modifier will never be used.
version - Defines which game version to use this monster modifier (<100 = Classic mode | 100 = Expansion mode)

xfer - Boolean Field. If equals 1, then this monster modifier can be transferred from the Boss monster to his Minion monsters, including auras. If equals 0, then the monster modifier will never be transferred.
champion - Boolean Field. If equals 1, then this monster modifier will only be used by Champion monsters. If equals 0, then the monster modifier can be used by any type of special monster.
fPick - Controls if this monster modifier is allowed on the monster based on the function code and the parameters it checks.

Code	Description
0 (or empty)	Ignore this
1	Monster class must have the Attack 1 mode (checks “mA1” field from monstats.txt)
2	Monster class cannot be flagged as Melee (checks “IsMelee” field from monstats.txt) Monster class cannot have the No Multishot flag (checks “nomultishot” field from monstats.txt)
3	Monster class must have the Walk mode (checks “mWL” field from monstats.txt)

exclude1 & exclude2 - This controls which Monster Types should not have this monster modifier (Uses the “type” field from MonType.txt)

cpick & cpick (N) & cpick (H) - Modifies the chances that this monster modifier will be chosen for a Champion monster, compared to other monster modifiers. The higher the value is, then the more likely this modifier will be chosen. This value acts as a numerator and a denominator. All “cpick” values get summed together to give a total denominator, used for the random roll. For example, if there are 3 possible monster modifiers, and their “cpick” values are 3, 4, 6, then their chances to be chosen are 3/13, 4/13, and 6/13 respectively.

upick & upick (N) & upick (H) - Modifies the chances that this monster modifier will be chosen for a Unique monster, compared to other monster modifiers. The higher the value is, then the more likely this modifier will be chosen. This value acts as a numerator and a denominator. All “upick” values get summed together to give a total denominator, used for the random roll. For example, if there are 3 possible monster modifiers, and their “upick” values are 3, 4, 6, then their chances to be chosen are 3/13, 4/13, and 6/13 respectively.

constants - These values control a special list of numeric parameters for special monsters. The row that each constant appears in the data file is unrelated. You can treat this column almost like a separate data file that controls other aspects of special monsters. See the description next to each value for more specific clarification on each constant.

monsounds.txt

Overview

This file controls the sounds that play for each of a monster's actions

This file relies on sounds from sounds.txt

This file is used by the monstats.txt file

Data Fields

Id - Defines the unique name ID for the monster sound

Attack1 & Attack2 - Play this sound when the monster performs Attack 1 and Attack 2, respectively. Points to a "Sound" value in the sounds.txt file.
Weapon1 & Weapon2 - Play this sound when the monster performs Attack 1 and Attack 2, respectively. This acts as an extra sound that can play with the "Attack1" and "Attack2" sounds. Points to a "Sound" value in the sounds.txt file.
Att1Del & Att2Del - Controls the amount of game frames to delay playing the "Attack1" and "Attack2" sounds, respectively.
Wea1Del & Wea2Del - Controls the amount of game frames to delay playing the "Weapon1" and "Weapon2" sounds, respectively.
Att1Prb & Att2Prb - Controls the percent chance (out of 100) to play the "Attack1" and "Attack2" sounds, respectively.
Wea1Vol & Wea2Vol - Controls the volume of the "Weapon1" and "Weapon2" sounds, respectively. Uses a range between 0 to 255, where 255 is the maximum volume.

HitSound - Play this sound when the monster gets hit or knocked back. Points to a "Sound" value in the sounds.txt file.
DeathSound - Play this sound when the monster dies. Points to a "Sound" value in the sounds.txt file.
HitDelay - Controls the amount of game frames to delay playing the "HitSound" sound
DeaDelay - Controls the amount of game frames to delay playing the "DeathSound" sound

Skill1 (to Skill4) - Play this sound when the monster uses the skill linked in the related "Skill#" field from the monstats.txt file. Points to a "Sound" value in the sounds.txt file.

Footstep - Play this sound while the monster is walking or running. Points to a "Sound" value in the sounds.txt file.
FootstepLayer - Play this sound while the monster is walking or running. This acts as an extra sound that can play with the "Footstep" sound. Points to a "Sound" value in the sounds.txt file.
FsCnt - Controls the footstep count which is used to determine how often to play the "Footstep" and "FootstepLayer" sound. A higher value would mean that the sounds would play more often.
FsOff - Controls the footstep offset which is used for calculating when to play the next "Footstep" and "FootstepLayer" sound, based on the current animation frame and the animation rate. A higher value would mean that the sounds would play less often.
FsPrb - Controls the probability to play the "Footstep" and "FootstepLayer" sound, with a random chance out of 100.

Neutral - Play this sound while the monster is in Neutral, Walk, or Run mode. Also play this sound when the monster "Id" equals "vulture1" and while the monster is in Skill1 mode. Also play this sound when the monster "Id" equals "batdemon1" and while the monster is in Skill4 mode. Points to a "Sound" value in the sounds.txt file.
NeuTime - Controls the amount of game frames to delay between re-playing the "Neutral" sound after it finishes.

Init - Play this sound when the monster spawns and is not dead and is not playing its Neutral sound. Points to a "Sound" value in the sounds.txt file.
Taunt - Play this sound when the server requests that the monster should play its Taunt. This is typically used for quest or story related moments. Points to a "Sound" value in the sounds.txt file.
Flee - Play this sound when the monster is told to flee. This depends on when the monster AI is told to play this sound. Points to a "Sound" value in the sounds.txt file.

CvtMo1 (to CvtMo3) - This is used to convert the mode for playing the sound. This field defines the original mode that the monster is using. (See MonMode.txt for the list of possible inputs)
CvtSk1 (to CvtSk3) - Defines the skill that the monster is using. If the monster uses a specific skill, then the game can change the monster's mode for sound functionalities to another mode to change how sounds are generally handled. Points to a "skill" in the skills.txt file.
CvtTgt1 (to CvtTgt3) - Defines the mode to convert the sound to when the monster is using the relative skill from the "CvtSk#" field. This does not actually change the monster's actual mode but only what mode that sounds think the monster is using. (See MonMode.txt for the list of possible inputs)

npc.txt

Overview

This file controls how each town NPC manipulates their store prices

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

npc - Points to the matching “Id” value in the monstats.txt file. This should not be changed.

buy mult - Used to calculate the item’s price when it is bought by the NPC from the player. This number is a fraction of 1024 in the following formula: [cost] * [buy mult] / 1024
sell mult - Used to calculate the item’s price when it is sold by the NPC to the player. This number is a fraction of 1024 in the following formula: [cost] * [sell mult] / 1024
rep mult - Used to calculate the cost to repair an item. This number is a fraction of 1024 in the following formula: [cost] * [rep mult] / 1024. This is then used to influence the repair cost based on the item durability and charges.

questflag A (to questflag C) - If the player has this quest flag progress, then apply the relative additional price calculations

Code	Quest Progress
0	Act 1 Prologue Seen
1	Den of Evil Completed
2	Sisters’ Burial Grounds Completed
3	Tools of the Trade Completed
4	The Search for Cain Completed
5	The Forgotten Tower Completed
6	Sisters to the Slaughter Completed
7	Act 1 Traversed
8	Act 2 Prologue Seen
9	Radament’s Lair Completed
10	The Horadric Staff Completed
11	Tainted Sun Completed
12	Arcane Sanctuary Completed
13	The Summoner Completed
14	The Seven Tombs Completed
15	Act 2 Traversed
16	Act 3 Prologue Seen
17	Lam Esen's Tome Completed
18	Khalim's Will Completed
19	Blade of the Old Religion Completed
20	The Golden Bird Completed
21	The Blackened Temple Completed
22	The Guardian Completed
23	Act 3 Traversed
24	Act 4 Prologue Seen
25	The Fallen Angel Completed
26	Terror's End Completed
27	Hell's Forge Completed
28	Act 4 Traversed
29	Rogue Warning Complete
30	Guard in Town Warning Complete
31	Guard in Desert Warning Complete
32	Dark Wanderer Seen
33	Angel Warning Complete
34	Act 5 Prologue Seen
35	Siege on Harrogath Completed
36	Rescue on Mount Arreat Completed
37	Prison of Ice Completed
38	Betrayal of Harrogath Completed
39	Rite of Passage Completed
40	Eve of Destruction Completed
41	Respecialization from Akara is Completed

questbuymult A (to questbuymult C) - Same functionality as the “buy mult” field, except this relies on the “questflag” field and applies after the “buy mult” field calculation
questsellmult A (to questsellmult C) - Same functionality as the “sell mult” field, except this relies on the “questflag” field and applies after the “sell mult” field calculation
questrepmult A (to questrepmult C) - Same functionality as the “rep mult” field, except this relies on the “questflag” field and applies after the “rep mult” field calculation

max buy & max buy (N) & max buy (H) - Sets the maximum price that the NPC will pay, when the player sells an item in Normal Difficulty, Nightmare Difficulty, and Hell Difficulty, respectively

objects.txt

Overview

This file controls the functionalities of all objects found in area levels

The order of each object defined in this file will convey what ID value it has, and thus should not be changed

This file uses the following files: ObjMode.txt, objpreset, ObjType.txt, shrines.txt

Any column field name starting with "*" is considered a comment field and is not used by the game

Objects are always set to be using a specific mode, which controls which fields to use for functionalities. There are 8 possible Object modes, each tied to an ID number. Specific fields are numbered to match each of these modes, meaning that the object will use that specific field number when in a certain mode (See ObjMode.txt)

Number	Object Mode	Token
0	Neutral	NU
1	Operating	OP
2	Opened	ON
3	Special 1	S1
4	Special 2	S2
5	Special 3	S3
6	Special 4	S4
7	Special 5	S5

Data Fields

Class - Defines the unique type class of the object which is used to reference this object. These are also defined in the objpreset.txt file.
Name - String key. Used as the display name of the object when being highlighted by the player.
Token - Determines what files to use to display the graphics of the object. These are defined by the ObjType.txt file.

Selectable0 (to Selectable7) - Boolean Field. If equals 1, then the object can be selected by the player and highlighted when hovered on by the mouse cursor. If equals 0, then the object cannot be selected and will not highlight when the player hovers the mouse over it. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).

SizeX & SizeY - Controls the amount of sub tiles that the object occupies using X and Y coordinates. This is generally used for measuring the object's size when trying to spawn objects in rooms and controlling their distances apart.

FrameCnt0 (To FrameCnt7) - Controls the frame length of the object's mode. If this equals 0, then that mode will be skipped. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
FrameDelta0 (to FrameDelta7) - Controls the animation frame rate of how many frames to update per delta (Measured in 256ths). Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
CycleAnim0 (to CycleAnim7) - Boolean Field. If equals 1, then the object's current animation will loop back to play again when it finishes. If equals 0, then the object will generally play the Opened mode after playing the Operating mode. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
Lit0 (to Lit7) - Controls the Light Radius distance value for the object. If this value equals 0, then the object will not emit a Light Radius. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
BlocksLight0 (to BlocksLight7) - Boolean Field. If equals 1, then the object will draw a shadow. If equals 0, then the object will not draw a shadow. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
HasCollision0 (to HasCollision7) - Boolean Field. If equals 1, then the object will have collision. If equals 0, then the object will not have collision, and units can walk through it. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
IsAttackable0 - Boolean Field. If equals 1, then the player can target this object to be attacked, and the player will use the Kick skill when operating the object. If the object has the Class equal to "CompellingOrb" or "SoulStoneForge", then instead of using the Kick skill, players will use the Attack skill when operating the object. If equals 0, then ignore this, and the player will not use a skill or animation when operating the object.
Start0 (to Start7) - Controls the frame for where the object will start playing the next animation. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
EnvEffect - Boolean Field. If equals 1, then enable the object to update its mode based on the game's time of day. This can mean that when the object is spawned, and it is current day time and the object is in Opened or Operating mode, then it will reset back to Neutral mode. Also, if the current time is dusk, night, or dawn and the object is in Neutral mode, then it will change to Operating mode. If equals 0, then the object will not update its mode based on the time of day.
IsDoor - Boolean Field. If equals 1, then the object will be treated as a door when the game handles its collision, animation properties, tooltips, and commands. If equals 0, then ignore this.
BlocksVis - Boolean Field. If equals 1, then the object will block the player's line of sight to see anything beyond the object. If equals 0, then ignore this. This field relies on the "IsDoor" field being enabled.
Orientation - Determines the object's orientation type, which can affect mouse selection priority of the object when a unit is being rendered in front of or behind the object (such as a door object covering a unit and how the mouse selection should handle that). This also affects the randomization of the coordinates when spawning the object near the edge of a room.

Code	Description
0 (or other #)	Center
1	Right
2	Left

OrderFlag0 (to OrderFlag7) - Controls how the object's sprite is drawn, which can affect how it is displayed in Perspective game camera mode. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).

Code	Description
0	Do nothing
1	Flat floor
2	Wall

PreOperate - Boolean Field. If equals 1, then enable a random chance that the object will spawn in already in Opened mode. The game will choose a 1/14 chance that this can happen when the object is spawned. If equals 0, then ignore this.
Mode0 (to Mode7) - Boolean Field. If equals 1, then confirm that this object has the correlating mode. If equals 0, then this object will not have the correlating mode. This flag can affect how the object functions work. Each field is numbered, correlating to 1 of 8 Object Modes that the object uses (See Overview section, or ObjMode.txt).
Xoffset & Yoffset - Controls the offset values in the X and Y directions for the object's visual graphics. This is measured in game pixels.

Draw - Boolean Field. If equals 1, then draw the object's shadows. If equal's 0, then do not draw the object's shadows.

Red - Controls the Red color gradient of the object's Light Radius. This field depends on the "Lit#" field having a value greater than 0.
Green - Controls the Green color gradient of the object's Light Radius. This field depends on the "Lit#" field having a value greater than 0.
Blue - Controls the Blue color gradient of the object's Light Radius. This field depends on the "Lit#" field having a value greater than 0.

HD - Boolean Field. If equals 1, then the object will be flagged to have a Head composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
TR - Boolean Field. If equals 1, then the object will be flagged to have a Torso composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
LG - Boolean Field. If equals 1, then the object will be flagged to have a Legs composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
RA - Boolean Field. If equals 1, then the object will be flagged to have a Right Arm composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
LA - Boolean Field. If equals 1, then the object will be flagged to have a Left Arm composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
RH - Boolean Field. If equals 1, then the object will be flagged to have a Right Hand composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
LH - Boolean Field. If equals 1, then the object will be flagged to have a Left Hand composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
SH - Boolean Field. If equals 1, then the object will be flagged to have a Shield composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
S1 (to S8) - Boolean Field. If equals 1, then the object will be flagged to have a Special # composite piece, and the game will use the component system to handle the object's mouse selection collision box. If equals 0, then ignore this.
TotalPieces - Defines the total amount of composite pieces. If this value is greater than 1, then the game will treat the object with the multiple composite piece system, and the player can hover the mouse over and select the object's different components.

SubClass - Determines the object's class type by declaring a specific value. This is used by the various functions ("InitFn", "OperateFn", "PopulateFn") for knowing how to handle specific types of objects.

Code	Description
0	None
1	Shrine
2	Obelisk
4	Portal (With a source & destination)
8	Trappable
16	Fixed Portal
32	Well
64	Waypoint
128	Hidden

Xspace & Yspace - Controls the X and Y distance delta values between adjacent objects when they are being populated together. This field is only used by the Populate Function ("PopulateFn") values 3 and 4, for the Add Barrels and Add Crates functions.
NameOffset - Controls the vertical offset of the name tooltip's position above the object when the object is being selected. This is measured in pixels.
MonsterOK - Boolean Field. If equals 1, then if a monster operates the object, then the object will run its operate function. If equals 0, then then if a monster operates the object, then the object will not run its operate function.
ShrineFunction - Controls what shrine function to use (See "Code" field in shrines.txt) when the object is told to do its Skill command.
Restore - Boolean Field. If equals 1, the game will restore the object in an inactive state when the area level repopulates after a player loads back into it. If equals 0, then the game will not restore the object.

Parm0 (to Parm4) - Used as possible parameters for various functions for the object

Lockable - Boolean Field. If equals 1, then the object will have a random chance to spawn with the locked attribute and have a display tooltip name with the "lockedchest" string key. This only works when the object has the Init Function ("InitFn") value equal to 3. If equals 0, then ignore this.
Gore - Controls if an object should call its Populate function ("PopulateFn") when it is chosen as an object that can spawn in a room. Objects with a gore value greater than 2 will not be populated in rooms.
Sync - Boolean Field. If equals 1, then the object's animation rate will always match the "FrameDelta#" field (depending on the object's mode) which means the client and server will have synced animations. If equals 0, then the animation rate will have random visual variation.
Damage - Controls the amount of damage dealt by the object when it performs an Operate Function ("OperateFn") that deals damage such as triggering a pulse trap or an explosion.
Overlay - Boolean Field. If equals 1, then add and remove an overlay on the object based on its current mode. If equals 0, then ignore this. This field will only work with specific object Classes and will use specific Overlays for those objects.

Object Class	Overlay
SpecialChest100 KhalimHeartChest KhalimEyeChest KhalimBrainChest HoradricCubeChest HoradricScrollChest StaffOfKingsChest ConsolationChest	multigleam
SevenTombsReceptacle	horadric_light
TaintedSunShrine	horadric_light

CollisionSubst - Boolean Field. If equals 1, then the game will handle the bounding box around the object for mouse selection. The game will use the object's pixel size and "Left", "Top", "Width", "Height" field values to determine the collision size. If equals 0, then ignore this.
Left - Controls the starting X position offset value for drawing the bounding collision box around the object for mouse selection. This field depends on the "CollisionSubst" field being enabled.
Top - Controls the starting Y position offset value for drawing the bounding collision box around the object for mouse selection. This field depends on the "CollisionSubst" field being enabled.
Width - Controls the ending X position offset value for drawing the bounding collision box around the object for mouse selection. This field depends on the "CollisionSubst" field being enabled.
Height - Controls the ending Y position offset value for drawing the bounding collision box around the object for mouse selection. This field depends on the "CollisionSubst" field being enabled.

OperateFn - Defines a function that the game will use when the player clicks on the object

Code	Description
0	Do nothing
1	Spawn Item And Maybe Monster - General function to operate an object, spawn items. Also can randomly spawn a monster and/or trigger a trap.
2	Shrine - General function for Shrine objects. Uses fields from the shrines.txt file for determining specific Shrine functions.
3	Spawn Item Sometimes - General function to operate the object and spawn random items. Has a 20% chance to spawn a random item. Can also randomly trigger a trap.
4	Chest Operate - General function for opening chest objects and spawning random items. Handles key interaction functionality if the chest object is locked
5	Barrel Operate - General function for breaking barrel objects and randomly spawning items or possibly a monster
6	Quest Tome Operate - Handles updating The Forgotten Tower quest progress
7	Barrel Exploding Operate - Explode the object and also explode adjacent Exploding Barrel object Classes
8	Door Operate - General function for opening and closing door objects
9	Quest Cairn Stone Operate - Handle operating the 5 Cairn Stone objects based on the player's progress for the Search for Cain quest and if the player has the deciphered Scroll of Inifuss item. Also removes the Scroll of Inifuss item once successfully operated.
10	Quest Gibbet Operate - Handle operating the object and updating the player's progress for the Search for Cain quest. This is used for the cage object that Deckard Cain is trapped in.
11	Brazier Operate - Switch the object from Neutral mode to Operating/Opened mode, or vice versa
12	Quest Inifuss Operate - Handle dropping the Bark Scroll item, based on the player's progress for the Search for Cain quest
13	Tiki Operate - Switch the object from Neutral mode to Operating mode, or vice versa
14	Spawn Item - General function to operate an object and have it spawn random items. Can also remove the object's collision and randomly trigger a trap.
15	Town Portal Operate - Controls the Town Portal functionalities, including how to teleport players back to town or to the current level, and handling how players interact with other player Town Portals
16	Trap Door Operate - Open a door type object and then control its level warp capabilities
17	Obelisk 1 - Use the transaction UI if the player has a gem in their inventory, and operate the object
18	Secret Door Operate - Handle operating an object and removing its collision
19	Armor Rack Operate - Activate the object to spawn a random armor item
20	Weapon Rack Operate - Activate the object to spawn a random weapon item
21	Quest Malus Operate - Handle dropping the Horadric Malus item, based on the player's progress for the Tools of the Trade quest
22	Well Operate - Handle healing the player and keeping track of the charges and regeneration of charges for the well object
23	Waypoint Operate - Handle activating a waypoint object and using the Waypoint UI when clicking on an activated waypoint object
24	Quest Tainted Sun Altar Operate - Create the Amulet of the Viper item and other treasure items based on The Horadric Staff quest progress and the number of players in the game. Also update the progress for the Tainted Sun quest.
25	Quest Seven Tombs Receptacle Operate - Handle using the Horadric Staff item with the transaction UI to operate the object
26	Bookshelf Operate - Randomly create either tomes or scrolls of Identify or Town Portal
27	Teleport Pad Operate - Teleport the player to another part of the room
28	Quest Lam Esens Tome Operate - Handle dropping the Lam Esen's Tome item, based on the player's progress for the Lam Esen's Tome quest
29	Breakable Operate - Animate the object and remove its collision
30	Exploding - Create an explosion around the object
31	Quest Gidbinn Operate - Handle dropping the Decoy Gidbinn item, based on the player's progress for the Blade of the Old Religion quest
32	Player Bank Operate - Control accessing the Stash UI while in town for the Bank object Class
33	Wirt Spurt - Create the Wirt's leg item and animate the object
34	Arcane Portal - Control how the warp object transitions the player from the Palace Cellar Level 3 to the Arcane Sanctuary
35	Return null
36	Return null
37	Return null
38	Return null
39	Quest Horadric Cube Chest Operate - Create the Horadric Cube item and other treasure items based on The Horadric Staff quest progress and the number of players in the game
40	Quest Horadric Scroll Chest Operate - Create the Horadric Scroll item and other treasure items based on The Horadric Staff quest progress and the number of players in the game
41	Quest Staff of Kings Chest Operate - Create the Staff of Kings item and other treasure items based on The Horadric Staff quest progress and the number of players in the game
42	Quest Arcane Tome Operate - Handles updating The Arcane Sanctuary quest progress
43	One Way Portal Operate - Controls the functionalities of the "DurielPortal" one way warp object
44	Quest Beneath The City Stairs Operate - Handles warp object operates based on the Khalim's Flail quest progress
45	Quest Beneath The City Lever Operate - Handles operating an object based on the Khalim's Flail quest progress
46	Hell Gate Operate - Handles how to transition the player to Act 4 based on The Guardian quest progress
47	Stairs Operate - Handles how the stairs object opens or warp the player to another level
48	Jack In The Box Operate - Handles the operating the object and having it spawn items and set its mode to Special 2.
49	Quest Soulstone Forge Operate - Handle operating the object based on The Hellforge quest progress and how it spawns items. Also remove the Hellforge Hammer weapon from the player.
50	Quest Mephisto Door Operate - Handles how the stairs object opens or warp the player to another level
51	Delay Spawn Operate - Waits until the object is done operating before updating events
52	Quest Diablo Seal Operate - Handle operating a Diablo Seal object while also tracking the progress on the other related Diablo Seal objects (5 in total).

53	Quest Compelling Orb Operate - Handle operating the object based on the Khalim's Flail quest progress and The Blackened Temple progress. Also remove the Khalim's Flail weapon from the player.
54	Quest Diablo Seal 1 Operate - Handle operating a Diablo Seal object Class and getting a spawn point for monsters. Also calls function 52.
55	Quest Diablo Seal 3 Operate - Handle operating a Diablo Seal object Class and getting a spawn point for monsters. Also calls function 52.
56	Quest Diablo Seal 5 Operate - Handle operating a Diablo Seal object Class and getting a spawn point for monsters. Also calls function 52.
57	Quest Khalim Heart Chest Operate - Create the Khalim's Heart item and other treasure items based on the Khalim's Flail quest progress and the number of players in the game
58	Quest Khalim Eye Chest Operate - Create the Khalim's Eye item and other treasure items based on the Khalim's Flail quest progress and the number of players in the game
59	Quest Khalim Brain Chest Operate - Create the Khalim's Brain item and other treasure items based on the Khalim's Flail quest progress and the number of players in the game
60	Return null
61	Town Gate - Handles how the gate object opens and closes
62	Handles the modes of one of the Ancient's statues based on the player's progress of the Rite of Passage quest
63	Same as function 62
64	Same as function 62
65	Quest Ancient Altar Operate - Handle displaying quest text and disabling the player's town portals, based on the player's progress of the Rite of Passage quest.
66	Quest Ancient Gateway Operate - Handle opening the door object based on the player's progress of the Rite of Passage quest.
67	Quest Frozen Anya Operate - Handles the object displaying quest text or validating that the player has the Malah's Potion item and updating the Prison of Ice quest
68	Evil Urn - Handle triggering a trap from the object
69	Quest Ancient Invisible Operate - Handle displaying the A5Q6InitAncients string conversation text based on the player's progress of the Rite of Passage quest.
70	Quest Last Exit Operate - Handle transitioning the player to the from the Throne of Destruction level to the Worldstone Chamber level
71	Quest Summit Door Operate - Handle opening the door object based on the player's progress of the Rite of Passage quest.
72	Quest Player Last Portal Operate - Handle transitioning the player to completing the game after completing the Destruction's End quest
73	Quest Tyrael Portal To Expansion Operate - Handle transitioning the player to Act 5 after completing the Act 4 Terror's End quest

PopulateFn - Defines a function that the game will use to spawn this object

Code	Description
0	Do not spawn the object
1	Add Clumped Group - Handles creating multiple of these objects randomly in a room, based on the object's size and Class. This function only handles specific object classes such as caskets, urns, and baskets.
2	Add Single Shrine - Handles the creation of a shrine object
3	Add Simple Objects - Handles randomly spawning the object in a room, based on the object's size.
4	Add Barrels - Handle creating multiple barrel or exploding barrel Class objects in a room.
5	Add Crates - Handle creating multiple crate or urn Class objects in a room.
6	Add Corpse - Use function 3 to handle spawning the object. Also call a random chance to spawn the "Flies" object class on top of the objects that spawn.
7	Add Staked Corpses - Handles how to specifically spawn the "RogueCorpse1" and "RogueCorpse2" objects, based on their sizes and the locations in the room. Also call a random chance to spawn the "Flies" object class on top of the objects that spawn.
8	Add Well - Handles the creation of one of these objects randomly in a room based on the object's size. A level can have a maximum of 4 these objects that spawn.
9	Add One - Handles the creation of one of these objects randomly in a room based on the object's size.

InitFn - Defines a function to control how the object works while active and when initially activated by a player

Code	Description
0	Do nothing
1	ObjectInitShrine - General function for determining which type of Shrine function to pick for the Shrine object. (See shrines.txt file for a list of shrine types) This also uses the "Parm0" field to define the Shrine Type • If equals 0, default to health shrine • If equals 1, then use Health Shrine • If equals 2, then use Mana Shrine • If equals 3, then pick a random stats shrine with a 10% chance to spawn a surprise shrine
2	ObjectInitTrappable - Handle a random chance to give the object 1 of the 9 random traps. This random chance depends on the area level's monster level.
3	ObjectInitChest - Run function 1, and also determine if the object should be Locked or not. The random chance to make the object Locked depends on the area level's monster level.
4	QuestObjectTowerTomeInit - If The Forgotten Tower quest is active, then set the object to run in Special 0 Mode.
5	Do nothing
6	QuestObjectStoneInit - Sets the object's mode to be Opened or Neutral, depending on the progress with the Portal to Tristram for the Search for Cain quest.
7	QuestObjectGibbetInit - Sets the object's mode, depending on the progress with Cain's Cage for the Search for Cain quest.
8	ObjectInitDungeonTorch - Sets the object's mode to Opened
9	Quest Object Inifuss Init - Sets the object's mode, depending on the progress with the Tree for the Search for Cain quest.
10	ObjectInitBonfire - If the current level is Act 1 Rogue Encampment, then tell the object to do a periodic skill, otherwise set the object mode to Opened.
11	ObjectInitTownPortal - Initializes the object's mode and adds the level ID as an attribute to keep track of.
12	ObjectInitPermanentPortal - Handles specific level transitions for permanent portals found throughout the game
13	QuestObjectStoneSoundInit - Attaches the object to the Search for Cain quest functions
14	ObjectInitDungeonTorch2 - Sets the object's mode to Operating
15	QuestObjectMalusInit - Attaches the object to the Tools of the Trade quest functions
16	ObjectInitWell - Sets the object's attributes for a well including amount of charges This also uses the "Parm2" field to define the amount of Life healed
17	ObjectInitWaypoint - Handles setting up the waypoint mechanic to the object for the current area level

18	QuestObjectJerhyn1Init - Handle where to place Jerhyn (near the palace entrance) based on Arcane Sanctuary quest progress
19	QuestObjectJerhyn2Init - Handle where to place Jerhyn (inside the palace) based on The Seven Tombs quest progress
20	QuestObjectTaintedSunAltarInit - Attaches the object to the Tainted Sun quest functions
21	QuestObjectSevenTombsReceptacleInit - Setup the object to be a receptacle for the Horadric Staff, based on The Seven Tombs quest progress
22	ObjectInitFire - Setup the object to act as fire
23	QuestObjectLamEsensTomeInit - Attaches the object to the Lam Esen's Tome quest functions
24	ObjectInitTrap1 - Handle setting up the object frame count and making sure it has full stats
25	QuestObjectGidbinnInit - Attaches the object to the Blade of the Old Religion quest functions
26	TestObjectInit - Sets the object's mode to Operating
27	ObjectInitTrappablePoison - Sets up the random chance of 333/1000 for the object to have a trap that creates a poison nova
28	ObjectInitGold - Create a random amount of gold piles (between 1 to 10) in random locations around the object
29	QuestObjectInitArcanePortal - Setup the object to link area levels between the Palace Cellar Level 3 and the Arcane Sanctuary
30	QuestObjectHaremBlockerInit - Setup the object's collision based on the Arcane Sanctuary quest progress
31	QuestObjectHoradricCubeChestInit - Sets up information about the object
32	QuestObjectHoradricScrollChestInit - Sets up information about the object
33	QuestObjectStaffOfKingsChestInit - Sets up information about the object
34	ObjectInitHellTorch - Randomly set the object's mode to Operating
35	Return null
36	Return null
37	QuestObjectDurielPassagewayInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the The Seven Tombs quest
38	QuestObjectTyraelDoorInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the The Seven Tombs quest
39	QuestObjectGidbinnTownAltarInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the The Blade of the Old Religion quest
40	Return null
41	QuestObjectBeneathTheCityStairsInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the Khalim's Flail quest
42	QuestObjectBeneathTheCityLeverInit - If the Khalim's Flail quest is complete, then set the object's mode to Opened
43	QuestObjectDarkWandererInit - Create the "darkwanderer" monster and order to walk to the object's location. This depends on the players character save from having witnessed this event before.
44	QuestObjectInitHellGate - Decide between setting the object's mode to Opened or Neutral, based on the progress of the The Guardian quest
45	QuestObjectMephistoBridgeInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the The Guardian quest. If the object is not Opened, then also tell it to do its unique event.
46	ObjectTrappedSoulInit - Determine where to spawn the "trappedsoul1" and "trappedsoul2" monster classes in the area level.
47	QuestObjectForgottenTowerChestInit - Decide between setting up the chest object, relying on the Forgotten Tower quest being in progress
48	QuestObjectSoulstoneForgeInit - Decide between setting the object's mode to Opened or Neutral, based on the progress of the Hell's Forge quest
49	QuestObjectHratliStartInit - Handle placing Hratli near the starting point of Act 3, based on the player's Act 3 prologue progress
50	QuestObjectHratliEndInit - Handle placing Hratli near his forge, if the player has progressed past the Act 3 prologue
51	ObjectJackInTheBoxInit - If the object is in Opened or Opening mode, then tell the object to do a periodic item skill event
52	QuestObjectNatalyaInit - Handle placing Natalya at her location based on the player's progress of The Guardian quest
53	QuestObjectMephistoDoorInit - Handle setting the object to Opened mode based on the player's progress of destroying the orb for The Blackened Temple quest
54	QuestObjectCainStartInit - Handle creating the Cain unit in the Rogue Encampment based on the player's progress of The Search for Cain quest
55	QuestObjectDiabloStartInit - Handle the spawning event of Diablo based on the player's progress of activating the seal objects in the Chaos Sanctuary
56	QuestObjectDiabloSealInit - Do nothing
57	ObjectInitBetterChest - Initialize the chest object, and give it the special magical property
58	ObjectInitFissure - Tell the object to do a periodic skill event at random times
59	ObjectVileDoggieInit - If the object is in Neutral mode, then set the object to Operating mode and tell it to do a unique event
60	QuestObjectCompellingOrbInit - Set the object to Opened based on the progress of The Blackened Temple quest
61	QuestObjectCainPortalInit - Set the object to Operating mode and tell it to do a unique event
62	QuestCagedWussie1Init - Spawn the "act5pow" units based on the player's progress of the Rescue on Mount Arreat quest
63	QuestMoeInit - Setup the Korlic statue object with quest data based on the Right of Passage quest progress
64	QuestLarryInit - Setup the Madawc statue object with quest data based on the Right of Passage quest progress
65	QuestCurlyInit - Setup the Talic statue object with quest data based on the Right of Passage quest progress
66	QuestAnyaInsideTownInit - Handle for creating the Anya NPC in town, based on the progress of the Prison of Ice quest
67	QuestAnyaOutsideTownInit - Handle this object during the progress of the Prison of Ice quest and tell it to do its unique event
68	QuestNihlathakInsideTownInit - Create the Nihlathak NPC in town, based on the progress of the Prison of Ice quest
69	QuestNihlathakOutsideTownInit - Create the "Nihlathak Boss" super unique monster, based on the progress of the Prison of Ice quest
70	QuestLarzukStartInit - Do nothing
71	QuestLarzukEndInit - Object placeholder to create the "Larzuk" NPC in town
72	QuestAncientTomeInit - Set the tome object mode to Opened or Neutral based on the progress of The Rite of Passage quest
73	QuestAncientGatewayInit - Set the door object mode to Opened or Neutral based on the progress of The Rite of Passage quest
74	QuestFrozenAnyaInit - Handle this object during the progress of the Prison of Ice quest and tell it to do its unique event
75	QuestLastExitInit - Set the Throne of Destruction exit object mode to Operating or Opened based on the progress of the Eve of Destruction quest
76	QuestSummitDoorInit - Set this door object mode to Operating or Opened based on the progress of the Rite of Passage quest
77	QuestPlayerLastPortalInit - Set the last portal object mode to Operating or Opened based on the progress of the Eve of Destruction quest
78	QuestTyraelPortalToExpansionInit - Set this object mode to Operating or Opened based on the progress of the Terror's End quest
79	QuestZooInit - Attempt a random chance based on successfully selecting a "zoo" type monster from the entire list of possible monsters (See monstats.txt). If selected, then send the quest update command to all players, based on the Eve of Destruction quest.

ClientFn - Defines a function that runs on the object from the game's client side.

Code	Description
0	Do nothing
1	Ambient Sound - Always return true

2	Ripple - Tells the object to randomly play between its Operating animation and loop back to its Neutral animation
3	Hell Fire - Same as function 2, except sound will also be processed
4	Drinker - Tells the object to randomly play between its Special 0 animation and loop back to its Neutral animation. Also processes sound.
5	Gesturer - Tells the object to randomly play between its Special 0 / Special 1 animation and loop back to its Neutral animation. Also processes sound.
6	Turner - Tells the object to randomly play between its Special 0 animation and loop back to its Neutral animation. Uses different tick counts than function 4. Also processes sound.
7	Skeleton - Tells the object to randomly play between its Operating animation and loop back to its Neutral animation.
8	Duriel Entrance - If the object is not in Neutral mode then preload the Duriel monster
9	Client Smoke - Controls how the object can be removed from the client based on distance to a player and if the object has a specific tick count.
10	Bubbles - Tells the object to randomly play between its Operating animation and loop back to its Neutral animation. Uses different tick counts than function 7.
11	Floaters - Always return true
12	Altar - If the object is not in Neutral mode then preload the Ancients statues
13	Invisible Ancient - If the object is in its Neutral mode and the player operating the object has not completed the Rite of Passage quest, then handle the control of operating the object
14	Bonfire - Updates the object's animation modes based on the time of day
15	Frozen Anya - If the object is in Neutral mode then play the "npcalert" overlay.
16	Last Exit - If the object is in its Operating mode, then modify the animation frames
17	Zoo - Handle the creation of monsters if monsters need to be created
18	Keeper - Randomly play the "barbarian_grunt_small_1" sound

RestoreVirgins - Boolean Field. If equals 1, then when the object has been used, the game will not restore the object in an inactive state when the area level repopulates after a player loads back into it. If equals 0, then ignore this.
BlockMissile - Boolean Field. If equals 1, then missiles can collide with this object. If equals 0, then missiles will ignore and fly through this object.
DrawUnder - Controls the targeting priority of the object

Code	Description
0	The object will not change its targeting priority
1	The object's target priority will equal a corpse only when the object is opened
2	The object's target priority always equals a corpse

OpenWarp - Boolean Field. If equals 1, then this object will be classified as an object that can be opened to warp to another area, and the UI will be notified to display a tooltip for opening or entering, based on the object's mode. If equals 0, then ignore this.

AutoMap - Used to display a tile in the Automap to represent the object. Defines which cell number to use in the tile list for the Automap. If this value equals 0, then this object will not display on the Automap. (See Automap.txt)

objgroup.txt

Overview

This file controls what group of possible Objects to spawn in a part of an area level.

This file uses the following files: objects.txt

The order of each Object Group defined in this file will convey what ID value it has, which is referenced by the Levels.txt file
The order of these Object Groups should not be changed

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

GroupName - This is a reference field to define the Object Group name

ID0 (to ID7) - Uses the "Id" field from objects.txt, which assigns an Object to this Object Group
DENSITY0 (to DENSITY7) - Controls the number of Objects to spawn in the area level. This is also affected by the Object's populate function defined by the "PopulateFn" field from the objects.txt file. The maximum value allowed is 128.
PROB0 (to PROB7) - Controls the probability that the Object will spawn in the area level. This is calculated in order so the first probability that is successful will be chosen. This also means that these field values should add up to exactly 100 in total to guarantee that one of the objects spawn.

objpreset.txt

Overview

This file controls which Objects are preloaded in a preset, based on the Act number

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Index - Assigns a unique numeric ID to the Object Preset so that it can be properly referenced
Act - Defines the Act number used for each Object Preset. Uses values between 1 to 5.
ObjectClass - Uses the “Class” field from objects.txt, which assigns an Object to this Object Preset

Overlay.txt

Overview

This file controls the overlay graphics related to states, auras, cast animations, curses, and buffs

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

overlay - Defines the name of the overlay, used in other data files
Filename - Defines which DCC file to use for the Overlay
version - Defines which game version to use this Overlay (0 = Classic mode | 100 = Expansion mode)
Character - Used for name categorizing Overlays for unit translation mapping

PreDraw - Boolean field. If equals 1, then display the Overlay in front of sprites. If equals 0, then display the Overlay behind sprites.
1ofN - Controls how to randomly display Overlays. This value will randomly add to the current index of the Overlay to possibly use another Overlay that is indexed after this current Overlay. The formula is as follows: Index = Index + RANDOM(0, [“1ofN”]-1).
Xoffset - Sets the horizontal offset of the overlay on the unit. Positive values move it toward the left and negative values move it towards the right.
Yoffset - Sets the vertical offset of the overlay on the unit. Positive values move it down and negative values move it up.
Height1 (to Height4) - These are additional values added to “Yoffset”. Only 1 of these “Height” fields are added, and which field that gets selected depends on the “OverlayHeight” field value from monstats2.txt (Example: If the “OverlayHeight” value is 4, then use the “Height4” field). If the “OverlayHeight” value is 0, then ignore these “Height” fields and add a default value of 75 to “Yoffset”. Player unit types will always use “Height2”.

AnimRate - Controls the animation frame rate of the Overlay. The value is the number of frames that will update per second.
LoopWaitTime - Controls the number of periodic frames to wait until redrawing the Overlay. This only works with Overlays that are a loop type.
Trans - Controls the alpha mode for how the Overlay is displayed, which can affect transparency and blending

Code	Description
0	Transparency at 25%
1	Transparency at 50%
2	Transparency at 75%
3	Black Alpha Transparency
4	White Alpha Transparency
5	No Transparency
6	Dark Transparency (Unused)
7	Highlight Transparency (Used when mousing over the unit)
8	Blended

InitRadius - Controls the starting Light Radius value for the Overlay (Max = 18)
Radius - Controls the maximum Light Radius value for the Overlay. This can only be greater than or equal to “InitRadius”. If greater than “InitRadius”, then the Light Radius will increase in size per frame, starting from “InitRadius” until it matches the “Radius” value (Max = 18)
Red - Controls the Red color gradient of the Light Radius
Green - Controls the Green color gradient of the Light Radius
Blue - Controls the Blue color gradient of the Light Radius

NumDirections - The number of directions in the cell file
LocalBlood - Controls how to display green blood or VFX on a unit.

Code	Quest Progress
0	null
1	Transform the default red blood splatter to green blood
2	Change the monster palette to green

pettype.txt

Overview

This file controls the various statistics for each type of pet from all the classes summon Skills

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

pet type - Defines the name of the pet type, used in the "pettype" column in skills.txt
group - Used as an ID field, where if pet types share the same group value, then only 1 pet of that group is allowed to be alive at any time. If equals 0 (or null), then ignore this.
basemax - This sets a baseline maximum number of pets allowed to be alive when skill levels are reset or changed.

warp - Boolean field. If equals 1, then the Pet will teleport to the player when the player teleports or warps to another area. If equals 0, then the pet will die instead.
range - Boolean field. If equals 1, then the Pet will die if the player teleports or warps to another area and is located more than 40 grid tiles in distances from the Pet. If equals 0, then ignore this.
partysend - Boolean field. If equals 1, then tell the Pet to do the Party Location Update command (find the location of its Player) when its health changes. If equals 0, then ignore this.
unsummon - Boolean field. If equals 1, then the Pet can be unsummoned by the Unsummon skill function. If equals 0, then the Pet cannot be unsummoned.
automap - Boolean field. If equals 1, then display the Pet on the Automap. If equals 0, then hide the pet on the Automap.

name - String Key. Used to define the Pet's name on its party frame
drawhp - Boolean field. If equals 1, then display the Pet's Life bar under the party frame. If equals 0, then hide the Pet's Life bar under the party icon.
icontype - Controls the functionality for how to display the Pet Icon and number of Pets counter

Code	Description
0	Do not display the Pet icon
1	Display the Pet icon and do not show the Pet counter
2	Display the Pet icon and show the Pet counter

baseicon - Define which DC6 file to use for the default Pet's icon in its party frame

mclass1 (to mclass4) - Defines the alternative pet to use for the "pet type" by using that specific unit's "hcIdx" from Monstats.txt
micon1 (to micon4) - Defines which DC6 file to use for the related "mclass" Pet's icon in its party frame

Properties.txt

Overview

This file defines how item modifiers work. It takes a stat defined from ItemStatCost.txt and uses a function to handle the stat's "min", "max" and "parameter" values.

Used by the following data files: UniqueItems.txt, SetItems.txt, QualityItems.txt, Sets.txt, Runes.txt

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

code - Defines the property ID. Used as a reference in other data files (this should not be changed)
func1 (to func7) - Code function used to define the Property. Uses numeric ID values to define what function to use.

Function ID	Function Name	Parameters	Description
0			null
1	ItemModsSetValueRegular	stat set	• Modify the stat to be randomly calculated between its "min" and "max" values • Sets the stat value to its "max" value if the item is High Quality (Superior)
2	ItemModsSetValueBaseToMax	stat set	Modify the stat to always be set to its "max" value
3	ItemModsSetValueRegular2	stat set	Same as function 1, but consecutive calls of this function will use the same stat value as the previous call
4	ItemModsSetValueBaseToMax2	stat set	Same as function 2, but consecutive calls of this function will use the same stat value as the previous call
5	ItemModsSetMinDamage	set	Sets the minimum damage value for an item
6	ItemModsSetMaxDamage	set	Sets the maximum damage value for an item (dependent on its minimum value)
7	ItemModsSetDamagePct	set	Sets the damage percent of the item based on its percentage damage "min" and "max" values
8	ItemModsSetSpeed	stat set	Modify the stat to be randomly calculated between its "min" and "max" values
9	ItemModsSetSingleSkill	stat set	• Used for modifying a single skill level • Requires the stat's "min" and "max" values for the skill's level modification • Requires the stat's "parameter" value for the skill ID
10	ItemModsSetTabSkills	stat set	• Used for modifying the levels of skills from a skill tab

			• The skill tab level modification is defined through the stat's value • The skill tab ID is defined through the stat's "parameter" value. The stat's "parameter" value is defined as the class ID and the number of tabs that the class has: ○ 0-2 = Amazon (Bow and Crossbow Skills / Javelin and Spear Skills / Passive and Magic Skills) ○ 3-5 = Sorceress (Fire Spells / Lightning Spells / Cold Spells) ○ 6-8 = Necromancer (Summoning Spells / Poison and Bone Spels / Curses) ○ 9-11 = Paladin (Combat Skills / Defensive Auras / Offensive Auras) ○ 12-14 = Barbarian (Warcries / Combat Masteries / Combat Skills) ○ 15-17 = Druid (Shape Shifting / Elemental / Summoning) ○ 18-20 = Assassin (Traps / Martial Arts / Shadow Disciplines)
11	ItemModsSetSkillOnAttack	stat set	• Used for item event modifiers to cast a skill • Requires the stat's param value as the skill ID • Requires the stat's "min" value as the percent chance to cast the skill (if 0, then default to 5) • Requires the stat's "max" value as the skill's level
12	ItemModsSetRandomParam	stat set	• Uses the stat's "min" and "max" value as a random selection of the stat's "parameter" value
13	ItemModsSetMaxDurability	stat set	• Modify the stat to be randomly calculated between its "min" and "max" values • Sets the stat value to its "max" value if the item is High Quality (Superior) • Always sets the current durability to its maximum durability after the calculation of the stat value
14	ItemModsSetSockets		• Determines the number of sockets on an item • If the stat has "min" and "max" values, then calculate a random number of sockets between these values. Otherwise, use the stat's "parameter" value as the number of sockets • The max number of sockets depends on the stat's "max" size, the item's inventory grid size, or the hard cap of 6 sockets maximum
15	ItemModsSetMin	stat set	• Always use the stat's "min" value • If the stat is physical minimum damage, then set the item's minimum damage to the stat's value. Otherwise, simply set the stat's value to its "min" value.
16	ItemModsSetMax	stat set	• Always use the stat's "max" value • If the stat is physical maximum damage, then set the item's maximum damage to the stat's value. Otherwise, simply set the stat's value to its "max" value.
17	ItemModsSetParam	stat set	• Use the stat's "parameter" value. Otherwise, calculate a random value between the stat's "min" and "max" value. Otherwise, use 0. • If the stat is physical maximum damage, then set the item's maximum damage to the stat's value
18	ItemModsSetByTime	stat	• Modifies the stat based on the current game's time of day, and the stat's preferred time period. The closer the current game's time of day is to the stat's preferred time period, then the stronger the stat's value will be, based on its "min" and "max" values • Requires the stat's "parameter" value as the time period. The allowed time periods are: ○ 0 = Day ○ 1 = Dusk ○ 2 = Night ○ 3 = Dawn
19	ItemModsSetChargedSkill	stat	• Used for creating a stat for a charged skill. • Requires the stat's "parameter" value as the skill ID • Requires the stat's "min" value to calculate the value MaxCharges (maximum number of charges) ○ If that value equals 0, then default to 5 max charges ○ If that value is less than 0, then equal to the following calculation: MaxCharges = |MaxCharges| + |MaxCharges| * [CURRENT ITEM LEVEL] / 8 ○ MaxCharges cannot exceed 255 • Requires the stat's "max" value as the skill's level • The spawned number of charges is calculated as the following: ○ Random(0 and (MaxCharges - MaxCharges / 8)) + MaxCharges / 8 + 1

20	ItemModsSetIndestructible		Adds the Indestructible stat to an item
21	ItemModsSetValueRegPropValParam	stat set val	Modify the stat to be randomly calculated between its “min” and “max” values and use the Property “val” value to offset the stat ID
22	ItemModsSetValueRegParam	stat set	Modify the stat to be randomly calculated between its “min” and “max” values and use the stat’s “parameter” value to offset the stat ID
23	ItemModsSetEthereal		Used to add the Ethereal stat to an item, only if the item has Durability
24	ItemModsSetParamAndValue	stat set	• Modify the stat’s value to be randomly calculated between its “min” and “max” values and use the stat’s “parameter” value to offset the stat ID • Consecutive calls of this function will use the same stat value as the previous call
25 to 35	null		null
36	ItemModsSetValueRegPropValParamSwapped	stat set val	• Switches the usage of the Property “val” value with the stat’s value • The Property “val” value is used as the stat value • The stat’s value (based on its “min” and “max” values) is used at the Property “val” value

stat1 (to stat7) - Stat applied by the property. Used by the “func” field as a possible parameter (uses “Stat” value from ItemStatCost.txt). A stat is comprised of a “min” and “max” value which it uses to calculate the actual numeric value. Stats also can have a “parameter” value, depending on its function.
set1 (to set7) - Boolean field. Used by the “func” field as a possible parameter. If equals 1, then set the stat value regardless of its current value. If equals 0, then add to the stat value.
val1 (to val7) - Integer field. sed by the “func” field as a possible input parameter for additional function calculations

QualityItems.txt

Overview

This file controls the groups item modifiers for High Quality (Superior) item types.

The game will randomly choose between one of these High Quality groups, if it is allowed for the item type.

Data Fields

mod1code & mod2code - Controls the item properties that are added to the item (Uses the “code” field from Properties.txt)
mod1param & mod2param - The stat’s “parameter” value associated with the related property (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)
mod1min & mod2min - The stat’s “min” value to assign to the related property (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)
mod1max mod2max - The stat’s “max” value to assign to the related property (mod#code). Usage depends on the property function (See the “func” field on Properties.txt)

armor - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on both torso armor and helmet item types. If equals 0, then ignore this.
weapon - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on melee weapon item types (except scepters, wands, and staves). If equals 0, then ignore this.
shield - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on shield item types. If equals 0, then ignore this.
scepter - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on scepter item types. If equals 0, then ignore this.
wand - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on wand item types. If equals 0, then ignore this.
staff - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on staff item types. If equals 0, then ignore this.
bow - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on bow or crossbow item types. If equals 0, then ignore this.
boots - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on boots item types. If equals 0, then ignore this.
gloves - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on gloves item types. If equals 0, then ignore this.
belt - Boolean Field. If equals 1, then allow this High Quality (Superior) modifier to be applied on belt item types. If equals 0, then ignore this.

RarePrefix.txt

Overview

This file controls the list of strings that are randomly selected to be used as the prefix part of the name when generating Rare items

Rare Prefixes are chosen at random from the list define in the data file

These item affixes will appear at the start of a Rare item’s name

Data Fields

name - Uses a string key to define the Rare Prefix name

version - Defines which game version to use this Set bonus (0 = Classic mode | 100 = Expansion mode)

itype1 (to itype7) - Controls what item types are allowed for this Rare Prefix to spawn on (Uses the ID field from ItemTypes.txt)
etype1 (to etype4) - Controls what item types are excluded for this Rare Prefix to spawn on (Uses the ID field from ItemTypes.txt)

RareSuffix.txt

Overview

This file controls the list of strings that are randomly selected to be used as the suffix part of the name when generating Rare items

Rare Suffixes are chosen at random from the list define in the data file

These item affixes will appear at the end of a Rare item's name

Data Fields

name - Uses a string key to define the Rare Suffix name
version - Defines which game version to use this Set bonus (0 = Classic mode | 100 = Expansion mode)

itype1 (to itype7) - Controls what item types are allowed for this Rare Suffix to spawn on (Uses the ID field from ItemTypes.txt)
etype1 (to etype4) - Controls what item types are excluded for this Rare Suffix to spawn on (Uses the ID field from ItemTypes.txt)

Runes.txt

Overview

This file controls the creation of Rune Words and their various modifiers.

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

Name - Controls the string key that is used to the display the name of the item when the Rune Word is complete
complete - Boolean field. If equals 1, then the Rune Word can be crafted in-game. If equals 0, then the Rune Word cannot be crafted in-game.
firstLadderSeason - Integer field. The first ladder season the Rune Word can be made on (inclusive). If blank or 0 then it is available in non-ladder.
lastLadderSeason - Integer field. The last ladder season the Rune Word is ladder-only (inclusive). Must be used in conjunction with firstLadderSeason.

itype1 (to itype6) - Controls what item types are allowed for this Rune Word (Uses the ID field from ItemTypes.txt)
etype1 (to etype3) - Controls what item types are excluded for this Rune Word (Uses the ID field from ItemTypes.txt)
Rune1 (to Rune6) - Controls what runes are required to make the Rune Word. The order of each of these fields matters. (Uses the ID field from misc.txt)

T1Code1 (to T1Code7) - Controls the item properties that the Rune Word provides (Uses the "code" field from Properties.txt)
T1Param1 (to T1Param7) - The stat's "parameter" value associated with the related property (T1Code). Usage depends on the property function (See the "func" field on Properties.txt)
T1Min1 (to T1Min7) - The stat's "min" value to assign to the related property (T1Code). Usage depends on the property function (See the "func" field on Properties.txt)
T1Max1 (to T1Max7) - The stat's "max" value to assign to the related property (T1Code). Usage depends on the property function (See the "func" field on Properties.txt)

SetItems.txt

Overview

This file controls the item modifiers for each Set item in a Set

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

index - Links to a string key for displaying the Set item name
set - Defines the Set to link to this Set Item (must match the "index" field from Sets.txt)
item - Defines the baseline item code to use for this Set item (must match the "code" field value from weapons.txt, armor.txt, or misc.txt)
rarity - Modifies the chances that this Unique item will spawn compared to the other Set items. This value acts as a numerator and a denominator. Each "rarity" value gets summed together to give a total denominator, used for the random roll for the item. For example, if there are 3 possible Set items, and their "rarity" values are 3, 5, 7, then their chances to be chosen are 3/15, 5/15, and 7/15 respectively. (The minimum "rarity" value equals 1)

lvl - The item level for the item, which controls what object or monster needs to be in order to drop this item
lvl req - The minimum character level required to equip the item

chrtransform - Controls the color change of the item when equipped on a character or dropped on the ground. If empty, then the item will have the default item color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

invtransform - Controls the color change of the item in the inventory UI. If empty, then the item will have the default item color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

invfile - An override for the “invfile” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Set Item will use what was defined by the baseline item from the “item” field**.**
flippyfile - An override for the “flippyfile” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Set Item will use what was defined by the baseline item from the “item” field.

dropsound - An override for the “dropsound” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Set Item will use what was defined by the baseline item from the “item” field.
dropsfxframe - An override for the “dropsfxframe” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Set Item will use what was defined by the baseline item from the “item” field.
usesound - An override for the “usesound” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Set Item will use what was defined by the baseline item from the “item” field.

cost mult - Multiplicative modifier for the Set item’s buy, sell, and repair costs
cost add - Flat integer modification to the Set item’s buy, sell, and repair costs. This is added after the “cost mult” has modified the costs.

add func - Controls how the additional Set item properties (aprop#a & aprob#b) will function on the Set item based on other related set items are equipped

Code	Description
0 (or empty)	Additional Set item properties will function like normal item properties, ignoring the Set
1	Additional Set item properties will be added depending on which specific Set item is equipped. Each Set item has their own index depending on their order in data and the “set” they belong to. For example, if a Set item is defined first in the list, that that it has the index equal to 1, which means this function will make “aprop1a” and “aprop1b” fields only be added to a Set Item when that specific Set item of index 1 is equipped.

2	Additional Set item properties will be added depending the number of related Set items equipped. For example, if 2 Set items are equipped, then the “aprop1a”, “aprop1b”, “aprop2a”, and “aprop2b” fields will be added to the Set item.

prop1 (to prop9) - Controls the item properties that are add baseline to the Set Item (Uses the “code” field from Properties.txt)
par1 (to par9) - The stat’s “parameter” value associated with the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
min1 (to min9) - The stat’s “min” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
max1 (to max9) - The stat’s “max” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)

aprop1a (to aprop5a) - Controls the item properties that are added to the Set Item when other pieces of the Set are also equipped (Uses the “code” field from Properties.txt)
apar1a (to apar5a) - The stat’s “parameter” value associated with the related property (aprop#a). Usage depends on the property function (See the “func” field on Properties.txt)
amin1a (to amin5a) - The stat’s “min” value to assign to the related property (aprop#a). Usage depends on the property function (See the “func” field on Properties.txt)
amax1a (to amax5a) - The stat’s “max” value to assign to the related property (aprop#a). Usage depends on the property function (See the “func” field on Properties.txt)

aprop1b (to aprop5b) - Controls the item properties that are added to the Set Item when other pieces of the Set are also equipped. Each of these numbered fields are paired with the related “aprop#a” field as an additional item property. (Uses the “code” field from Properties.txt)
apar1b (to apar5b) - The stat’s “parameter” value associated with the related property (aprop#b). Usage depends on the property function (See the “func” field on Properties.txt)
amin1b (to amin5b) - The stat’s “min” value to assign to the related property (aprop#b). Usage depends on the property function (See the “func” field on Properties.txt)
amax1b (to amax5b) - The stat’s “max” value to assign to the related property (aprop#b). Usage depends on the property function (See the “func” field on Properties.txt)

diablocloneweight - The amount of weight added to the diablo clone progress when this item is sold. When offline, selling this item will instead immediately spawn diablo clone.

Sets.txt

Overview

This file controls the item modifiers for Set bonus statistics when the player has equipped enough Set Items

Data Fields

index - Defines the specific Set ID
name - Uses a string for displaying the Set name in the inventory tooltip
version - Defines which game version to use this Set bonus (0 = Classic mode | 100 = Expansion mode)

PCode2a (to PCode5a) - Controls the each of the different pairs of Partial Set item properties. These are applied when the player has equipped the related # of Set items. This is the first part of the pair for each Partial Set bonus. (Uses the “code” field from Properties.txt)
PParam2a (to PParam5a) - The stat’s “parameter” value associated with the relative property (PCode#a). Usage depends on the property function (See the “func” field on Properties.txt)
PMin2a (to PMin5a) - The stat’s “min” value associated with the listed relative (PCode#a). Usage depends on the property function (See the “func” field on Properties.txt)
PMax2a (to PMax5a) - The stat’s “max” value to assign to the listed relative (PCode#a). Usage depends on the property function (See the “func” field on Properties.txt)

PCode2b (to PCode5b) - Controls the each of the different pairs of Partial Set item properties. These are applied when the player has equipped the related # of Set items. This is the second part of the pair for each Partial Set bonus. (Uses the “code” field from Properties.txt)
PParam2b (to PParam5b) - The stat’s “parameter” value associated with the relative property (PCode#b). Usage depends on the property function (See the “func” field on Properties.txt)
PMin2b (to PMin5b) - The stat’s “min” value associated with the listed relative (PCode#b). Usage depends on the property function (See the “func” field on Properties.txt)
PMax2b (to PMax5b) - The stat’s “max” value to assign to the listed relative (PCode#b). Usage depends on the property function (See the “func” field on Properties.txt)

FCode1 (to FCode8) - Controls the each of the different Full Set item properties. These are applied when the player has all Set item pieces equipped (Uses the “code” field from Properties.txt)
FParam1 (to FParam8) - The stat’s “parameter” value associated with the relative property (FCode#b). Usage depends on the property function (See the “func” field on Properties.txt)
FMin1 (to FMin8) - The stat’s “min” value associated with the listed relative (FCode#b). Usage depends on the property function (See the “func” field on Properties.txt)
FMax1 (to FMax8) - The stat’s “max” value to assign to the listed relative (FCode#b). Usage depends on the property function (See the “func” field on Properties.txt)

shrines.txt

Overview

This file controls the functionalities of shrine objects found in area levels

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Name - This is a reference field to define the Shrine index

Code - Code function used to define the Shrine’s function. Uses ID values to define what function to use.

Code ID	Parameter	Description
0		None
1		Gain full Life and Mana
2		Gain full Life
3		Gain full Mana
4	Arg0 = Life percent consumed Arg1 = Mana percent added	Exchange your current Life to restore Mana
5	Arg0 = Mana percent consumed Arg1 = Life percent added	Exchange your current Mana to restore Life
6	Arg0 = Defense percent	Increases Defense
7	Arg0 = Attack Rating percent Arg1 = Physical Damage percent	Increases Physical Damage and Attack Rating
8	Arg0 = Resist Fire percent	Increases Fire Resistance
9	Arg0 = Resist Cold percent	Increases Cold Resistance
10	Arg0 = Resist Lightning percent	Increases Lightning Resistance
11	Arg0 = Resist Poison percent	Increases Poison Resistance
12	Arg0 = Bonus Skill Levels	Increases all Skill levels
13	Arg0 = Mana Recharge percent	Increases Mana Recharge Rate
14	Arg0 = Stamina percent	Gain infinite Stamina
15	Arg0 = Bonus Experience percent	Temporarily gain bonus Experience from kills
16		Temporarily reverse your character's Name (Not Used)
17		Create a neutral Town Portal back to the current Act Town
18		Randomly select a gem in your inventory and upgrade its level (Otherwise, create a random chipped gem)
19	Arg0 = Life percent damage Arg1 = Range to find units	Release a nova of fireballs that cause any player or monster to lose a percentage of Life
20		Causes the nearest monster to upgrade a Unique or Champion type
21	Arg0 = Minimum potions Arg1 = Maximum potions	Deal Fire damage to nearby monsters and create a random number of Exploding Potions
22	Arg0 = Minimum potions Arg1 = Maximum potions	Create Poison Gas that damages nearby monsters and create a random number of Choking Gas Potions

Arg0 & Arg1 - Integer value used as a possible parameter for the “Code” function

Duration in frames - Duration of the effects of the Shrine (Calculated in Frames, where 25 Frames = 1 Second)
reset time in minutes - Controls the amount of time before the Shrine is available to use again. Each value of 1 equals 1200 Frames or 48 seconds. A value of 0 means that the Shrine is a one-time use.

StringName - Uses a string to display as the Shrine’s name
StringPhrase - Uses a string to display as the Shrine’s activation phrase when it is used
effectclass - Used to define the Shrine’s archetype which is involved with calculating region stats
LevelMin - Define the earliest area level where the Shrine can spawn. Area levels are determined from levels.txt

skills.txt

Overview

This file controls all skill functionalities. Skills are abilities used by all units in the game.

This file uses many other data files, and other data files will reference fields in this file to verify certain functionalities.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

skill - Defines the unique name ID for the skill, which is how other files can reference the skill. The order of the defined skills will determine their ID numbers, so they should not be reordered.

charclass - Assigns the skill to a specific character class which affects how skill item modifiers work and what skills the class can learn.

Code	Description
ama	Amazon
bar	Barbarian
pal	Paladin
nec	Necromancer
sor	Sorceress
dru	Druid
ass	Assassin

skilldesc - Controls the skill's tooltip and general UI display. Points to a "skilldesc" field in the skilldesc.txt file.

srvstfunc - Server Start function. This controls how the skill works when it is starting to cast, on the server side. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1		StartAttack - Check that the attack is melee or ranged. If the attack is ranged, then verify the ammunition.
2		StartKick - Calculate the damage and attack the target unit with a Hand-To-Hand hit class.
3		StartUnsummon - Validate that the target unit is not a monster or player and that the unit is owned by the caster unit. Check that the pet can be unsummoned (See "unsummon" in pettype.txt)
4		AmaStartCheckQuantity - Verify that the caster unit has enough weapon ammunition
5		AmaStartJab - Return true.
6	calc1 calc4	AmaStartPowerStrike - Validate the target enemy and attempt to attack it. Use "calc1" to control the percentage increase for physical damage dealt. Use "calc4" to control the percent of damage converted to elemental, if the "Etype" field is used.
7	calc1 calc2 calc3 calc4	AmaStartImpale - Validate the target enemy and attempt to attack it. Use "calc1" to control the percentage increase for physical damage dealt. Use "calc4" to control the percent of damage converted to elemental, if the "Etype" field is used. Use "calc2" to control the percent chance of losing weapon durability. Use "calc3" to control the flat amount of durability lost.
8	aurarangecalc calc1 calc3	AmaStartStrafe - Attempt to find nearby valid targets and shoot them. Use "aurarangecalc" to control the range to find targets. Use "calc1" and "calc3" to control the minimum and maximum amount of shots fired.
9	calc1	AmaStartFend - Find a valid target to attack in melee and then perform multiple attacks to nearby enemies. Use "calc1" to control the max targets to attack.
10	calc1 calc4	AmaStartLightningStrike - Validate the target enemy and attempt to attack it to deal a random amount of lightning damage. Use "calc1" to control the percentage increase for damage dealt. Use "calc4" to control the percent of damage converted to elemental, if the "Etype" field is used.
11	srvmissilea calc2	SorStartInferno - Continuously create missiles while the caster has the "inferno" state, and adjust the animations and modes based on the inferno frames. Use "calc2" to control the monster channel duration.
12	aurarangecalc	SorStartTelekinesis - Validate the range distance and the target type
13		SorStartThunderStorm - Validate the skill use and setup targeting parameters
14		SorStartHydra - Validate the target location
15		NecStartRaiseSkeleton - Check for a valid target corpse that can be raised
16	calc1 calc4	NecStartPoisonDagger - Validate the target enemy and attempt to attack it. Use "calc1" to control the percentage increase for physical damage dealt. Use "calc4" to control the percent of damage converted to elemental, if the "Etype" field is used.
17		NecStartCorpseExplosion - Check for a valid target corpse that can explode
18		NecStartAttract - Return true
19		NecStartBonePrison - Validate that the target is an enemy and make sure that the target is not located in town
20		NecStartIronGolem - Validate that the target is an identified item located on the ground
21		NecStartRevive - Check for a valid target corpse that can be revived
22		AssStartPsychicHammer - Check for a valid target player or monster
23		AssStartProgressiveAttack – Reset internal variable used for keeping track of skill steps.
24	calc1	AssStartDragonTalon - Use "calc1" to control the number of attacks.
25		AssStartDragonClaw - Reset internal variable used for keeping track of skill steps.
26	aurastate aurastat1 srvmissilea srvmissileb srvmissilec	AssStartBladeFury - Use the "srvmissilea" missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the "aurastate" field and "aurastat1" fields. If the caster unit does not have the "inferno" state, then add it and handle the animation frames. If the caster unit does have the "inferno" state, then periodically create missiles.
27	Param4 Param8	AssStartDragonTail - Validate the target to attack and calculate the kick damage. Use "Param4" to control the attack speed. Use "Param8" to control if the attack should always hit.
28	aurastate aurastat1 srvmissilea srvmissileb srvmissilec auralencalc	AssStartBladeShield - Use the "srvmissilea" missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the "aurastate" field and "aurastat1" fields. Add the "aurastate" state that lasts a duration controlled by "auralencalc".
29	calc1 calc4	PalStartSacrifice - Validate the target enemy unit and determine if the caster unit can melee attack it. Use "calc1" to control the percentage increase for physical damage dealt. Use "calc4" to control the percent of damage converted to elemental, if the "Etype" field is used.
30		Do nothing
31	calc2	PalStartCharge - Validate the target enemy unit and determine if the caster unit can melee attack it. Adjust the caster unit's movement speed. Do special adjustments for the "duriel" and "clawviper1" monster attack modes. Use "calc2" to add bonus movement speed percentage while charging.
32	aurastate calc1	BarStartBash - Validate the target enemy and attack it. Use "calc1" to control the physical damage percent increase. Use "calc2" to control the flat damage increase. Use "calc3" to

	calc2 calc3 calc4	control the attack speed bonus. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used. Applies “aurastate” to self.
33		BarStartFindHeart - Check for a valid target corpse that can spawn potions. This relies on the “soft” and “noSel” flags from the monstats2.txt file.
34		BarStartFindItem - Check for a valid target corpse that can spawn items. This relies on the “soft” and “noSel” flags from the monstats2.txt file.
35	calc1 calc2 calc3	PalStartVengeance - Validate the target enemy and attack it. Use the calculation fields to control fire, cold, and lightning damage percentages added to the attack.
36		PalStartHolyShield - Check that the player has a shield equipped
37	calc1	AmaStartFend2 - Find nearby enemy targets to melee attack. Use “calc1” to control the maximum number of targets to attack.
38	aurastate auralencalc auraevent1 auraeventfunc1 auraevent2 auraeventfunc2 auraevent3 auraeventfunc3	BarStartWhirlwind - Stop any skills and validate the target location. Modify the caster unit’s collision to only collide with walls and objects and save the target location. Apply the “aurastate” state with the length controlled. Applies “auraevents” if they exist.
39	aurastate calc1 calc2 calc4	BarStartBerserk - Validate the target enemy unit and determine if the caster unit can melee attack it. Use “calc1” to control the percentage increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used. Use “calc2” to control the duration for how long the caster unit has the state.
40	aurarangecalc	BarStartLeap - Adjust the caster unit collision, validate the target location, and store the location in a parameter. If the caster unit is in a monster, then handle how the monster can attack the target while leaping.
41		BarStartLeapAttack - Adjust the caster unit collision, validate the target location, and store the location in a parameter. Make the caster unit uninterruptable after landing in order to melee attack a nearby target.
42		MonStartFirehit - If the caster unit is a player, then use the BarStartBash function (Code 32). Otherwise make the caster attack the target while in “Skill 1” mode and deal damage
43		MonStartMagottEgg - Make the unit unattackable, unselectable, and unable to be hit by missiles
44		MonStartMaggotUp - Set the unit to have ground collision and adjust the collision and pathing. Teleport the unit to a viable location in the area.
45		MonStartMaggotDown - Make the unit unattackable, unselectable, and unable to be hit by missiles. Adjust the unit’s collision to not have pathing.
46		MonStartAndariel - Validate the target unit and store the target’s location in a parameter
47	calc1	MonStartJump - Validate the target location. Return false if the caster unit has the “freeze” state. Use the “calc1” field to control the damage percent bonus. Make the caster unit attack the target, if possible.
48		MonStartSwarmMove - Find and validate a path to the target.
49		MonStartNest - Validate the caster unit’s location and modify its collision
50		MonStartQuickStrike - Validate the target unit and attack it
51		MonStartSubmerge - Make the unit unattackable, unselectable, and unable to be hit by missiles
52		MonStartEmerge - Make the unit unattackable, unselectable, and unable to be hit by missiles
53	calc2	MonStartDiabLight - Add the “inferno” state to the caster unit. Use “calc2” to control the number of frames to add to the animation while channeling and save it in a parameter.
54		MonStartDiabRun - Validate the target unit and save its location in parameters
55	calc1 calc2	MonStartMosquito - Validate the target unit and use the calculation fields to control the minimum and maximum number of animation loops for the skill.
56	calc1 calc4	DruStartChargeUpAttack - Validate the target enemy and attempt to attack it. Use “calc1” to control the percentage increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.
57	aurastate	DruStartRabies - Validate the “aurastate” state. Reset internal variables used to keep track of hits.
58	calc1	DruStartFireClaws - Validate the target enemy and attempt to attack it. Use “calc1” to control the percentage increase for physical damage dealt.
59	calc1	MonStartImpInferno - Add the “inferno” state to the caster unit. Use “calc1” to control the number of frames to add to the animation while channeling and save it in a parameter.
60	calc1 calc2 calc3	MonStartBatSuckBlood - Validate the target enemy and attempt to attack it. Use “calc1” to control the damage penalty percentage. Use “calc2” to control the life steal percent. Use “calc3” to control the mana steal percent.
61		MonStartSelfResurrect - Validate that the caster unit is a monster, then resurrect the monster, making it have proper pathing, be attackable, selectable, and able to be hit by missiles.
62		MonStartSpawner - Save the monster’s position, class, and mode as parameters
63	srvmissilea	MonStartCorpseCycler - Validate that the caster unit is a monster. Check for the “noSel” state on the target (see monstats2.txt) or add it on the target. Create the missile at the target location and corpse explode the corpse on the client.
64		MonStartFrenzy - Validate that the target is an enemy. Reset internal variables used to keep track of frames and attacks.
65		StartThrow - Validate that the caster unit has enough ammunition and durability
66	auraevent1 auraeventfunc1 auraevent2 auraeventfunc2 auraevent3	ApplyPassiveAuraEvents - Check each of the aura events on the skill and apply the event handler to use the aura event functions

	auraeventfunc3	
67		BarStartFrenzy - Resets internal variables used to keep track of frames and attacks.
68	calc2	BarStartDoubleSwing - Uses “calc2” to apply an attack rate bonus.

srvdofunc - Server Do function. This controls how the skill works when it finishes being cast, on the server side. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1		DoAttack - If using a ranged weapon, then launch the weapon’s missile. Otherwise, perform a standard melee attack to deal damage.
2	srvoverlay aurastate auratargetstate auralencalc	DoApplyDamage - Apply the overlay on the target unit when dealing damage. Apply the “auratargetstate” state on the target, if possible, with “auralencalc” controlling its duration. Apply the “aurastate” state on the caster unit, if possible.
3		DoThrow - Check that primary equipped weapon is a throwing weapon and handle launching the weapon’s missile
4		DoUnsummon - Remove the pet from the caster owner
5		DoLeftThrow - Check that non-primary equipped weapon is a throwing weapon and handle launching the weapon’s missile
6	auratargetstate auralencalc aurarangecalc	AmaDoInnerSight - Apply the “auratargetstate” state to all units in the area with a radius controlled by “aurarangecalc”, that lasts a duration controlled by “auralencalc”. The state can use any of the aura stats and their related calculation values.
7	calc1 calc4	AmaDoJab - Attempt to attack the target unit. Use “calc1” to control the percent increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.
8	srvmissilea srvmissileb calc1 calc2 calc3 calc5	AmaDoMultipleShot - Shoot a number of missiles toward a target location. If the weapon class is a bow then use “srvmissilea”, otherwise use “srvmissileb” as the missile to create. Use “calc1” to control the number of missiles created. Use “calc2” to control the activation frame for each missile created. Use “calc3” to control the number of triggering missiles. Use the "calc5" value to calculate Guided Arrow synergy.
9	aurastate auralencalc calc1 calc4	BarDoFrenzy - Attack with both weapons on a target or to nearby targets. Add the “aurastate” state to the caster unit, with a duration controlled by “auralencalc”. Use “calc1” to control the percentage increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.
10	srvmissilea srvmissileb calc1	AmaDoGuidedArrow - Shoot a missile that will change its path to find a nearby target to hit. If the weapon class is a bow then use “srvmissilea”, otherwise use “srvmissileb” as the missile to create. Use “calc1” to control the physical damage dealt by the missile.
11	aurastate aurastat1 srvmissilea srvmissileb srvmissilec calc1	AmaDoChargedStrike - Create the number of missiles with randomized pathing. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “calc1” to control the number of missiles created.
12	aurarangecalc srvmissilea srvmissileb calc2 Param6	AmaDoStrafe - Use “aurarangecalc” to control the range to find a target. If the weapon class is a bow then use “srvmissilea”, otherwise use “srvmissileb” as the missile to create. Use “calc2” to control the percent increase for physical damage dealt. Use “Param6” to control what percent within the entire animation to trigger rolling back the loop within the animation.
13	calc1 calc4 Param2	AmaDoFend - Attempt to attack the target unit or nearby enemies with multiple attacks. Use “calc1” to control the percent increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used. Use “Param2” to control what percent within the entire animation to trigger rolling back the loop within the animation.
14	aurastate aurastat1 srvmissilea srvmissileb srvmissilec calc1 calc2	AmaDoLightningStrike - Damage the target and create a missile that bounces to different targets. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “calc1” to control the range of the missile to find a nearby target. Use “calc2” top control the number target chain jump hits for the missile.
15	pettype calc2 calc3	AmaDoDopplezon - Create a pet unit that is a duplicate of the caster unit. Use “calc1” to control the life percent of the pet. Use “calc3” to control the Life percent increase on the pet unit based on the caster’s Life. Use “calc2” to control the duration that the pet exists. Apply the “dopllezon_appear” overlay on the pet.
16	pettype calc1 calc2	AmaDoValkyrie - Create a pet unit with generated item equipment and stats. Use “calc1” to control the life percent of the pet. Use “calc2” to control the item level for the generated items on the pet.
17	aurastate aurastat1 srvmissilea srvmissileb srvmissilec calc1	SorDoChargedBolt - Create a number of missiles that have a randomized path. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “calc1” to control the number of missiles created.
18	aurastate auralencalc	SorDoFrozenArmor - Apply the state on the target unit with its length controlled by “auralencalc”.
19	srvmissilea calc1	SorDoInferno - Create the missile where “calc1” controls the range. Continue creating missiles while having the “inferno” state.
20	aurarangecalc calc1 calc2	SorDoStaticField - Apply damage to all units in the area. Use “aurarangecalc” to control the damage radius. Use “calc1” to control the Life percent damage. Use “calc2” to control the minimum damage dealt.

21	calc1	SorDoTelekinesis - If the target is a monster or player, then deal damage and use “calc1” to control the knockback chance. If the target is an item, then ensure that the item type is a scroll, gold, or potion. If the target is a object, then call the object’s operate function.
22	aurastate aurastat1 srvmissilea srvmissileb srvmissilec calc1	SorDoFrostNova - Shoot missiles in a circular array. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “calc1” to add to the velocity of the missiles created.
23	aurastate auralencalc	SorDoBlaze - Add the “aurastate” state on the caster with a duration controlled by “auralencalc”. Apply any aura stats or events.
24	srvmissilea srvmissileb calc1	SorDoFirewall - Create 2 “srvmissilea” missiles in opposite directions and create 2 “srvmissileb” missiles also at those locations. Use “calc1” to control how many different groups of these missiles can exist at once.
25	aurastate auralencalc	SorDoEnchant - Add the “aurastate” state on the target with a duration controlled by “auralencalc”. Apply any aura stats or events.
26	srvmissilea calc1	SorDoChainLightning - Create the missile, which can jump off targets hit, where “calc1” controls the number of missile chain jump hits
27		SorDoTeleport - Check that the level allows teleporting (see “Teleport” in Levels.txt), then validate the target location and warp the unit to that location.
28	srvmissilea	SorDoMeteor - Check that the target location is valid to spawn the missile, then create it
29	aurastate auralencalc srvmissilea Param7	SorDoThunderStorm - While the caster unit has the “aurastate” state with the “auralencalc” duration, find nearby a nearby enemy and shoot the missile. Use “Param7” to control the radius size for finding nearby enemies.
30	auratargetstate aurarangecalc auralencalc	NecDoAmplifyDamage - Apply the “auratargetstate” state on enemies in an area where “aurarangecalc” controls the radius and “auralencalc” controls the duration. Also apply and aura stats, events, and filters.
31	pettype calc1 calc2	NecDoRaiseSkeleton - Validate the target corpse and then create a pet unit. Use “calc1” to control the life percent of the pet. Use “calc2” to control the percent chance to spawn the skeleton with a shield (only works for the “necroskeleton” monster).
32		NecDoApplyDamage - Validate the target enemy and perform damage from the attacker
33	calc1 calc2 calc3 calc4	AssDoPsychicHammer - Validate that the target unit is a monster or player and is not in town. Use the calculation fields to control the chance to knockback the target if it is a monster, unique monster, boss, or player, respectively.
34	aurastate auralencalc aurastat1 aurastat2 aurastatcalc2	AssDoProgressiveAttack - Attempt to attack the target unit and deal damage. Calculate the progressive damage. Use “auralencalc” to determine the length of the charges. Use “aurastat1” to control the progressive charges. Use “aurastat2” as a stat when the player attacks and has no charges. Use “aurastatcalc2” to control that stat’s value.
35	aurastate auralencalc aurastat1 aurastat2 aurastatcalc2	AssDoDualProgressiveAttack - Check that the player has 2 weapons equipped. Attempt to attack the target unit twice with a frame delay, and use the “AssDoProgressiveAttack” (Code = 34) function for each attack.
36	aurastate aurastat1 srvmissilea srvmissileb srvmissilec calc1	ApplyClawsOfThunderLvl2 - Shoot missiles in a circular array. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “calc1” to add to the velocity of the missiles created.
37	aurastate aurarangecalc aurastat1 srvmissilea srvmissileb srvmissilec	ApplyClawsOfThunderLvl3 - Shoot missiles in an arc array. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “aurarangecalc” to control how many missiles are created, only if there are no values from the progressive charge calculation fields (see “prgcalc1”)
38	aurastate aurarangecalc aurastat1	AssDoAreaDamage - Deal damage to enemies in an area at a target location. Use the progressive calculation fields to determine the radius increase per charge, controlled by the “aurastate” state and the “aurastat1” value, otherwise use “aurarangecalc” for the radius.
39	aurastate aurarangecalc aurastat1 srvmissilea srvmissileb srvmissilec	AssMissileDisc - Create a disc of randomly positioned missiles. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use the progressive calculation fields to determine the radius increase per charge, controlled by the “aurastate” state and the “aurastat1” value, otherwise use “aurarangecalc” for the radius.
40	aurastate aurastat1 srvmissilea srvmissileb srvmissilec	ApplyRoyalStrikeLvl1 - Create a missile at a target location. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields.
41	aurastate aurarangecalc aurastat1 srvmissilea srvmissileb srvmissilec	ApplyChaosIce - Create multiple missiles from the caster unit. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use the progressive calculation fields to determine the number of missiles created per charge, controlled by the “aurastate” state and the “aurastat1” value. Use “aurarangecalc” for the radius.
42	calc2 calc3 calc4	AssDoDragonTalon - Attempt to melee attack the target unit multiple times. Use the progressive fields to control the charge functions. Use each calculation field to control the

	Param1 Param2	percent chance to knockback for a monster, boss, or player unit. Use the parameters to control a linear calculation for the percentage of bonus physical damage dealt.
43	aurastate aurarangecalc aurastat1 srvmissilea srvmissileb srvmissilec	AssDoShockField - Create multiple missiles using the lob function. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use the progressive calculation fields to determine the number of missiles created per charge, controlled by the “aurastate” state and the “aurastat1” value. Use “aurarangecalc” for the radius.
44	pettype	AssDoBladeSentinel - Summon a pet at the target location and cause it to oscillate
45	pettype	AssDoWakeOfFireSentry - Summon a pet at the target location
46	calc1 calc4	AssDoDragonClaw - Attempt to attack the target unit twice with a frame delay. Use “calc1” to control the percent increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used. Use the progressive fields to determine progressive damage changes.
47	aurastate auratargetstate auralencalc aurarangecalc	AssDoCloakOfShadows - Apply the “aurastate” state on the caster unit with a duration controlled by “auralencalc”. Apply the “auratargetstate” state on nearby enemies in a radius controlled by “aurarangecalc”. Use any applicable aura stats or filters.
48	aurastate aurastat1 srvmissilea srvmissileb srvmissilec	AssDoBladeFury - Create a missile after a frame delay. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use the progressive calculation fields to modify the delay between creating the next missile, based on the charges.
49	pettype Param5 Param6	AssDoShadowWarrior - Create a pet unit with generated item equipment and stats. Use “Param5” and “Param6” to control the item level for the generated items on the pet.
50	aurarangecalc calc1	AssDoDragonTail - Attempt to attack a target unit and deal damage in an area. Use the progressive fields to control the charge functions. Use “aurarangecalc” to control the radius. Use “calc1” to control the percent increase in damage dealt.
51	aurarangecalc calc1 calc2 Param4	AssDoMindBlast - Randomly damage and convert enemies in an area. Use “aurarangecalc” to control the radius, or if using the progressive charges, then use their field values instead. Use “calc1” to control the chance to convert enemies. Use “calc2” to control the duration enemies are converted. Use “Param4” to add an additional randomized duration value.
52	calc1	AssDoDragonFlight - Teleport the caster to the target unit and attempt to attack it. Use the progressive fields to control the charge functions. Use “calc1” to control the percent increase for physical damage dealt.
53	aurastate aurarangecalc aurastat1	AssDoAreaDamage2 - Deal damage to enemies in an area around the caster. Use the progressive calculation fields to determine the radius increase per charge, controlled by the “aurastate” state and the “aurastat1” value, otherwise use “aurarangecalc” for the radius.
54	aurastate aurarangecalc aurastat1	AssDoBladeShield - Deal damage to enemies in an area around the caster. Use the progressive calculation fields to determine the radius increase per charge, controlled by the “aurastate” state and the “aurastat1” value, otherwise use “aurarangecalc” for the radius.
55	aurarangecalc calc1 calc2 calc3 EType	NecDoCorpseExplosion - Hide the target corpse and deal damage in an area. Use “aurarangecalc” to control the radius. Use “calc1” and “calc2” to control the min and max percent damage dealt based on the max Life from the corpse unit. Use “calc3” to control the percent of damage converted to elemental.
56	pettype	NecCreateGolem - Create a pet unit with defined stats. If the skill has the “TargetableOnly” and “TargetCorpse” flag enabled, then the summoned pet will copy the modifiers of the corpse.
57	pettype	NecCreateIronGolem - Validate that the target is an identified item on the ground. Remove the item and create the pet, inheriting properties from the item.
58	pettype	NecDoRevive - Validate that the target is a corpse that can be revived, then revive the monster, applying any valid stats
59	auratargetstate auralencalc aurarangecalc aurastat1	NecDoAttract - Validate that the target is a monster and can have its AI changed. Apply the “auratargetstate” state on any valid monsters in an area controlled by “aurarangecalc” which lasts a duration controlled by “auralencalc”.
60	pettype calc2 srvmissilea	NecDoBoneWall - Create the pet, and then create 2 missiles that shoot in opposite directions, where “calc2” controls the number of sub missiles to create within each of these missile parameters.
61	auratargetstate auralencalc aurarangecalc aurastat1	NecDoConfuse - Validate that the target is a monster and can have its AI changed. Apply the “auratargetstate” state on any valid monsters in an area controlled by “aurarangecalc” which lasts a duration controlled by “auralencalc”.
62	pettype	NecDoBonePrison - Create a number of pets around the target unit.
63	srvmissilea	NecDoPoisonExplosion - Validate that the target corpse can explode, then update the corpse to be unselectable and create a radial ring of poison missiles.
64	calc2 Param4	PalDoSacrifice - Attack and deal damage to the target. Deal damage to the caster based on a percentage of life controlled by “calc2” and “Param4”.
65	aurastate auratargetstate aurarangecalc	PalDoMight - Apply the aurastate” state to the caster. Apply the “auratargetstate” state to nearby units, where “aurarangecalc” controls the radius size.
66	aurastate auratargetstate aurarangecalc	PalDoHolyFire - Apply the aurastate” state to the caster. Apply the “auratargetstate” state to nearby units, where “aurarangecalc” controls the radius size. Deal damage to the nearby units in the area.
67	calc1 calc4	PalDoCharge - Listen for the event frames and attempt to attack the target unit to deal damage. Use “calc1” to control the percent increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.
68	aurastate auralencalc	BarDoBattleCry - Create a circular array of missiles and add the “aurastate” state to the caster unit where the duration is controlled by “auralencalc”

	srvmissilea	
69	calc1 Param3 Param4	BarDoFindHeart - Validate the target corpse, then update the corpse to be unselectable and roll a random chance to create a potion. Use “calc1” to control the chance of finding a potion. Use “Param3” and “Param4” to control the chance for finding a mana potion and rejuvenation potion. Potions depend on the current Act level and Game Difficulty.
70	aurastate calc1 calc2 calc3 calc4	BarDoDoubleSwing - Validate the target enemy and attack it based on the animation sequence frame to determine if it is the first attack or second attack. Uses the BarStartBash function (Code 32 for “srvstfunc”).
71	auratargetstate auralencalc	BarDoTaunt - Validate that the target is a monster and can have its AI changed to force it to attack the caster. Apply the “auratargetstate” state on the target, which lasts a duration controlled by “auralencalc”. If there is no target selected, then find the nearest target within a radius value of 20.
72	calc1 Param1 Param2 Param3 Param4	BarDoFindItem - Validate the target corpse, then update the corpse to be unselectable and roll a random chance to spawn a treasure class item. Use “calc1” to control the chance of finding an item. Use the parameter values to control the chances for finding Low Quality, Normal, High Quality, and Magic items.
73	srvmissilea calc1	PalDoBlessedHammer - Create a missile in a spiral path pattern. Use “calc1” to control the damage percent bonus when the caster has the “concentration” state.
74	calc1	BarDoDoubleThrow - Launch the weapon missile, where “calc1” controls the bonus damage percent
75	srvmissilea srvmissileb srvmissilec	BarDoGrimWard - Validate the target corpse, then update the corpse to be unselectable and create the “srvmissilea” missile. If the target unit has the “large” flag enabled (see monstats2.txt) then use “srvmissileb” instead. If the target unit has the “small” flag enabled (see monstats2.txt) then use “srvmissilec” instead.
76	aurastate calc1	BarDoWhirlwind - If the caster unit is at the target location, then remove the aura and stop whirlwinding. Otherwise, find nearby enemies and deal damage, where “calc1” controls the damage bonus.
77	calc1 calc2	BarDoLeap - Validate the target position and caster’s collision, and move the unit. Use “calc1” to control the range, and “calc2” to control the speed.
78	calc1 calc2 calc4	BarDoLeapAttack - Validate the target position and the caster’s collision, and move the unit. Make the unit attack the target, if nearby, or find another nearby target. Use “calc2” to control the speed. Use “calc1” to control the percent increase for physical damage dealt. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.
79	auratargetstate auralencalc calc1 Param5	PalDoConversion - Validate that the target is a monster and can have its AI changed to fight alongside the player. Apply the “auratargetstate” state on the target, which lasts a duration controlled by “auralencalc”. Use “calc1” to control the chance to convert. Use “Param5” to enable a expire effect.
80	srvmissilea srvoverlay	PalDoFistOfTheHeavens - Create the missile and apply the overlay on the target unit
81	aurastate auratargetstate aurarangecalc	PalDoHolyFreeze - Apply the “aurastate” state on the caster, and apply the “auratargetstate” on any nearby enemies with a radius controlled by “aurarangecalc”.
82	aurastate aurarangecalc calc1 calc2 calc3	PalDoRedemption - Apply the “aurastate” state on the caster. Use “aurarangecalc” to control the radius of the aura, which will look for valid corpses to redeem. Use “calc1” to control the chance to redeem. Use “calc2” to control the life gain. Use “calc3” to control the mana gain.
83		MonDoFirehit - Apply damage to the target
84	calc1	MonDoMaggotEgg - Spawn a number of units around the caster and kill the caster. Use “calc1” to control the number of spawned units.
85	srvmissilea	MonDoShamanFire - Get the missile and possibly add the monster’s number in class to the missile ID to get another missile ID instead, and then fire that missile
86	calc1	MonDoMaggotDown - Check for the proper frame count and then heal the caster by a percentage of Life controlled by “calc1”
87		MonDoMaggotLay - Spawn a unit in 1 of 8 possible directions nearby the target location
88	srvmissilea	MonDoAndariel - For multiple frames, spawn a missile in 1 of 8 possible directions
89		MonDoJump - Validate the target path and check for the completion of the animation or the arrival to destination, then update the unit’s collision. Handle special cases for the “sandleaper1” monster frame counts.
90	calc1 calc2	MonDoSwarmMove - Check for the skill flag to stop the sequence, otherwise reset the sequence. Use “calc1” to control the animation starting frame at the beginning of the sequence. Use “calc2” to control the animation frame when ending the sequence.
91	sumoverlay	MonDoNest - Make the caster unit interruptable. Create the monster saved in the function parameter and make that monster unable to reward Experience and Item Drops, and add the overlay on that monster.
92	srvmissilea	MonDoQuickStrike - Apply damage to the target. Check for the monster’s missile frame and then launch the missile.
93	srvmissilea	MonDoGargoyleTrap - Create the missile in one of the allowed directions that is closest to the target unit
94		MonDoSubmerge - When on the last frame, set the animation sequence rate to 0 and clear the frame events
95	srvmissilea calc1 calc3	MonDoMonsterInferno - Continuously create missiles based on the monster’s Inferno fields (See monstats2.txt). Use “calc1” to adjust the missile range. Use “calc3” to adjust the density when to create the next missile.
96	calc1 calc2	MonDoZakarumHeal - Heal the ally target by a random percentage of the target’s life, where “calc1” controls the min percent and “calc2” controls the max percent
97	srvoverlay	MonDoResurrect - Validate that the monster is dead and can be resurrected. Make the monster have proper collision, be attackable, be selectable, able to be hit by missiles, not

		provide experience, and not provide item drops. Also update the monster's mode and add an overlay.
98		MonDoTeleport - Validate the target location and teleport the unit. Adjust the location if the monster has a boss owner.
99	srvmissilea calc1 calc2	MonDoPrimePoisonNova - Create 2 rings of missiles. Use "calc1" to control the number of missile subloops. Use "calc2" to control how many missiles are created per ring. Use the missile's "Param1" and "Param2" values to control its velocity per ring (See Missiles.txt).
100	srvoverlay	MonDoDiabCold - Deal elemental damage to the target and apply the overlay. Adjust the Freeze Length using the "ELen" fields from the skill.
101	srvmissilea calc1	MonDoFingerMageSpider - Create the missile with a facing opposite of the target or caster unit, and use "calc1" to control the missile's subloops.
102	srvmissilea calc1	MonDiabWallMaker - Create a number of missiles with a randomized path and range. Use "calc1" to control the number of missiles created.
103	calc1 Param1 Param2 Param3 Param4 Param5 Param6	MonDoDiabRun - Move the caster unit with increased speed towards a target, and then attack the target, dealing damage. Use "calc1" to control the increase in movement speed. The 6 parameter values controls the run animation's stop frame length, stop event frame, start event frame, start frame length, loop repeat event frame, loop frame length, and loop start event frame.
104	summon	MonDoDiabPrison - Create multiple of the "summon" units around the target, based on the type of unit that is being targeted
105	srvmissilea calc1	MonDoDesertTurret - Create a number of missiles in 8 possible directions. Use "calc1" to control the number of missiles created.
106	srvmissilea	MonDoArcaneTower - Create a circular array of missiles
107	calc3 Param1	MonDoMosquito - Validate that the caster is in melee range for the target and then deal damage, including randomized poison damage, mana drain, and stamina drain. Use "calc3" to control the heal percentage on the caster based on the damage dealt. Use "Param1" to control the animation frame to start the repeat loop.
108	calc1	MonDoRegurgitatorEat - Validate that the target corpse is a monster and then remove it and heal the caster by a percentage of the target's life, controlled by "calc1"
109	aurastate auralencalc calc1	MonDoFrenzy - Attempt to attack the target unit, dealing damage. Add the "aurastate" state to the caster unit, with a duration controlled by "auralencalc". Use "calc1" to control the percentage increase for physical damage dealt.
110	srvmissilea	MonDoHireFireMissile - Launch the weapon missile to the target. Use "srvmissilea" if the weapon missile is "arrow" or "bolt".
111	Param4	MonDoFetishAura - Apply an aura to nearby "fetish1" or "fetishblow1" monster types, increasing their attack rate. Use "Param4" to control the radius.
112	auratargetstate aurarangecalc auaralencalc Param5 Param6 Param7	MonDoCurse - Apply a random curse to enemy units in an area. Randomly select between the following curses: Amplify Damage, Weaken, Life Tap, Decrepify, Lower Resist. Use "aurarangecalc" to control the radius. Use "auralencalc" to control the duration. Use "Param5" and "Param6" to control the resistance percentage change for the Lower Resist curse. Use "Param7" to control the Decrepify attack speed and movement speed change.
113		ItemDoBookSkill - Check the caster's inventory for an item that has the "Book" or "Scroll" Item Type (See ItemTypes.txt), and then use that item's skill and update its quantity.
114	pettype calc1 calc2	DruDoRaven - Create the pet unit and make it unattackable. Use "calc1" to control the pet's bonus Life percent. Use "calc2" to control the summoned pet's unit level.
115	pettype calc1 calc2	DruCreateVineCreature - Create the pet unit in the "Skill 1" mode. Use "calc1" to control the pet's bonus Life percent. Use "calc2" to control the summoned pet's unit level.
116	aurastate auralencalc Param12	DruDoWerewolf - Add/Remove the "aurastate" state on the caster, depending if the caster unit does or does not have the state. Use "auralencalc" to control the state duration. Use "Param12" to control whether you should be able to shift directly to a form without first going back to human form using "1" to allow and "0" to not allow.
117	srvmissilea calc1	DruDoFireStorm - Create a number of missiles that move in a randomized pattern. Use "calc1" to control the number of missiles created.
118	srvmissilea calc1	DruDoTwister - Create a number of missiles that move in a randomized pattern. Use "calc1" to control the number of missiles created.
119	pettype calc1 calc2	DruCreateTotem - Create the pet unit at a valid target location. Use "calc1" to control the pet's bonus Life percent. Use "calc2" to control the summoned pet's unit level.
120	aurastate auralencalc calc2	DruDoFeralRage - After hitting the target, apply the state on the caster unit with a duration controlled by "auralencalc" and with the capability to stack with charges. Use "calc2" to control the max charges.
121	auratargetstate calc1	DruDoRabies - Attack the target to deal damage. Apply the "auratargetstate" where its duration is controlled by the "ELen" field. Use "calc1" to control the percent increase in physical damage.
122	calc1 calc2 calc3	DruDoHunger - Attempt to attack the target, dealing damage and restoring Life and Mana. Use Use "calc1" to control the percent change in physical damage. Use "calc2" and "calc3" to control the life steal and mana steal gained from the attack damage.
123	srvmissilea	DruDoVolcano - Validate the target location and then create the missile
124	aurastate auralencalc Param4	DruDoArmageddon - Apply the "aurastate" state on the caster unit which lasts a duration controlled by "auralencalc". Use "Param4" to control the duration between sending an event state update.
125	srvmissilea	MonDoWakeOfFire - Create a missile from the source unit with a saved direction
126	srvmissilea calc2 Param2	MonDoImpInferno - Based on the animation frames set in monstats2.txt, create a missile where "calc2" controls the missile's range (and also "Param1" from Missiles.txt).

127	calc1	MonDoBatSuckBlood - Deal damage to the target and then use “calc1” to control the percentage of max Life healed on the caster unit
128	calc1 srvoverlay	MonDoCryHelp - If the caster is a monster, then command the caster to attack the target for a duration controlled by “calc1”. Apply the overlay on the target.
129	aurastate	MonDoImpTeleport - If the target location is on the ground, then warp the unit to that location. If the type of unit targeted is the “barricadetower” or “siegebeast1” and the caster unit type is “imp1”, then add the “attached” state to the caster unit, and update the AI, animation events, and stats to be not interactable.
130	srvmissilea calc1	MonDoVineAttack - Create a number of missiles in a random spread pattern with 4 possible spread directions. Use “calc1” to control the number of missiles created.
131	auratargetstate calc1	MonDoOverseerWhip - Validate that the target is a living monster. If the target unit type is “minion1”, then perform a random chance to bloodlust the target. Use “calc1” to control the percent chance to bloodlust the target, which will apply the “auratargetstate” state on the target. If bloodlust does not happen, then change the monster class and AI to the suicide minion.
132	srvmissilea	MonDoImpFireMissile - Based on the monster’s number in class, increment the index of the linked “srvmissilea” missile and then create that missile based on the index.
133		MonDoImpregnate - Validate that the target is a friendly dead monster and that it does not have the “pregnant” state. Then add the “pregnant” state to the target and create a “painworm1” type monster.
134	srvmissilea aurarangecalc	MonDoSiegeBeastStomp - Deal damage to nearby enemies in an aura where “aurarangecalc” controls the radius of the damage
135	sumoverlay	MonDoSpawner - Create the monster saved in the parameter with the added overlay, with no experience provided, and with no item spawning.
136	srvmissilea calc1	MonDoDeathMauler - Validate the target location and then create the missile with an animation rate controlled by “calc1”. The missile’s range is modified based on the distance from the caster.
137	aurastate	MonDoFenrisRage - Validate that the target is an enemy corpse that has not been used. Apply the state on the caster, adding any aura events on the caster.
138	srvmissilea calc1 calc2	MonDoBaalInferno - Shoot multiple missiles from the caster unit to a target location. Use “calc1” to control the number of missiles created. Use “calc2” to control the range of the missiles. Check the inferno frame events for the monster (see monstats2.txt).
139	srvmissilea	MonDoBaalCold - Validate the target location and then create the missile using the directions from the saved parameters.
140		MonDoBaalTentacle - Based on the monster class of the caster, create a number of “baaltentacle1” summoned monsters randomly positioned in a location with a radius size of 9. Make sure these monsters do not provide exp or reward items.
141	aurarangecalc calc1 calc2 calc3 EType Param5 Param6	MonDoBaalCorpseExplosion - Find a nearby dead monster and then do the “NecDoCorpseExplosion” function (Code = 55) except where “Param5” and “Param6” control the baseline radius and radius increase per skill level.
142	prgcalc1 prgcalc2 prgcalc3 aurastate aurastat1 aurarangecalc	AssDoAreaAttack - Deal attack damage in an area. Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, use the proper “progcalc#” value to control the radius, based on the number of progressive charges. Otherwise, default to using “aurarangecalc” for the radius.
143	prgcalc1 prgcalc2 prgcalc3 aurastate aurastat1 srvmissilea srvmissileb srvmissilec	ApplyRoyalStrikeLvl2 - Create a number of missiles that move in a randomized pattern. Use the “srvmissilea” missile by default, or use 1 of the 3 missiles depending on the progressive charges controlled by the “aurastate” field and “aurastat1” fields. Use “aurarangecalc” value by default for the number of missiles created, or use the appropriate progressive calculation fields, based on the number of progressive charges.
144	pettype calc1	SorDoHydra - Validate the target location and create 3 pets, where “calc1” controls their duration and bonus life percent.
145	aurastate aurarangecalc Param4	DruApplyHurricane - Apply the state on the caster with the radius controlled by “aurarangecalc” and the frame state updates to deal radius damage controlled by “Param4”.
146	aurastate aurarangecalc srvmissilea Param4	DruApplyArmageddon - Update the state on the caster unit based on the events, and then create a missile in a radius controlled by “aurarangecalc” at a periodic frame interval controlled by “Param4”.
147		MonApplyAttached - Get the source unit of the caster and update the caster’s pathing to follow the source unit, like a rider attached to its mount.
148	srvmissilea	MonDoDoomKnightMissile - Create the “srvmissilea” using either the lob function or the normal linear function, depending on the “lob” flag. Also select the missile or up to 3 index values higher, depending on the monster’s graphics variation for the “Special 3” component variation.
149	srvmissilea	MonDoNecroMageMissile - Create the “srvmissilea” using either the lob function or the normal linear function, depending on the “lob” flag. Also select the missile or up to 3 index values higher, depending on the monster’s graphics variation for the “Special 4” component variation.
150	calc1 calc2 calc4	PalDoSmite - Validate the target enemy and that the target is in melee range, and then attempt to attack the target. Use “calc1” to control the percent increase for physical damage dealt. Use “calc2” to control the stun length. Use “calc4” to control the percent of damage converted to elemental, if the “Etype” field is used.

151	srvmissilea calc1	SorDoChainLightning2 - Launch the missile that can do chain hits to other nearby enemies. Use “calc1” to control the number of chain hits.
152	srvmissilea calc1 calc3	MonDoDiabloLight - Shoot a missile from the caster to the target location, adhering to the caster’s inferno animation frames (See monstats2.txt). Use “calc1” to control the missile range, otherwise default to using the missile’s “Param2” value calculated with its current level. Use “calc3” to control the periodic frame count for how often to create the missile.

srvstopfunc - Server Stop function. This controls how the skill cleans up after ending. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1	aurastate	BarStopWhirlwind - Handles changing the collision, pathing, and aura state of the caster.

prgstack - Boolean Field. If equals 1, then all “srvprgfunc#” functions will execute, based on the current number of progressive charges. If equals 0, then only the relative “srvprogfunc#” function will execute, based on the current number of progressive charges.
srvprgfunc1 (to srvprgfunc3) - Controls what Server Do function is used when executing the progressive skill with a charge number equal to 1, 2, and 3, respectively. This field uses the same functions as the “srvdofunc” field.
prgcalc1 (to prgcalc3) - Calculation Field. Used as a possible parameter for calculating values when executing the progressive skill with a charge number equal to 1, 2, and 3, respectively.
prgdam - Calls a function to modify the progressive damage dealt

Code	Parameters	Description
0		Do nothing
1	aurastat1 calc1 tgtoverlay	ModifyProgressiveDamage - Modify the percentage of the physical damage dealt and apply an overlay on the target
2	aurastat1 calc1	ModifyProgressiveSteal - Modify the percentage of the life steal and mana steal gained
3	aurastat1 EType Param2	ModifyProgressiveElemental - If the progressive damage elemental type equals Cold, then at 3 charges, modify the Freeze Length based on the Cold Length and a divisor (using Param2)
4	aurastat1 EType calc1 Param5	ModifyProgressiveElementalConvert - Convert a percentage of the physical damage dealt to elemental damage, based on the “calc1” field. If the elemental type equals Cold, then at 3 charges, modify the Freeze Length based on the Cold Length and a divisor (using Param5)

srvmissile - Used as a parameter for controlling what main missile is used for the server functions used (See “Missile” field in Missiles.txt)
decquant - Boolean Field. If equals 1, then the unit’s ammo quantity will be updated each time the skill’s Server Do function is called. If equals 0, then ignore this.
lob - Boolean Field. If equals 1, then missiles created by the skill’s functions will use the missile lobbing function, which will cause the missile fly in an arc pattern. If equals 0, then missiles that are created will use the normal linear function.
srvmissilea (to srvmissilec) - Used as a parameter for controlling what secondary missile is used for the server functions used (See “Missile” field in Missiles.txt)
useServerMissilesOnRemoteClients - Control new missile changes per player skill. Values of 1 will force enable it for that skill. Skills that have matching server/client missiles sets for the skill get auto enabled. Setting this to a value greater than 1 will force it to skip this auto enable logic. If equals 0, then ignore this. Note: this feature is disabled.
srvoverlay - Creates an overlay on the target unit when the skill is used. This is a possible parameter used by various skill functions (See the “overlay” field in Overlay.txt)

aurafilter - Controls different flags that affect how the skill’s aura will affect the different types of units. Uses an integer value to check against different bit fields. For example, if the value equals 4354 (binary = 1000100000010) then that returns true for the 4096 (binary = 1000000000000), 256 (binary = 0000100000000), and 2 (binary = 0000000000010) bit field values.

Bit Field Value	Binary Equivalent Value	Description
1	0000000000000000001	Find Players
2	0000000000000000010	Find Monsters
4	0000000000000000100	Find Undead Monsters
8	0000000000000001000	Find Missiles
16	0000000000000010000	Find Objects
32	0000000000000100000	Find Items
64	0000000000001000000	Limit number of targets
128	0000000000010000000	Attackable units only
256	0000000000100000000	Make sure the target is not in town
512	0000000001000000000	No missile barriers to center
1024	0000000010000000000	Missile units only
2048	0000000100000000000	Custom check function
4096	0000001000000000000	Find dead units only
8192	0000010000000000000	Units not in town
16384	0000100000000000000	No bosses
32768	0001000000000000000	Run a target check function, checking for unit types and player alignment
65536	0010000000000000000	Allies
131072	0100000000000000000	Target in melee range
262144	1000000000000000000	No Act boss units (See the “primeevil” field in monstats.txt)
524288	0000000000000000000	Units not in the Just Hit state

aurastate - Links to a state that can be applied to the caster unit when casting the skill, depending on the skill function used (See the “state” field in states.txt)
auratargetstate - Links to a state that can be applied to the target unit when using the skill, depending on the skill function used (See the “state” field in states.txt)
auralencalc - Calculation Field. Controls the aura state duration on the unit (where 25 Frames = 1 second). If this value is empty, then the state duration will be controlled by other functions, or it will last forever. This can also be used as a parameter for certain skill functions.
aurarangecalc - Calculation Field. Controls the aura state’s area radius size, measured in grid sub-tiles. This can also be used as a parameter for certain skill functions.
aurastat1 (to aurastat6) - Controls which stat modifiers will be altered or added by the aura state (See the “Stat” field from ItemStatCost.txt)

aurastatcalc1 (to aurastatcalc6) - Calculation Field. Controls the value for the relative “aurastat#” field.
auraevent1 (to auraevent3) - Controls what event will trigger the relative “auraeventfunc#” field function. Links to an event listed in the events.txt file.
auraeventfunc1 (to auraeventfunc3) - Controls the function used when the relative “auraevent#” event is triggered.

Code	Parameters	Description
0		Do nothing
1	srvmissilea	SorApplyChillingArmor - Shoot a missile at the target unit
2	cltoverlaya calc1	SorApplyFrozenArmor - Deal skill damage with the freeze length controlled by the calc field and apply an overlay to the attacker
3	cltoverlaya	SorApplyShiverArmor - Deal elemental skill damage and apply an overlay to the attacker
4	auratargetstate calc1 calc2 calc3 calc4	NecApplyIronMaiden - Deal damage to the attacking unit using the calc fields as damage modifiers, based on the state being active. If the target monster is equal to “bloodgolem” then calculate a life steal and also create the “steallife” overlay on the golem’s caster unit.
5	auratargetstate calc1 prgoverlay	NecApplyLifeTap - If the target has the “auratargetstate” state, then apply a percentage heal to the attacker that is determined by the calc field. Also apply an overlay to the attacker when the attack is healed.
6		ItemApplyAttackerTakesDamage - Deal physical damage to the attacker
7		ItemApplyKnockback - Determine a chance to knockback the attacker monster
8		ItemApplyHowl - Apply terror to the monster, changing its AI to run away
9		ItemApplyDimVision - Based on a random chance, apply the “Dim Vision” skill to the enemy attacker with a random skill level between 1 to 20.
10		ItemApplyAttackerTakesLtngDamage - Deal lightning damage to the attacker
11		ItemApplyAttackerTakesFireDamage - Deal fire damage to the attacker
12		ItemApplyAttackerTakesColdDamage - Deal cold damage to the attacker, and apply a cold length that is modified based on the level difference between the attacker and defender
13		ItemApplyDamageToMana - Add mana to the source unit, based on a percentage of the damage taken. Also add a “stealmana” overlay to the source unit.
14		ItemApplyFreeze - Based on a random chance, deal damage to the enemy attacker and apply a freeze length
15		ItemApplyOpenWounds - Based on a random chance, apply open wounds on the target, using the “openwounds” state
16		ItemApplyCrushingBlow - Based on a random chance, deal crushing blow percentage life damage to the enemy attacker and apply the “bash” overlay
17		ItemApplyManaAfterKill - Add mana to the source unit and add a “stealmana” overlay
18		ItemApplyHealAfterDemonKill - If a Demon enemy is killed, then add life to the source unit and add a “steallife” overlay
19		ItemApplySlow - Apply the “slowed” state to the target
20		ItemApplyHitOrAttack - Based on a random chance, cast a skill on the target when the source unit attacks
21		ItemApplyGetHit - Based on a random chance, cast a skill on the attacker when the source unit gets hit
22	aurastate aurastat1 aurastat2	NecApplyBoneArmor - Calculates the minimum and maximum damage to absorb based on the aura stat values. If no remaining absorbing damage is left, then remove the state.
23	calc2 calc3	NecApplyBloodGolemSteal - Calculate a life steal for the attacking unit and its boss unit. Also create the “steallife” overlay on the both units.
24	aurastate calc1 calc2 prgoverlay	SorApplyEnergyShield - If the “aurastate” state is active, then calculate the percentage of damage taken by the source unit that should be absorbs and the amount of mana that should be consumed by the percent absorbed. Also apply an overlay on the source unit. If the source unit runs out of mana, then remove the state.
25	aurastate aurastat1 aurastat2	DruApplyCycloneArmor - Calculates the minimum and maximum elemental damage to absorb based on the aura stat values. If no remaining absorbing damage is left, then remove the state.
26	Param5	NecApplyBloodGolemShare - Controls the percentage of damage taken by the source unit that should be transferred to the source unit’s boss.
27		NecApplyClayGolemSlow - Apply the “slowed” state to the target
28		ItemApplyHealAfterKill - Add life to the source unit and add a “steallife” overlay
29		ItemApplyRestInPeace - Apply the “restinpeace” state on the enemy unit
30		ItemApplyOnDeath - Based on a random chance, cast a skill when the source unit dies
31		ItemApplyReanimate - Based on a random chance, revive a dead monster that is not a champion or unique.
32	aurarangecalc HitClass	SkillApplyRadiusDamage - Use a skill’s damage to deal damage in an area based on the skill’s hit class
33	aurastatcalc1 aurastatcalc2 passiveitype passivereqweaponcount sumskill1 sumsk1calc	SkillActivateSubskill - Based on a random chance controlled by the “aurastatcalc1” value, cast a skill (determined by “sumskill1”) with a skill level controlled by “sumsk1calc”. If “aurastatcalc2” value is greater than 0, then factor the “passiveitype” and “passivereqweaponcount” fields for determining if the skill should be cast or not.

passivestate - Links to a state that can be applied by the passive skill, depending on the skill function used (See the “state” field in states.txt)
passiveitype - Links to an Item Type to define what type of item needs to be equipped in order to enable the passive state (See the “ItemType” field in ItemTypes.txt)
passivereqweaponcount - Controls how many equipped weapons are needed for this passive state to be enabled. If the value equals 1, then the player must have 1 weapon equipped for this passive state to be enabled. If the value equals 2, then the player must be dual wielding weapons for this passive state to be enabled. If the value equals 0, then ignore this field.
passivestat1 (to passivestat10) - Assigns stat modifiers to the passive skill (See the “Stat” field from ItemStatCost.txt)
passivecalc1 (to passivecalc10) - Calculation Field. Controls the value for the relative “passivestat#” field.

summon - Controls what monster is summoned by the skill (See the “Id” field in monstats.txt). This field’s usage will depend on the skill function used. This field can also be used as reference for AI behaviors and for the skilldesc.txt file.
pettype - Links to a pet type data to control how the summoned unit is displayed on the UI (See “pet type” field in pettype.txt)
petmax - Calculation Field. Used skill functions that summon pets to control how many summon units are allowed at a time.
summode - Defines the animation mode that the summoned monster will be initiated with

Token	Description
DT	Death / Reset
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	Block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
GH	Knockback
xx	Sequence
RN	Run

sumskill1 (to sumskill5) - Assigns a skill to the summoned monster. Points to another “skill” id. This can be useful for adding a skill to a monster to transition its synergy bonuses.
sumsk1calc (to sumsk5calc) - Calculation Field. Controls the skill level for the designated “sumskill#” field when the skill is assigned to the monster.
sumumod - Assigns a monster modifier to the summoned monster (See the “id” in monumod.txt)
sumoverlay - Creates an overlay on the summoned monster when it is first created (see the “overlay” field in Overlay.txt)

stsuccessonly - Boolean Field. If equals 1, then the following sound and overlay fields will only play when the skill is successfully cast, instead of always being used even when the skill cast is interrupted. If equals 0, then the following sound and overlay fields will always be used when the skill is cast, regardless if the skill was interrupted or not.
stsound - Controls what client sound is played when the skill is used, based on the client starting function (see the “Sound” field in sounds.txt)
stsoundclass - Controls what client sound is played when the skill is used by the skill’s assigned class (See “charclass” field), based on the client starting function (see the “Sound” field in sounds.txt). If the unit using the skill is not the same class as the “charclass” value for the skill, then this sound will not play.
stsounddelay - Boolean Field. If equals 1, then use the weapon’s hit class to determine the delay in frames (where 25 frames = 1 second) before playing the skill’s start sound. If equals 0, then the skill’s start sound will play immediately.

Hit Class	Sound Used	Sound Delay
None	None	0 frames
Hand-To-Hand	weapon_punch_1	6 frames
One Handed Swing Small	weapon_1hs_small_1	6 frames
One Handed Swing Large	weapon_1hs_large_1	6 frames
Two Handed Swing Small	weapon_2hs_small_1	8 frames
Two Handed Swing Large	weapon_2hs_large_1	8 frames
One Handed Thrust	weapon_1ht_1	6 frames
Two Handed Thrust	weapon_2ht_1	6 frames
Club	weapon_1hs_large_1	6 frames
Staff	weapon_staff_1	6 frames
Bow	None	0 frames
Crossbow	None	0 frames
Claw	None	0 frames

weaponsnd - Boolean Field. If equals 1, then play the weapon’s hit sound when hitting an enemy with this skill. The sound chosen is based on the weapon’s hit class. Also use the sound delay based on the weapon’s hit class to determine the delay in frames (where 25 frames = 1 second) before playing the weapon hit sound (See “stsounddelay” for the types of hit class sounds and delays used). If equals 0, then do not play the weapon hit sound when hitting an enemy with the skill attack.
dosound - Controls the sound for the skill each time the Client Do function is used (see the “Sound” field from sounds.txt)
dosound a & dosound b - Used as a possible parameter for playing additional sounds based on the Client Do function used (see the “Sound” field in sounds.txt)
tgtoverlay - Used as a possible parameter for adding an Overlay on the target unit, based on the Client Do function used (see the “overlay” field in Overlay.txt)
tgtsound - Used as a possible parameter for playing a sound located on the target unit, based on the Client Do function used (see the “Sound” field in sounds.txt)
prgoverlay - Used as a possible parameter for adding an Overlay on the caster unit for progressive charge-up skill functions, based on the Client Do function used and how many progressive charges the caster unit has (see the “overlay” field in Overlay.txt)
prgsound - Used as a possible parameter for playing a sound when using the skill for progressive charge-up skill functions, based on the Client Do function used and how many progressive charges the caster unit has (see the “Sound” field in sounds.txt)
castoverlay - Used as a possible parameter for adding an Overlay on the caster unit when using the skill, based on the Client Start/Do function used (see the “overlay” field in Overlay.txt)
cltoverlaya & cltoverlayb - Used as a possible parameter for adding additional Overlays on the caster unit, based on the Client Start/Do function used (see the “overlay” field in Overlay.txt)

cltstfunc - Client Start function. This controls how the skill works when it is starting to cast, on the client side. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1		StartAttack - Check that the weapon is not a “Missile Potion” item type and if the player has enough ammunition if it is a ranged weapon
2		StartThrow - Check that the player has enough ammunition
3		StartUnsummon - Check that the target is a monster owned by the player and that the monster’s Pet Type has “unsummon” enabled (See pettype.txt)

4		StartLeftAttack - Return true
5		AssStartPsychicHammer - Check that the target is a valid player or monster
6	calc1	AssStartDragonClaw - Validate that the target is a proper enemy, and use the skill's "calc1" field to save the number of kicks to be used by the skill
7	aurastate	AssStartCloakOfShadows - Check that the player does not already have the state
8	prgoverlay prgsound seqinput	AssStartBladeFury - Add the overlay and sound if the player does not have the "inferno" state. If the player does not have the "inferno" state, then add it. Otherwise set the player's animation sequence frame.
9	Param4	AssStartDragonTail - Adjust the player's attack speed using the skill's parameter
10		AssStartDragonFlight - Validate that the target is an enemy monster or player
11		AmaStartCheckQuantity - Check that the player has enough ammunition for the weapon
12		AmaStartJab - Validate the skill and prepare to track the max targets to attack
13	aurarangecalc calc1 calc3	AmaStartStrafe - Use the skill's calculation fields to track the minimum and maximum number of shots. Use the skill's "aurarangecalc" value to count nearby valid targets. Have the caster unit face the first valid target found.
14	calc1	AmaStartFend - Find at least an initial valid target and prepare to track the max targets to attack
15	prgoverlay prgsound	SorStartInferno - Add the overlay and sound if the player does not have the "inferno" state. If the player does not have the "inferno" state, then add it. Otherwise set the player's animation sequence frame.
16	aurarangecalc	SorStartTelekinesis - Check the range of the skill using the "aurarangecalc" value and ensure there is a valid monster or player to target in the area
17		SorStartHydra - Check for a valid area and ensure that the skill cannot be used in town
18	cltmissilec	NecStartCurse - Validate and launch the client missile
19	cltmissilec	NecStartTeeth - Validate and create the missile to launch in a direction based on the cast position
20		NecStartRaiseSkeleton - Check that the target corpse is valid
21		NecStartCExplosion - Check that there is a valid enemy corpse
22		NecStartBonePrison - Check for a valid area and ensure that the skill cannot be used in town
23		NecStartIronGolem - Check that the target item on the ground is valid and that it is identified
24	cltmissilea cltmissileb cltmissilec	NecStartRevive - Validate that the target monster can be revived. Based on the monster's "OverlayHeight" value (See monstats2.txt), create 1 of the 3 client missiles in a random direction. If the monster's "OverlayHeight" value equals 1, then create "cltmissileb". If the monster's "OverlayHeight" value equals 3, then create "cltmissilec".
25	dosound a Param1	PalStartCharge - If the player is in melee range of the target, then start an attack. If the caster unit is a player then play the "dosound a" sound. If the caster unit is a monster, then play the monster's skill sound (see monsounds.txt). Ensure that the target is not in an uninterruptable state. Adjust the movement speed of the caster unit. Set the caster unit's movement velocity speed percent so the skill's "Param1" value. Add movement parameters for the skill function.
26		BarStartFindHeart - Check that the target corpse is valid and has not been used yet
27	cltcalc1	BarStartDoubleSwing - Adjust the caster unit's attack speed based on the "cltcalc1" field
28		BarStartFindItem - Check that the target corpse is valid and has not been used yet
29	aurarangecalc	BarStartLeap - Check that the caster unit is not in an uninterruptible state. If the caster unit is a monster, then find a valid location past the target unit. If the caster unit is not a monster, then find a valid location and ensure that the target location is not in town. Store the target location. The target range is controlled by the "aurarangecalc" field.
30		BarStartLeapAttack - Check that the caster unit is not in an uninterruptible state. If the player is in melee range of the target, then start an attack. Validate the target location, checking for proper player collision. Store the target location.
31		BarStartWhirlwind - Check that the caster unit is not in an uninterruptible state. Modify the caster unit's collision to only collide with walls and objects. Adjust the caster unit's velocity. Add movement parameters for the skill function. Minimum whirlwind distance equals 5.
32	cltmissilea cltmissileb	MonStartMaggotUp - Set the caster unit to be attackable and selectable. If the current area level is in Act 2, then create the "cltmissileb" missile, otherwise create the "cltmissilea" missile. Find a valid location, checking for collision, and then warp the caster unit to that location.
33		MonStartMaggotDown - Set the unit to no longer be attackable or selectable and remove its collision
34		MonStartAndariel - Validate the target enemy and store the target unit's location for the skill function
35		MonStartSwarmMove - Find a valid path with a proper distance, and set movement parameters
36		MonStartNest - Validate the monster class, store the target location, and set the collision in the caster unit's location to be a monster collision
37		MonStartSubmerge - Set the unit to no longer be attackable or selectable
38		MonStartEmerge - Set the unit to be attackable or selectable
39		MonStartResurrect - Unhide the unit
40	cltcalc1	MonStartDiabLight - Use the "cltcalc1" field to calculate a periodic delay for spawning missile and store that value in a parameter
41		MonStartDiabRun - Clear all function flags on the skill
42	calc1 calc2	MonStartMosquito - Validate the target enemy. Use the "calc1" and "calc2" fields as min and max values to randomly select a value to control a loop count, and store that loop count as a parameter for the skill function.
43	cltmissilea	MonStartCurse - Validate and launch the client missile
44		MonStartHellMeteor - Create the following missiles: "hellmeteordown", "hellmeteorball", "hellmeteorup", "hellmeteorball", "hellmeteorlaunch1", "hellmeteorlaunch2"

45	aurastate cltoverlaya cltoverlayb	DruStartWereWolf - Add the “cltoverlayb” overlay to the caster unit if the unit has the “aurastate” state. Otherwise, add the “cltoverlaya” overlay to the caster unit.
46	aurastate aurastat1 cltmissilea prgsound	MonStartBaalTaunt - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile. Create the “baal taunt control” missile. Play the “prgsound” sound.
47	aurastate aurastat1 cltcalc1 cltmissilea cltmissileb cltmissilec	MonStartCatapultDropMissile - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the missile and use the “cltcalc1” value to set the missile’s fall rate.
48	cltmissilea	MonStartTeleport2 - Create the missile at the target location
49	cltmissilea cltmissileb cltcalc1	MonStartVines - Create the “cltmissilea” missile. Use the “cltcalc1” value to control the number of created “cltmissileb” missiles.
50		MonStartDeathSentry - Validate the target enemy and store the target unit’s location for the skill function
51		sMonStartFenrisRage - Validate the target enemy corpse. Store the target unit’s location and unit class for the skill function.
52	calc2	MonStartInfernoSentry - Add the “inferno” state to the caster unit if it is not already added. Use the “calc1” field to control the animation frame tick and store the value for the skill function.
53	calc1	AmaStartFend2 - Find at least an initial valid target and track the max targets to attack using the “calc1” value. Have the caster unit face the target unit.

cltdofunc - Client Do function. This controls how the skill works when it finishes being cast, on the client side. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1		DoAttack - If this is a ranged attack, then launch the client missile. Otherwise, apply damage to the target.
2		DoThrow - If the weapon is a “Missile Potion” item type then launch a missile using the lob function. Otherwise, launch a missile with the normal linear function.
3		AssDoPsychicHammer - Validate the “AssStartPsychicHammer” function
4		AssDoDragonClaw - Check the number of attacks. Roll back the animation by 100%.
5	prgcalc1 prgcalc2 prgcalc3 aurarangecalc aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssDoShockField - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges. Get the range by using the “aurarangecalc” field. Create the calculated number of missiles using the lob function.
6	prgcalc1 prgcalc2 prgcalc3 aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssDoBladeFury - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges. Every periodic delay create a client missile and set the Z position to 15. If the caster unit has the “inferno” state, then repeat the sequence do frame.
7	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssDoDragonTail - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the client missile and set the caster unit’s mode event.
8	aurarangecalc cltmissilea cltmissileb	AssDoMindblast - Create the “cltmissilea” missile at the target location. Use the “aurarangecalc” field to calculate the area radius value of the missile. Set the missile’s spawn class to “cltmissileb”.
9	prgcalc1 prgcalc2 prgcalc3 aurarangecalc aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssDoMissileDisc - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges. Get the radius by using the “aurarangecalc” field. Create a ring of client missiles where the number of missile depends on the size of the radius value.
10	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssClawsOfThunderLvl2 - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the client missile where its velocity is calculated based on the skill level instead of the missile level.
11	prgcalc1 prgcalc2 prgcalc3 aurarangecalc aurastate aurastat1	AssClawsOfThunderLvl3 - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges. Use the “aurarangecalc” field to calculate the number of missiles. Create the client missiles using random pattern directions.

	cltmissilea cltmissileb cltmissilec	
12	prgoverlay prgsound aurastate aurastat1	AssTigerStrike - Based on the “progressive” flag, if the caster unit has the “aurastate” state, or if the “aurastat1” field value is greater than 0, determine whether to add the overlay and sound on the caster unit. If the caster unit has progressive charges, then increase the index of the overlay and the sound by 1 per charge and add those overlay and sounds instead.
13	prgoverlay prgsound aurastate aurastat1	AssCobraStrike - Based on the “progressive” flag, if the caster unit has the “aurastate” state, or if the “aurastat1” field value is greater than 0, determine whether to add the overlay and sound on the target unit. If the caster unit has progressive charges, then increase the index of the overlay and the sound by 1 per charge and add those overlay and sounds instead.
14	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssRoyalStrikeMeteorLvl1 - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the selected client missile at the target location.
15	prgcalc1 prgcalc2 prgcalc3 aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AssRoyalStrikeChaosIce - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges. Create the number of selected client missiles using randomized directions.
16	dosound a dosound b	AmaDoJab - On the frame event of type “Sound”, if the caster unit is a player then play the “dosound a” sound, or if the caster unit is a monster than play the “dosound b” sound
17	cltmissilea cltmissileb calc1 calc2 calc5	AmaDoMultipleShot – If the weapon class is not equal to a “bow” then use “cltmissileb”, otherwise use “cltmissilea”. Use the “calc1” value to determine the number of missiles to create. Use the “calc2” value to determine the activation frame of the missiles. Create the missiles. Use the "calc5" value to calculate Guided Arrow synergy.
18	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	AmaDoGuidedArrow - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the selected client missile with flags of either following a target or going to a location to be retargeted later.
19	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	AmaDoChargedStrike - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “calc1” value to determine the number of missiles to create. Create the missiles that move in a randomized path towards the direction.
20	aurarangecalc cltmissilea cltmissileb Param6	AmaDoStrafe - Use “aurarangecalc” to determine the skill range. The max targets and current target are controlled by a saved parameter. If the weapon class is not equal to a “bow” then use “cltmissileb”, otherwise use “cltmissilea”. Use the “Param6” value to determine the percentage of animation frames to rollback. Find the next target to attack. Create a client missile, making the caster unit face the direction, and update the target count parameter.
21	Param2	AmaDoFend - Use the weapon range to determine the skill range. Find the next valid target to attack and then update the maximum targets to attack next time. Use the “Param2” value to determine the percentage of animation frames to rollback. Based on the hit class, determine what weapon sound to play.
22	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1 calc2	AmaDoLightningStrike - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “calc1” value to determine the range of the missile to find the next target. Use the “calc2” value to determine the maximum number of chain hits for the missile. Create the missile targeting the target unit.
23	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	SorDoChargedBolt - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “calc1” value to determine the number of missiles to create. Create the missiles that move in a randomized path towards the direction.
24	cltmissilea cltmissileb calc1	SorDoInferno - Randomly select between either “cltmissilea” or “cltmissileb”. Use the “calc1” value to determine the range of the missile. Create the client missile and adjust the caster unit’s animation frames. If the caster unit still has the “inferno” state, then repeat the animation do frame.
25	aurastate aurastat1 cltmissilea cltmissileb cltmissilec cltcalc1	SorDoFrostNova - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “cltcalc1” value to determine the additional velocity to add to the missiles created. Create a ring of missiles.
26	cltmissilea calc1	SorDoFirewall - Validate the target location. Validate that the missile create has a missile linked in its “SubMissile1” field (see Missiles.txt). Use the “calc1” value to determine the maximum number of fire wall spawning missiles. Create 2 of the “cltmissilea” missiles thar are launched in opposite directions. Create 1 of the “cltmissilea” missile’s sub missile.

27	aurastate aurastat1 cltmissilea cltmissileb cltmissilec cltcalc1	SorDoChainLightning - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “calc1” value to determine the maximum number of chain hits for the missile. Create the missile targeting the target unit and update the number of chain hits in a parameter.
28	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	SorDoMeteor - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Validate that the target location is valid and then create the missile.
29	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	SorDoFrozenOrb - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Launch the selected client missile.
30	aurarangecalc cltmissilea	NecDoCurse - Use “aurarangecalc” to determine the radius of the skill and always subtract a value of 3 (Min value = 2). Create the client missile at the cursor location and also create a light at the location with RGB values equal to 255, 0, 0.
31	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	NecDoRaiseSkeleton - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Launch the selected client missile facing the directions saved in the parameters.
32	cltmissilea cltmissileb cltmissilec tgtsound	NecDoCExplosion - Create “cltmissilea” at the target location facing a random direction. If the target monster is “small” then spawn 1 “cltmissilea”. If the target monster is “large” then spawn 3 “cltmissilea” missiles in a radius value of 3, and if that missile has a “SubMissile1” value, then also spawn 4 sub missiles in a radius value of 2. If the target monster is neither “small” nor “large”, then spawn 2 “cltmissilea” missiles in a radius value of 2, and spawn 3 of its sub missiles in a radius value of 1. (See Missiles.txt and monstats2.txt). Spawn 1 “cltmissileb” normally. Spawn 1 “cltmissilec” missile with its level equal to 2. If there is no target enemy, then play the “tgtsound” sound.
33	cltmissilea cltmissileb cltmissilec tgtsound	NecDoPoisonExplosion - Create “cltmissilea” at the target location facing a random direction. Spawn an inner and outer radial ring of “cltmissileb” missiles, based on the missile’s “Param1” and “Param2” values (See Missiles.txt). Spawn 1 “cltmissilec” missile with its level equal to 2. If there is no target enemy, then play the “tgtsound” sound.
34	cltmissilea cltcalc1	PalDoSacrifice - Validate the target enemy and spawn the client missile in a random direction, where the missile can receive additional range that is randomly selected between 0 and the “cltcalc1” value.
35	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	PalDoBlessedHammer - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Launch the selected client missile with a spiral path.
36	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	PalDoFistOfHeavens - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Launch the selected client missile and save the target unit type and ID as parameters.
37		PalDoCharge - Setup a sequence of frames to play the animation. If the caster unit is a player or monster, then play its attack sound at the sound frame event. Set the unit’s animation and frame length and order the unit to move to a location or the previously targeted unit. If the player does not need to move, then attack the target or a nearby target.
38		BarDoFindHeart - If the target is valid, then set the mode event and create blood missiles from the monster (see “bleed” in monstats2.txt)
39		BarDoDoubleSwing - If the caster unit’s animation sequence frame is less than 6, then play the weapon sound and have the character turn to face the target. Otherwise, have the caster unit find a proper target and face that target.
40		BarDoFindItem - If the target is valid (see “soft” monstats2.txt), then set the mode event and create blood missiles from the monster (see “bleed” in monstats2.txt)
41	cltmissilea cltmissileb cltmissilec	BarDoGrimWard - If the target is valid (see “soft” monstats2.txt) and there is valid space at that target’s location, then create one of the following missiles. By default, use the “cltmissilea” missile. If the monster is large (see “large” in monstats2.txt), then use the “cltmissilec” missile. If the monster is small (see “large” in monstats2.txt), then use the “cltmissileb” missile. Also, always create a “corpseexplosion” missile.
42		BarDoDoubleThrow - Check that the attacking weapon is throwable and shoot a missile based on the weapon’s missile class. If the weapon’s item type is a “Missile Potion” (see ItemTypes.txt), then use the lobbing missile launch function, otherwise use the normal linear missile launch function.
43		BarDoLeap - Check different flags and parameters to determine when to stop leaping. There are special cases to handle the “sandleadper1” and “ancientbarb1” monsters.
44		BarDoLeapAttack - Check different flags and parameters to determine when to stop leaping. After leaping, if there is a valid target, then attack the target.
45		BarDoWhirlwind - Continue to whirlwind based until at reaching the target location or if the skill flags have been changed.
46		MonDoMagottEgg - Set the unit’s animation sequence rate to 0
47		MonDoMaggotDown - When the unit’s animation frame reaches 0, then set the unit’s animation sequence rate to 0
48	cltmissilea	MonDoAndariel - Based on the unit’s current direction, launch the missile in one of 8 directions

49	calc1 calc2	MonDoSwarmMove - Set the unit's animation sequence start and stop frames based on the skill's calculation values
50		MonDoNest - Remove the monster collision at the target location
51	cltmissilea	MonDoGargoyleTrap - Launch the missile in one of 4 directions.
52		MonDoSubmerge - This equals the "MonDoMaggotDown" function (Code = 47)
53	aurarangecalc cltmissilea cltcalc1	MonDoFetishAura - Create a disc of missiles where "aurarangecalc" controls the disc radius size (Minimum value = 1) and "cltcalc1" controls the density of missiles created (higher value means less missiles).
54	cltmissilea cltmissileb calc1	sMonDoFetishInferno - Randomly create 2 of either "cltmissilea" or "cltmissileb" missiles. Use "calc1" to determine the range of the missile if it is greater than 0, otherwise use the missile's "Param2" value (See Missiles.txt).
55	cltmissilea calc1 calc2	MonDoPrimePoisonNova - Creates 8 missiles in different directions using a velocity set by the missile's "Param1" value (See Missiles.txt). Then uses "calc2" to control how many additional missiles to create using a velocity set by the missile's "Param2" value. Uses "calc1" to set the missile's subloops.
56	cltmissilea cltcalc1 calc1	MonDoDiabLight - Create the missile at an interval controlled by the "cltcalc1" value. Use "calc1" to determine the range of the missile if it is greater than 0, otherwise use the missile's "Param2" value (See Missiles.txt). Also use the monster's inferno values to set the animation frames (see monstats2.txt)
57	cltmissilea calc1	MonDoFingerMageSpider - Create the missile with its subloops controlled by "calc1" and have that missile positioned and face the caster unit
58	cltmissilea calc1	MonDiabWallMaker - Create a number of missiles controlled by the "calc1" value where their pathing and direction is randomized
59	calc1 Param1 Param2 Param3 Param4 Param5 Param6	MonDoDiabRun - Modifies the caster unit's movement speed by a percentage controlled by "calc1" and controls its animations to adhere to this run mode. The 6 parameter values controls the run animation's stop frame length, stop event frame, start event frame, start frame length, loop repeat event frame, loop frame length, and loop start event frame.
60	cltmissilea calc1	MonDoDesertTurret - Fire a number of missiles controlled by the "calc1" value that are directed in 8 possible directions
61	cltmissilea	MonDoArcaneTower - Fire missiles in all possible directions with particles
62	Param1	MonDoMosquito - Check that the caster unit cannot melee the target and then repeat the animation for a number of loops saved in a parameter, where "Param1" controls the frame to repeat the animation.
63	cltmissilea cltmissileb	MonDoRegurgitatorEat - Create 1 "cltmissilea" missile in a random direction. Create 5 "cltmissileb" missiles in a radius value of 4.
64		MonDoQueenDeath - Repeatedly loop the animation using hardcoded frame counts, and then set the unit to Dead mode when finished
65	aurarangecalc cltmissileb	MonDoCurseRadius - Find a valid target and create a "cursecenter" missile. Then create the "cltmissileb" missile with a radius controlled by "aurarangecalc"
66		MonDoHireFireMissile - Use the lob launch function or normal linear launch function when creating the missile
67	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	DruDoFirestorm - Based on the "progressive" flag, the "aurastate" field, or the "aurastat1" field, validate the "cltmissilea" missile or use the other missiles based on the "aurastat1" value. Launch a number of selected client missiles, controlled by the "calc1" value. These missiles has randomized directions and pathing, and they have an increased animation rate.
68	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	DruDoTwister - Based on the "progressive" flag, the "aurastate" field, or the "aurastat1" field, validate the "cltmissilea" missile or use the other missiles based on the "aurastat1" value. Launch a number of selected client missiles, controlled by the "calc1" value. These missiles start with a linear direction and then change to randomized pathing.
69	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	DruDoTornado - Based on the "progressive" flag, the "aurastate" field, or the "aurastat1" field, validate the "cltmissilea" missile or use the other missiles based on the "aurastat1" value. Launch a number of selected client missiles, controlled by the "calc1" value. These missiles start with a linear direction and then change to randomized pathing, and they have an increased animation rate.
70	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	MonDoWakeofFire - Based on the "progressive" flag, the "aurastate" field, or the "aurastat1" field, validate the "cltmissilea" missile or use the other missiles based on the "aurastat1" value. Launch the selected missile in the opposite direction.
71	cltmissilea cltmissileb cltcalc1 cltcalc2	MonDoInferno - Randomly create 1 of either "cltmissilea" or "cltmissileb" missiles. Use "cltcalc1" to control the missile's Z offset. Use "cltcalc2" to control add to the missile's range, which is also determined by the missile's "Param2" value (See Missiles.txt)
72	aurastate aurastat1 cltmissilea cltmissileb cltmissilec cltcalc1	MonDoImpFireBall - Based on the "progressive" flag, the "aurastate" field, or the "aurastat1" field, validate the "cltmissilea" missile or use the other missiles based on the "aurastat1" value. Launch the missile to the target location. Use the "cltcalc1" value to control the missile's Z offset value.
73	cltmissileb	MonDoTeleport2 - Create the missile at the caster unit's location with its skill level set to 1
74	cltmissileb	MonDoTeleport3 - Create the missile at the caster unit's location with its skill level set to 1. Also create a particle and add it to the target unit.

75	Param1 Param2 Param3 Param4	MonDoSiegeBeastStomp - Shake the camera screen, where the parameters control the magnitude, shake build up duration, shake loop duration, and shake fade duration (controlled in frames where 25 frames = 1 second).
76	cltmissilea cltmissileb calc1	MonDoDeathMauler - Continuously create “cltmissileb” missiles as the trail missile’s using the missile’s “Param1” and “Param2” values (See Missiles.txt) to control the frequency and range/frames of the missile’s creation. Create the “cltmissilea” missile with an animation rate controlled by “calc1”, set this missile to not draw, and update its range and activation frame delay based on the “cltmissileb” missile’s “Param1” and “Param2” values.
77	cltmissilea cltmissileb cltcalc1 calc1	MonDoInfernoSentry - Randomly create 1 of either “cltmissilea” or “cltmissileb” missiles. Use “cltcalc1” to control the missile’s Z offset. Use “calc1” to control the missile range duration. Repeat the Do frame while the caster unit has the “inferno” state.
78	cltmissilea cltmissileb prgsound	MonDoDeathSentry - Create the “cltmissilea” missile at the target location and play the “prgsound” sound. Then create the “cltmissileb” missile at the target location.
79	cltmissilea	MonDoFenrisRage - Create the “cltmissilea” missile at the target location
80	aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1 calc2	MonDoBaalInferno - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create a number of the selected missiles where “calc1” controls the number of missiles created, and “calc2” controls the range duration of the missiles. Repeat the Do frame while the caster unit has the “inferno” state.
81	aurastate aurastat1 cltmissilea cltmissileb cltmissilec	MonDoBaalCold - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Launch the selected missile to the target location.
82	Param5 Param6	MonStartBaalCExplode - Use “Param5” and “Param6” to calculate the baseline and increase per skill level change in the radius to find a target. Search for valid dead targets and create a “baalcorpseexplodedelay” missile for each target found.
83	prgcalc1 prgcalc2 prgcalc3 aurastate aurastat1 cltmissilea cltmissileb cltmissilec aurarangecalc Param2	ApplyRoyalStrikeLvl2 - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Get the number of missiles using the “progcalc#” values, based on the number of progressive charges, and if they are equal to or less than 0, then use the “aurarangecalc” value instead. Create the missile, using the “Param2” value to control the number of chain hits that the missile bounces.
84	cltmissilea prgoverlay	SorDoThunderStorm - Launch the missile at the target’s position, with a starting Z position equal to 280 and a starting velocity equal to -40. Also add the overlay to the target unit.
85		ItemDoOpenWounds - Create a blood missile from the unit every 5 frames (see “bleed” in monstats2.txt)
86	aurastate aurastat1 cltmissilea cltmissileb cltmissilec aurarangecalc	PalDoSanctuary - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Use the “aurarangecalc” value to determine the radius to randomly create missiles and also the number of missiles to create.
87	aurastate aurastat1 cltmissilea cltmissileb cltmissilec cltcalc1 cltcalc2 cltcalc3	SorApplyShiverArmor - Based on the “progressive” flag, the “aurastate” field, or the “aurastat1” field, validate the “cltmissilea” missile or use the other missiles based on the “aurastat1” value. Create the selected missile using the lob function. Use “cltcalc1” to control the delay between creating missiles. Use “cltcalc2” to control the radius to create the missile. Use “cltcalc3” to control the vertical height of the missile when it is created.
88		MonApplyAttached - Attach the source unit on the caster unit.
89	cltmissilec cltmissiled cltcalc1	MonDoVineBeast - Validate that the caster unit is a monster type. Validate that the skill used is the “Vine Attack” skill and get its stats, otherwise use a default level 1 “Vine Attack” skill. Use the “cltcalc1” value to determine the frame delay between creating missiles and also the minimum distance between missiles created. If the current mode of the unit is “Walk” then randomly choose between creating the “cltmissilec” or “cltmissiled” missile. If the current mode of the unit is “Neutral” then kill any existing old missile and create the “cltmissiled”.
90	cltmissilea cltmissileb cltmissilec prgsound cltcalc1 cltcalc2	DruDoHurricane - Play the “prgsound” sound and stop playing it if the caster unit is in town. Use “cltcalc1” to how many sets of 3 missiles to create at a time, at a random height. Randomly choose between one of the 3 missiles to create each time, and use “cltcalc2” to control the maximum Z height position of the missile.
91	cltmissilea cltmissileb	DruDoVolcano - Create both missiles at the valid target position
92	cltmissilea cltmissileb cltcalc1 cltcalc2	DruDoArmageddon - Create both missiles with a negative fall rate and slide rate. Use the “aurarangecalc” value to control a radius value, but this value is not used. Use the “cltcalc1” value to control the frame count for determining the starting height and offset. Use the

	cltcalc3 aurarangecalc	“cltcalc2” value to control the missile fall rate per frame. Use the “cltcalc3” value to control the missile slide rate per frame.
93	cltmissileb	MonDoCorpseCycler - Validate the target unit, and create the “cltmissileb” missile on the target and 3 blood missiles (see “bleed” in monstats2.txt)
94	cltmissilea	MonDoDoomKnightMissile - Create the “cltmissilea” using either the lob function or the normal linear function, depending on the “lob” flag. Also select the missile or up to 3 index values higher, depending on the monster’s graphics variation for the “Special 3” component variation.
95	cltmissilea	MonDoNecroMageMissile - Create the “cltmissilea” using either the lob function or the normal linear function, depending on the “lob” flag. Also select the missile or up to 3 index values higher, depending on the monster’s graphics variation for the “Special 4” component variation.
96	aurarangecalc aurastate aurastat1 cltmissilea cltmissileb cltmissilec calc1	SorDoChainLightning2 - Based on the “progressive” flag or the “aurastate” field, create the “cltmissilea” missile. Otherwise, use one of the 3 client missiles based on the “aurastat1” value. Create the missile and use the “calc1” value to set the number of chain hits.

cltstopfunc - Client Stop function. This controls how the skill cleans up after ending. This uses a code value to call a function, affecting how certain fields are used.

Code	Parameters	Description
0		Do nothing
1		SkillBarRemoveWhirlwind - Handles changing the collision and pathing of the caster, and also stops the whirlwind sound.

cltprgfunc1 to (cltprgfunc3) - Controls which Client Do function is used when the skill is executed while having a progressive charge number equal to 1, 2, and 3, respectively. (uses “cltdofunc” values)
cltmissile - Used as a parameter for controlling what main missile is used for the client functions used (See “Missile” field in Missiles.txt)
cltmissilea (to cltmissiled) - Used as a parameter for controlling what secondary missile is used for the client functions used (See “Missile” field in Missiles.txt)
cltcalc1 (to cltcalc3) - Calculation Field. Use as a possible parameter for controlling values for the client functions.

warp - Boolean Field. If equals 1 and the skill uses a function that involves warping/teleporting, then check for a scene transition loading screen, if necessary. If equals 0, then ignore this.
immediate - Boolean Field. If equals 1 and the skill has a periodic function, then immediately perform the skill’s function when the periodic skill is activated. If equals 0, then wait until the skill’s first periodic delay before performing the skill’s function.
enhanceable - Boolean Field. If equals 1, then the skill will be included in the plus to all skills item modifier. If equals 0, then the skill will not be included in the plus to all skills item modifier.
attackrank - Controls the skill’s AI score value for determining what is the best target for the AI. The higher the value, then the more likely the AI will select a target with this skill equipped.

noammo - Boolean Field. If equals 1, then the skill will not check that weapon’s ammo when performing an attack. This relies on the “Shoots” field from the ItemType.txt file. If equals 0, then the weapon will consume its ammo when being used by the skill.
range - Determines how the unit uses the skill, which affects the weapon requirements and the type of attack used

Code	Description
none	No restrictions to use the skill
h2h	This is a melee skill, which requires a melee weapon
rng	This is a ranged skill, which requires a ranged weapon
both	This is both a melee and ranged skill, which can use either a melee or ranged weapon
loc	This is a location based skill, which ignores the weapon equipped and instead uses the player’s location

weapsel - If the unit can dual wield weapons, then this field will control how the weapons are used for the skill

Code	Description
0	Use the Left or Right Hand weapon
1	Use the Left Hand weapon
2	Use the Left and/or Right Hand weapon
3	Always use both weapons
4	Ignore the weapon used

itypea1 (to itypea3) - Controls what Item Types are included, or allowed, when determining if this skill can be used (See the “Code” field from ItemTypes.txt)
etypea1 & etypea2 - Controls what Item Types are excluded, or not allowed, when determining if this skill can be used (See the “Code” field from ItemTypes.txt)
itypeb1 (to itypeb3) - Controls what alternate Item Types are included, or allowed, when determining if this skill can be used (See the “Code” field from ItemTypes.txt). This acts as a second set of Item Types to check.
etypeb1 & etypeb2 - Controls what alternate Item Types are excluded, or not allowed, when determining if this skill can be used (See the “Code” field from ItemTypes.txt). This acts as a second set of Item Types to check.

anim - Controls the animation mode that the player character will use when using this skill. Setting the mode to Sequence (SQ) will cause the player character to play a time controlled animation sequence, utilizing certain sequence fields.

Code	Description
DT	Death
NU	Neutral
WL	Walk
RN	Run
GH	Get Hit
TN	Town Neutral
TW	Town Walk
A1	Attack 1
A2	Attack 2
BL	Block

SC	Cast
TH	Throw
KK	Kick
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
SQ	Sequence
KB	Knockback

seqtrans - Uses the same inputs as the “anim” field. If the “anim” field equals SQ (Sequence) and this field equals SC (Cast), then the sequence animation speed can be modified by the faster cast rate stat modifier.

monanim - Controls the animation mode that the monster will use when using this skill. This is similar to the “anim” field except with monster units using the skill instead of player units.

Code	Description
DT	Death
NU	Neutral
WL	Walk
GH	Get Hit
A1	Attack 1
A2	Attack 2
BL	block
SC	Cast
S1	Skill 1
S2	Skill 2
S3	Skill 3
S4	Skill 4
DD	Dead
KB	Knockback
xx	Sequence
RN	Run

seqnum - If the unit is a player and the “anim” used for the skill is a Sequence, then this field will control the index of which sequence animation to use. These sequences are specifically designed for certain skills, and each sequence has a set number of frames using certain animations.

Code	Description
0 (or empty)	Do nothing
1	Jab
2	Sacrifice
3	Chastise
4	Charge
5	Defiance
6	Inferno
7	Strafe
8	Impale
9	Fend
10	Whirlwind
11	Double Swing
12	Lightning
13	Leap
14	Leap Attack
15	Double Throw
16	Tiger Fist
17	Projection
18	Arctic Blast
19	Triple Kick
20	Dragon Breath
21	Dragon Flight
22	Druid Unmorph
23	Assassin Blade Fury

seqinput - For skills that can repeat, this controls the number of frames to wait before the “Do” frame in the sequence. This acts as a delay in frames (25 Frames = 1 second) to wait within the sequence animation before it is allowed to be cast again or for looping back to the start of the sequence, such as for the Sorceress Inferno skill.

durability - Boolean Field. If equals 1 and when the player character ends an animation mode that was not Attack 1, Attack 2, or Throw, then check the quantity and durability of the player’s items to see if the valid weapons are out of ammo or are broken. If equals 0, then ignore this.

UseAttackRate - Boolean Field. If equals 1, then the current attack speed should be updated after using the skill, relative to the “attackrate” stat (See ItemStatCost.txt), and if the skill was an attacking skill. If equals 0, then the attack speed will not be updated after using the skill.

LineOfSight - Controls the skill’s collision filters for valid target locations when it is being cast

Code	Description
0	No collision filter
1	Missile Barrier
2	Player Path - Walls, no pathing, objects, doors, no player pathing
3	Player Monster - Monsters, Players
4	Player Flying - Missile barriers, doors
5	Radial Barrier - Missile barriers, doors, walls

TargetableOnly - Boolean Field. If equals 1, then the skill will require a target unit in order to be used. If equals 0, then ignore this.
SearchEnemyXY - Boolean Field. If equals 1, then when the skill is cast on a target location, it will automatically search in different directions in the target area to find the first available enemy target. If equals 0, then ignore this. This field can be overridden if the “SearchEnemyNear” field is enabled.
SearchEnemyNear - Boolean Field. If equals 1, then when the skill is cast on a target location, it will automatically find the nearest enemy target. If equals 0, then ignore this.
SearchOpenXY - Boolean Field. If equals 1, then automatically search in a radius of size 7 in around the clicked target location to find an available unoccupied location to use the skill, testing for collision. If equals 0, then ignore this. This field can be overridden if the “SearchEnemyNear” or “SearchEnemyXY” field is enabled.
SelectProc - Uses a function to check that the target is a corpse with various parameters before validating the usage of the skill

Code	Description
0	Do nothing
1	CorpseExplode - Check that the target is a corpse
2	RaiseSkeleton - Check that the target is a monster corpse, and that the corpse was a unit that has a velocity (if the unit does not move, then the corpse cannot be used)
3	Revive - Use the RaiseSkeleton function (Code = 2) and that the monser has a switchable AI
4	HeartMonster - Check that the target is a monster corpse and that the monster has the “soft” flag enabled (see monstats2.txt)
5	ItemMonster - Check that the target is a monster corpse
6	WardMonster - Check that the target is a monster corpse and that the monster has the “soft” flag enabled (see monstats2.txt)

TargetCorpse - Boolean Field. If equals 1, then the skill is allowed to target corpses. If equals 0, then skill cannot target corpses.
TargetPet - Boolean Field. If equals 1, then the skill is allowed to target pets (summons and mercenaries). If equals 0, then the skill cannot target pets.
TargetAlly - Boolean Field. If equals 1, then the skill is allowed to target ally units. If equals 0, then the skill cannot target ally units.
TargetItem - Boolean Field. If equals 1, then the skill is allowed to target items on the ground. If equals 0, then the skill cannot target items.
AttackNoMana - Boolean Field. If equals 1, then then when the caster does not have enough mana to cast the skill and attempts to use the skill, the caster will default to using the Attack skill. If equals 0, then attempting to use the skill without enough mana will do nothing.
TgtPlaceCheck - Boolean Field. If equals 1, then check that the target location is available for spawning a unit, testing collision. If equals 0, then ignore this. This is only used for skills that monsters use to spawn new units.

KeepCursorStateOnKill - Boolean Field. If equals 1, then the mouse click hold state will continue to stay active after killing a selected target. If equals 0, then after killing the target, the mouse click hold state will reset.
ContinueCastUnselected - Boolean Field. If equals 1, then while the mouse is held down and there is no more target selected, then the skill will continue being used at the mouse’s location. If equals 0, then while the mouse is held down and there is no more target selected, then the player character will cancel the skill and move to the mouse location.
ClearSelectedOnHold - Boolean Field. If equals 1, then when the mouse is held down, the target selection will be cleared. If equals 0, then ignore this.

ItemEffect - Uses a Server Do function (See “srvdofunc”) for handling how the skill is used when it is triggered by an item, on the server side.
ItemCltEffect - Uses a Client Do function (See “cltdofunc”) for handling how the skill is used when it is triggered by an item, on the client side.
ItemTgtDo - Boolean Field. If equals 1, then use the skill from the enemy target instead of the caster. If equals 0, then ignore this.
ItemTarget - Controls the targeting behavior of the skill when it is triggered by an item.

Code	Description
0 (or empty)	Default to targeting the attacker
1	Target the caster
2	Target a random location in a radius with a size of 20. Also tests collision.
3	Target a random nearby corpse
4	Target the attacker, and if that attacker is not found then target a previous attacker or the previous attacker’s location

ItemUseRetrict – Boolean Field. If equals 1, then use the state restriction defined in the “restrict” field when being triggered by an item.
ItemCheckStart - Boolean Field. If equals 1, then use the skill’s Server Start function (See “srvstfunc”) when the skill is trigged by an item. If equals 0, then the skill’s Server Start function is ignored.
ItemCltCheckStart - Boolean Field. If equals 1, then when the skill is triggered by an item, and if the target is dead and the skill has a Client Start function (See “cltstfunc”), then add the “corpse_noselect” to the target. If equals 0, then ignore this.
ItemCastSound - Play a sound when the skill is used by an item event. Points to a “Sound” value in the sounds.txt file.
ItemCastOverlay - Add a cast overlay when the skill is used by an item event. Points to an “overlay” value in the Overlay.txt file.

skpoints - Controls the number of Skill Points needed to level up the skill
reqlevel - Minimum character level required to be allowed to spend Skill Points on this skill
maxlvl - Maximum base level for the skill from spending Skill Points. Skill levels can be increased beyond this through item modifiers.
reqstr - Minimum Strength attribute required to be allowed to spend Skill Points on this skill
reqdex - Minimum Dexterity attribute required to be allowed to spend Skill Points on this skill
reqint - Minimum Intelligence attribute required to be allowed to spend Skill Points on this skill
reqvit - Minimum Vitality attribute required to be allowed to spend Skill Points on this skill
reqskill1 (to reqskill3) - Points to a “skill” field to act as a prerequisite skill. The prerequisite skill must be least base level 1 before the player is allowed to spend Skill Points on this skill.

restrict - Controls how the skill is used when the unit has a restricted state (See the “restrict” field in states.txt)

Code	Description
0	None - The skill cannot be used when the unit has a restricted state
1	Any - The skill can be used in when the unit has any restricted state
2	State Only - The skill can only be used when the unit has a restricted state (See “State1” to “State3”)
3	Pop State - The skill can be used at any time but when used, it will remove the unit’s restrict states

State1 (to State3) - Points to a “state” field from the states.txt file. Used as parameters for the “restrict” field to control what specific states will restrict the usage of the skill.

localdelay - Controls the Casting Delay duration for this skill after it is used (25 frames = 1 second)
globaldelay - Controls the Casting Delay duration for all other skills with Casting Delays after this skill is used (25 frames = 1 second)

leftskill - Boolean Field. If equals 1, then the skill will appear on the list of skills to assign for the Left Mouse Button. If equals 0, then the skill will not appear on the Left Mouse Button skill list.
rightskill - Boolean Field. If equals 1, then the skill will appear on the list of skills to assign for the Right Mouse Button. If equals 0, then the skill will not appear on the Right Mouse Button skill list.

repeat - Boolean Field. If equals 1 and the skill has no “anim” mode declared, then on the client side, the unit’s mode will repeat its current mode (this can also happen if the unit has the “inferno” state). If equals 0, then the unit will have its mode set to Neutral when starting to use the skill.
alwayshit - Boolean Field. If equals 1, then skills that rely on attack rating and defense will ignore those stats and will always hit enemies. If equals 0, then ignore this.

usemanaondo - Boolean Field. If equals 1, then the skill will consume mana on its do function instead of its start function. If equals 0, then the skill will consume mana on its start function, which is the general case for skills.
startmana - Controls the required amount of mana to start using the skill. This only works with certain “srvstfunc” functions such as SorStartInferno (Code = 11) or AssStartBladeFury (Code = 26).
minmana - Controls the minimum amount of mana to use the skill. This can control skills that reduce in mana cost per level to not fall below this amount.
manashift - This acts as a multiplicative modifier to control the precision of the mana cost after calculating the total mana cost with the “mana” and “lvlmana fields”. Mana is calculated in 256ths in code which equals 8 bits. This field acts as a bitwise shift value, which means it will modify the mana value by the power of 2. For example, if this value equals 5 then that means divide the mana value of 256ths by 2^5 = 32 (or multiply the mana by 32/256). A value equal to 8 means 256/256 which means that the mana of 256ths value is not shifted.
mana - Defines the base mana cost to use the skill at level 1
lvlmana - Defines the change in mana cost per skill level gained

interrupt - Boolean Field. If equals 1, then the casting the skill will be interruptible such as when the player is getting hit while casting a skill. If equals 0, then the skill should be uninterruptible.
InTown - Boolean Field. If equals 1, then the skill can be used while the unit is in town. If equals 0, then the skill cannot be used in town.
aura - Boolean Field. If equals 1, then the skill will be classified as an aura, which will make the skill execute its function periodically (using the “perdelay” field), based on the if the “aurastate” state is active. Aura skills will also override a preexisting state if that state matches the skill’s “aurastate” state. If equals 0, then ignore this.
periodic - Boolean Field. If equals 1, then the skill will execute its function periodically (using the “perdelay” field), based on the if the “aurastate” state is active. If equals 0, then ignore this.
perdelay - Calculation Field. Controls the periodic rate that the skill continuously executes its function. Minimum value equals 5. This field requires “periodic” or “aura” field to be enabled.
finishing - Boolean Field. If equals 1, then the skill will be classified as a finishing move, which can affect how progressive charges are consumed when using the skill and how the skill’s description tooltip is displayed. If equals 0, then ignore this.
prgchargestocast - Controls how many progressive charges are required to cast the skill
prgchargesconsumed - Controls how many progressive charges are consumed when the skill attack hits an enemy
passive - Boolean Field. If equals 1, then the skill will be treated as a passive, which can mean that the skill will not show up in the skill selection menu and will not run a server do function. If equals 0, then the skill is an active skill that can be used.
progressive - Boolean Field. If equals 1, then the skill will use the progressive calculations to act as a charge-up skill that will add charges. This is only used for certain skill functions and will generally require the usage of the “progcalc#” fields and the “aurastat#” fields. If equals 0, then ignore this.
scroll - Boolean Field. If equals 1, then the skill can appear as a scroll version in the skill selection UI, if the skill allows for the scroll mechanics and if the player has the skill’s scroll item in the inventory. If equals 0, then ignore this.

calc1 (to calc6) - Calculation Field. It is used as a possible parameter for skill functions or as a numeric input for other calculation fields.
Param1 (to Param12) - Integer Field. It is used as a possible parameter for skill functions or as a numeric input for other calculation fields.

InGame - Boolean Field. If equals 1, then the skill is enabled to be used in-game. If equals 0, then the skill will be disabled in-game.
ToHit - Baseline bonus Attack Rating that is added to the attack when using this skill at level 1
LevToHit - Additional bonus Attack Rating when using this skill, calculated per skill level
ToHitCalc - Calculation Field. Calculates the bonus Attack Rating when using the skill. This will override the “ToHit” and “LevToHit” fields if it is not blank.
ResultFlags - Controls different flags that can affect how the target reacts after being hit by the skill attack. Uses an integer value to check against different bit fields by using the “&” operator. For example, if the value equals 5 (binary = 101) then that returns true for both the 4 (binary = 100) and 1 (binary = 1) bit field values.

Bit Field Value	Binary Equivalent Value	Description
1	0000000000000001	Hit
2	0000000000000010	Death
4	0000000000000100	Get Hit
8	0000000000001000	Knockback
16	0000000000010000	Block
32	0000000000100000	No Passive
128	0000000010000000	Dodge
256	0000000100000000	Avoid
512	0000001000000000	Evade
4096	0001000000000000	Ignore Friendly
8192	0010000000000000	Double Damage
16384	0100000000000000	Soft Hit
32768	1000000000000000	Two Hand-to-Hand Block

HitFlags - Controls different flags that can affect the damage dealt when the target is hit by the skill. Uses an integer value to check against different bit fields by using the “&” operator. For example, if the value equals 6 (binary = 110) then that returns true for both the 4 (binary = 100) and 2 (binary = 10) bit field values.

Bit Field Value	Binary Equivalent Value	Description
1	00000000001	Do not add physical damage

2	00000000010	Do not add any damage
4	00000000100	No Life Steal
8	00000001000	No Mana Steal
16	00000010000	No Stamina Steal
32	00000100000	Use Source Damage
128	00010000000	No Triggered Events
256	00100000000	Bypass Undead
512	01000000000	Bypass Demons
1024	10000000000	Bypass Beasts

HitClass - Defines the skill's damage routines when hitting, mainly used for determining hit sound effects and overlays. Uses an integer value to check against different bit fields by using the "&" operator. For example, if the value equals 6 (binary = 110) then that returns true for both the 4 (binary = 100) and 2 (binary = 10) bit field values. There are binary masks to check between Base Hit Classes and Hit Class Layers, which can distinguish bewteen overlays or sounds are displayed.

Bit Field Value	Binary Equivalent Value	Description
Base Hit Classes		
0	00000000	None
1	00000001	Hand to Hand
2	00000010	One Handed Swing Small
3	00000011	One Handed Swing Large
4	00000100	Two Handed Swing Small
5	00000101	Two Handed Swing Large
6	00000110	One Handed Thrust
7	00000111	Two Handed Thrust
8	00001000	Club
9	00001001	Staff
10	00001010	Bow
11	00001011	Crossbow
12	00001100	Claw
13	00001101	Do Overlay
Hit Class Layers		
16	00010000	Double Layer
32	00010100	Fire Layer
48	00011110	Cold Layer
64	01000000	Lightning Layer
80	01010000	Poison Layer
96	01100000	Stun Layer
112	01110000	Bash Layer
128	10000000	Thorns Layer
144	10010000	Sanctuary Layer
160	10100000	Silent Voice Layer
176	10110000	Goo Layer

Kick - Boolean Field. If equals 1, then a separate function is used to calculate the physical damage dealt by the skill, ignoring the following damage fields. This function will factor in the player character's Strength and Dexterity attributes (or Monster's level) to determine the baseline damage dealt. If equals 0, then ignore this.

HitShift - Controls the percentage modifier for the skill's damage. This value cannot be less than 0 or greater than 8. This is calculated in 256ths.

Value	Description	Percentage
8	256/256	100%
7	128/256	50%
6	64/256	25%
5	32/256	12.5%
4	16/256	6.25%
3	8/256	3.125%
2	4/256	1.5625%
1	2/256	.78125%
0	1/256	.390625%

SrcDam - Controls the percentage modifier for how much weapon damage is transferred to the skill's damage (Out of 128). For example, if the value equals 64, then 50% (64/128) of the weapon's damage will be added to the skill's damage.

MinDam - Minimum baseline physical damage dealt by the skill

MinLevDam1 (to MinLevDam5) - Controls the additional minimum physical damage dealt by the skill, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field number.

MaxDam - Maximum baseline physical damage dealt by the skill

MaxLevDam1 (to MaxLevDam5) - Controls the additional maximum physical damage dealt by the skill, calculated using the leveling formula between 5 level thresholds of the missile's current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field number.

DmgSymPerCalc - Calculation Field. Determines the percentage increase to the physical damage dealt by the skill.

EType - Defines the type of elemental damage dealt by the skill. If this field is empty, then the related elemental fields below will not be used.

Code	Description
(empty)	None
fire	Fire
ltng	Lightning
mag	Magic
cold	Cold (Uses "ELen")
pois	Poison (Uses "ELen")
life	Life Drain

mana	Mana Drain
stam	Stamina Drain
stun	Stun (Uses “ELen”)
rand	Randomly select between Fire, Lightning, Magic, Cold, or Poison
burn	Burning (Uses “ELen”)
frze	Freeze (Uses “ELen”)

EMin - Minimum baseline elemental damage dealt by the skill
EMinLev1 (to EMinLev5) - Controls the additional minimum elemental damage dealt by the skill, calculated using the leveling formula between 5 level thresholds of the skill’s current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field number.
EMax - Maximum baseline elemental damage dealt by the skill
EMaxLev1 (to EMaxLev5) - Controls the additional maximum elemental damage dealt by the skill, calculated using the leveling formula between 5 level thresholds of the missile’s current level. The level thresholds are levels 2-8, 9-16, 17-22, 23-28, 29 and beyond. These 5 level thresholds correlate to each field.
EDmgSymPerCalc - Calculation Field. Determines the percentage increase to the total elemental damage dealt by the skill.
ELen - The baseline elemental duration dealt by the skill. This is calculated in frame lengths where 25 Frames = 1 second. These fields only apply to appropriate elemental types with a duration.
ELevLen1 (to ELevLen3) - Controls the additional elemental duration added by the skill, calculated using the leveling formula between 3 level thresholds of the missile’s current level. The level thresholds are levels 2-8, 9-16, 17 and beyond. These 3 level thresholds correlate to each field number. These fields only apply to appropriate elemental types with a duration.
ELenSymPerCalc - Calculation Field. Determines the percentage increase to the total elemental duration dealt by the skill.

aitype - Controls what the skill’s archetype for how the AI will handle using this skill. This mostly affects the mercenary AI and Shadow Warrior AI, but some types are used for general AI.

Code	Description
0	None
1	Buff
2	Debuff
3	Summon
4	Melee
5	Ranged
6	Aura
7	Teleport
8	Heal
9	Resurrect
10	Passive
11	Area Range
12	Steal
13	Move Attack

aibonus - This is only used with the Shadow Master AI. This value is a flat integer rating that gets added to the AI’s parameters when deciding which skill is most likely to be used next. The higher this value, then the more likely this skill will be used by the AI.

cost mult - Multiplicative modifier of an item’s gold cost, only when the item has a stat modifier with this skill. This will affect the item’s buy, sell, and repair costs (Calculated in 1024ths).
cost add - Flat integer modification of an item’s gold cost, only when the item has a stat modifier with this skill. This will affect the item’s buy, sell, and repair costs. This is added after the “cost mult” has modified the costs.

skilldesc.txt

Overview

This file controls a skill’s tooltip description and how it is displayed on the Skill Tree

Used by the following data files: Missiles.txt, Monstats.txt, skills.txt

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

skilldesc - The name of the skill description, as a reference for associated Data files
SkillPage - Determines which page on the Skill tree to display the skill

Code	Description
0	Do not display on any skill tree page
1	Skill Tree Page 1
2	Skill Tree Page 2
3	Skill Tree Page 3

SkillRow - Determines which row on the Skill tree page to display the skill

Code	Description
0	Do not display on the skill tree page
1	Row 1
2	Row 2
3	Row 3
4	Row 4

5	Row 5
6	Row 6

SkillColumn - Determines which column on the Skill tree page to display the skill

Code	Description
0	Do not display on any skill tree page
1	Left Column
2	Middle Column
3	Right Column

ListRow - Determines which row the skill will be listed in, for the skill select UI.

Code	Description
0	Common Skill Row
1	Skill Tree 1 Row
2	Skill Tree 2 Row
3	Skill Tree 3 Row
(Other)	Skill will not appear in the Skill Selection UI

IconCel - Determines the icon asset for displaying the skill. This requires an ID value based on the skillicon files. Class specific skills use their designated class skillicon files (controlled by the “charclass” field in skills.txt), and non-class skills use the global skillicon file. This will use the value as the standard icon to display, and the next value (value + 1) as the button pressed icon display.

HireableIconCel - Determines the icon asset for displaying the skill on a hireable (mercenary). This requires an ID value based on the hireable skillicon files. This will use the value as the standard icon to display, and the next value (value + 1) as the button pressed icon display.

str name - Uses a string to display as the skill name
str short - Uses a string to display as the skill description in shortcuts or when selecting a skill
str long - Uses a string to display as the skill description on the Skill Tree
str alt - Uses a string to display the skill name on the Character Screen when the skill is selected

descdam - Use a function to calculate a skill’s damage and determine how to display it. These functions sometimes require certain skill fields, especially the damage related fields.

Code	Parameters	Description
0		null
1	ddam calc1 ddam calc2	Calculates the basic Attack damage (Uses function 7) “ddam calc1” is used as a percent bonus “ddam calc2” is used as a flat number bonus
2		Calculates the character’s kick damage
3		Calculates the character’s throwing weapon damage
4		Calculates the character’s left throwing weapon damage
5		Calculates damage using the equipped weapon damage and the linked skill’s physical and elemental damage
6		Similar to function 5 Calculates the damage of the skill but with carry-over of elemental damage from the source, such as a missile direct hit and then a missile explosion
7	ddam calc1 ddam calc2	Calculates the damage of a skill, including damage bonuses “ddam calc1” is used as a percent bonus “ddam calc2” is used as a flat number bonus
8	ddam calc1 ddam calc2	Calculates the elemental damage of a skill as a periodic damage (every 25 frames = 1 second) “ddam calc1” is used as a multiplier of the damage (If equals 0 then default to 1) “ddam calc2” is used as a divisor of the damage (If equals 0 then default to 1)
9		Calculates the elemental damage of a skill as a periodic damage (every 25 frames = 1 second) The damage is always multiplied by 3
10		Calculates damage based on the shield equipped and the damage provided by the skill Holy Shield. Also adds a damage percent bonus based on the linked skill’s Param3 & Param4 values, plus the stat bonuses from Strength and Dexterity
11		Calculates damage by obtaining the current weapon damage, and then adds the following: Fire percent damage based on the linked skill’s Calc1 field Cold percent damage based on the linked skill’s Calc2 field Lightning percent damage based on the linked skill’s Calc3 field
12		Calculates a skill’s damage based on the status of the Concentration Aura state If the game is in Expansion, then use the linked skill’s Calc1 field to define the bonus damage when using Concentration If the game is in Classic, then use the Concentration skill’s damage percent increase to define the bonus damage when using Concentration
13		Calculates throwing damage, where it adds a damage percent bonus defined by the linked skill’s Calc1 field
14		Calculates the damage of a skill, and uses the linked skill’s Param5 field as an overall damage percent penalty
15		Calculates the total damage by adding a damage percent bonus from the linked skill’s Param1 & Param2 linear increase calculation, the progressive increase from charges from the linked skill, and the boot damage
16		Calculates the total damage by adding a damage percent bonus from the linked skill’s Calc1 value, the progressive increase from charges from the linked skill, and the boot damage
17	ddam calc1 ddam calc2	Calculates the damage of a skill and displays the physical damage and the elemental damage separately “ddam calc1” is used as a percent bonus “ddam calc2” is used as a flat number bonus

18	ddam calc1 ddam calc2	Calculates the damage of a skill, including damage bonuses (Uses function 7) “ddam calc1” is used as a percent bonus “ddam calc2” is used as a flat number bonus
19	ddam calc1 ddam calc2	Calculates the damage of a dual wielding attack. If not dual wielding, then it calculates a normal attack damage “ddam calc1” is used as a percent bonus “ddam calc2” is used as a flat number bonus
20	ddam calc1 ddam calc2	Same as function 19, but does not add elemental damage
21		Calculates the throwing weapon damage with the linked skill’s elemental damage added
22		Calculates the throwing weapon damage for dual wielding throwing weapons and displays them as two values
23		Calculates the damage of a skill and displays the physical damage and elemental damage separately (Similar to function 17)
24		Calculates damage using the equipped weapon damage and the linked skill’s physical and elemental damage (Similar to function 5)
25	ddam calc1 ddam calc2	Same as function 5 with a percentage multiplier to min and max.
26	ddam calc1 ddam calc2	Calculates weapon damage and skill damage as two values.

ddam calc1 & ddam calc2 - Integer calc value used as a possible parameter for the descdam function
p1dmelem (to p3dmelem) - Used for skills that have charge-ups to display the damage on the Character Screen, controls the elemental type for that charge
p1dmmin (to p3dmmin) - Used for skills that have charge-ups to display the damage on the Character Screen, controls the minimum damage for that charge
p1dmmax (to p3dmmax) - Used for skills that have charge-ups to display the damage on the Character Screen, controls the maximum damage for that charge
descatt - Used to display the overall Attack Rating of the skill in the Character Screen

Code	Description
0	null
1	Displays the overall Attack Rating the character’s primary weapon
2	If the character can dual wield two weapons, then display the overall Attack Rating for each weapon
3	Displays the overall Attack Rating for throwing the right-hand weapon
4	Displays the overall Attack Rating for throwing the left-hand weapon
5	Displays the overall Attack Rating for a skill marked with the “finishing” flag

descmissile1 (to descmissile3) - Links a missile from Missiles.txt to be used as a reference value for calculations

descline1 (to descline6) - Uses an ID value to select a description function to format the string value. Displays this text as the current level and next level description lines in the skill tooltip.

Code	Parameters	Description
0 (or empty)		None
13	desctexta desccalca desccalcb	Calculates the Life value of the monster referenced from the “summon” field in the linked skill. Also multiplies this value with [desccalca] as a Life Percent bonus or adds to this value with [desccalcb] as a Life Add bonus. Inserts this calculated Life value into [desctexta] and output that string
31	desctexta desctextb desccalca desccalcb	Performs the calculation using the “AiCurseDivisor” from difficultylevels.txt based on the current game’s difficulty mode: [desccalca] / [AiCurseDivisor] / [desccalcb] If this value is equals to 1, then insert the calculated value into [desctexta] and output that string If this value is greater than or less than 1, then insert the calculated value into [desctextb] and output that string
34	desctexta	Calculates the Damage value of the monster referenced from the “summon” field in the linked skill. Then this function inserts that value into [desctexta] and outputs that string
36	desctexta desctextb desccalca desccalcb	Performs the calculation of a value: [desccalca] / [desccalcb] If this value is equals to 1, then insert the value into [desctexta] and output that string If this value is greater than or less than 1, then insert the value into [desctextb] and output that string
40	desctexta desctextb desccalca	Use [desccalca] as a code to change the color of the string 0 = White (R=255, G=255, B=255) 1 = Red (R=255, G=77, B=77) 2 = Green (R=0, G=255, B=0) 3 = Blue (R=105, G=105, B=255) 4 = Light Gold (R=199, G=179, B=119) 5 = Grey (R=105, G=105, B=105) 6 = Black (R=0, G=0, B=0) 7 = Dark Gold (R=208, G=194, B=125) 8 = Orange (R=255, G=168, B=0) 9 = Yellow (R=255, G=255, B=100) 10 = Dark Green (R=0, G=128, B=0) 11 = Purple (R=174, G=0, B=255) 12 = Medium Green (R=0, G=200, B=0) Inserts [desctextb] into [desctexta] and outputs that string
56		Gets the quantity of the item that is connected to the linked skill and inserts this value into the “scrollbooktext” string and outputs that string

74	desctexta desccalca	Inserts [desccalca] into [desctexta] and outputs that string
75	desctexta desccalca desccalcb	Inserts [desccalca] and [desccalcb] into [desctexta] and outputs that string
76	desctexta desctextb desccalca	Inserts [desctextb] and [desccalca] into [desctexta] and outputs that string
77	desctexta desctextb desccalca desccalcb	Inserts [desctextb], [desccalca], and [desccalcb] into [desctexta] and outputs that string

desctexta1 (to desctexta6) - String value used as the first possible string parameter for the descline function
desctextb1 (to desctextb6) - String value used as the second possible string parameter for the descline function
desccalca1 (todesccalca6) - Integer calculation value used as the first possible numeric parameter for the descline function
desccalcb1 (todesccalcb6) - Integer calculation value used as the second possible numeric parameter for the descline function

dsc2line1 (to dscline5) - Uses an ID value to select a description function to format the string value. Displays this text as a pinned line, after the skill description. (Uses the same function codes as descline1)
dsc2texta1 (to dsc2texta5) - String value used as the first possible string parameter for the dsc2line function
dsc2textb1 (to dsc2textb5) - String value used as the second possible string parameter for the dsc2line function
dsc2calca1 (to dsc2calca5) - Integer Calc value used as the first possible numeric parameter for the dsc2line function
dsc2calcb1 (to dsc2calcb5) - Integer Calc value used as the second possible numeric parameter for the dsc2line function

dsc3line1 (to dsc3line7) - Uses an ID value to select a description function to format the string value. Displays this text as a pinned line at the bottom of the skill tooltip. (Uses the same function codes as descline1)
dsc3texta1 (to dsc3texta7) - String value used as the first possible string parameter for the dsc3line function
dsc3textb1 (to dsc3textb7) - String value used as the second possible string parameter for the dsc3line function
dsc3calca1 (to dsc3calca7) - Integer Calc value used as the first possible numeric parameter for the dsc3line function
dsc3calcb1 (to dsc3calcb7) - Integer Calc value used as the second possible numeric parameter for the dsc3line function

item proc text – String value used as an override format for when the skill appears as a “chance to cast” property on an item. Can be formatted to include descline1 to descline6 in the string using “%s” entries. Leave blank to ignore and use the string format in ItemStatCost.txt.
item proc descline count – Integer value for how many descline entries should be formatted into the “item proc text“ string.

sounds.txt

Overview

This file controls settings for all sounds in the game

The order of each sound defined in this file will convey what ID value it has. This existing order should not be changed.

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

Sound - Defines the unique name ID for the sound, which is how other files can reference the sound

Redirect - Points the sound so the index of another sound in the data file. If this field is not empty, the game will use the redirected sound instead of this sound. This can be used when playing the game in the new graphics mode.
Channel - Declares which channel the sound is initialized in. This can affect how different volume or sound settings handle this sound.
FileName - Defines the file path and name of the sound file to play

IsLocal - Boolean Field. If equals 1, then this sound is considered a localized sound and will change based on the game’s localization setting. If equals 0, then ignore this.
IsMusic - Boolean Field. If equals 1, then the sound is flagged as a music sound, which affects how music related settings handle this sound. If equals 0, then ignore this.
IsAmbientScene - Boolean Field. If equals 1, then the sound is flagged as an ambient scene sound, which affects how the game handles the sound when the player transitions between areas. If equals 0, then ignore this.
IsAmbientEvent - Boolean Field. If equals 1, then the sound is flagged as an ambient event sound, which affects how the game treats the sound when the player transitions between areas. If equals 0, then ignore this.
IsUI - Boolean Field. If equals 1, then the sound is flagged as a UI sound, which affects how UI related settings handle this sound. If equals 0, then ignore this.

Volume Min - Controls the minimum volume of the sound. Uses a range of 0 to 255.
Volume Max - Controls the maximum volume of the sound. If both “Volume Min” and “Volume Max” fields differ in value, then the sound will randomly select a volume value in between these values when it is played. Uses a range of 0 to 255.

Pitch Min - Controls the minimum pitch percentage of the sound.
Pitch Max - Controls the maximum pitch percentage of the sound. If both “Pitch Min” and “Pitch Max” fields differ in value, then the sound will randomly select a pitch value in between these values when it is played.

Group Size - Defines a sound Group by declaring a size value. When the sound has this value greater than 0, then this sound is declared as the group's base sound. Any link to use a sound should use the base sound, to signify that the game should use this group of sounds. This field's value controls the number of sounds indexed after base sound that should be added to the group. For example, if the sound has a "Group Size" value equal to 5, then this sound is declared as the group's base sound, and the next 4 sounds indexed after this base sound will be added to the group.
Group Weight - Controls the chance to pick the sound when it is part of a group with other sounds. If all sounds in the group do not have a "Group Weight" value, then the group sounds will play in historical order. This value controls a weighted random chance, meaning that all related sounds have their weights added together for a total chance and each sound's weight value is rolled against that total value to determine if the sound is successfully picked. The higher this value, the more likely the sound will be picked. This is only used when the sound is part of a group (See "Group Size").

Loop - Boolean Field. If equals 1, then the sound will replay itself after it finishes playing. If equals 0, then the sound will only play once.
Fade In - Controls how long to gradually increase the sound's volume starting from 0 when the sound starts playing. Measured in audio game ticks, where 1 game frame is 40 audio ticks, and the game runs at 25 frames per second.
Fade Out - Controls how long to gradually decrease the sound's volume to 0 when the sound stops playing. Measured in audio game ticks, where 1 game frame is 40 audio ticks, and the game runs at 25 frames per second.

Defer Inst - Boolean Field. If equals 1, then when a duplicate instance of this sound plays the game will stop that request. If equals 0, then ignore this.
Stop Inst - Boolean Field. If equals 1, then when a duplicate instance of this sound plays the previous instance of the sound will stop and the new instance of the sound will play. If equals 0, then ignore this.

Duration - Controls the length of time to play the sound. When the sound has been playing for this length of time, then the sound will stop. If this equals 0, then ignore this functionality.
Compound - Controls the game tick time limit for when a sound can join in playing based on the previous sound played in the Group. If equals 0, then the sound will not be compounded.

Falloff - Defines the range of falloff for hearing the sound, based on distance. Uses a code to determine the range value presets.

Code	Description
0	Short - falloff range is 60 to 400 pixels
1	Medium - falloff range is 60 to 700 pixels
2	Large - falloff range is 200 to 1000 pixels
3	Ambient - falloff range is 400 to 1500 pixels
4	Voice - falloff range is 2000 pixels (no falloff)

LFEMix - Controls the percentage (out of 100) of the sound's Low-Frequency Effects channel.
3dSpread - Controls the 3D spread angle of the sound. This only works if the sound is considered a 3D sound (See "Is2D").

Priority - Controls which if the sound should play before other sounds when too many sounds are playing at once. This value is compared to the priority value of other sounds, and the sound that has the higher priority will play first. Sounds belonging to the player will get an increased priority value of 80.

Stream - Boolean Field. If equals 1, then the sound will be file streamed into the game when called to play. If equals 0, then the entire sound will be loaded into the game before playing.
Is2D - Boolean Field. If equals 1, then the sound is considered a 2D sound and will not have 3D spread settings. If equals 0, then the sound is considered a 3D sound and will use the 3D spread settings.
Tracking - Boolean Field. If equals 1, then the sound will track a unit and will update its position to follow that unit. If equals 0, then the sound will not move and will be stationary.
Solo - Boolean Field. If equals 1, then reduce the volume of other sounds while this sound is playing. If equals 0, then ignore this.
Music Vol - Boolean Field. If equals 1, then the sound's volume will be affected by the music volume in the game options menu. If equals 0, then ignore this.
Block 1 (to Block 3) - Defines an offset time value in the sound. If this sound is used in a Sound Environment (See SoundEnviron.txt) then these fields control when to periodically update the current song sound to an offset. If this sound is not used in a Sound Environment and if only "Block 1" is used and the "Loop" field is enabled, then use this block value as the time in the sound when to start looping. If this equals -1, then the field is ignored.
HDOptOut - Boolean Field. If equals 1, then the sound will not play in the new graphics mode. If equals 0, then the sound will play in the new graphics mode.
Delay - Adds a delay to the starting tick of the sound when the sound starts playing. Measured in audio game ticks, where 1 game frame is 40 audio ticks, and the game runs at 25 frames per second.

SoundEnviron.txt

Overview

This file controls the music and ambient sounds that are played while the player is in the area level

The order of each Sound Environment defined in this file will convey what ID value it has

This file relies on sounds from sounds.txt

This file is used by levels.txt

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

Handle - A reference field to define the name of the Sound Environment

Song - Play this sound as the background music while the player is in an area level. Points to a "Sound" value in the sounds.txt file.

Day Ambience - Play this sound as an ambient sound while it is currently daytime in the game. Points to a “Sound” value in the sounds.txt file.
HD Day Ambience - Play this sound as an ambient sound while it is currently daytime in the game while playing in the new graphics mode. Points to a “Sound” value in the sounds.txt file.
Night Ambience - Play this sound as an ambient sound while it is currently nighttime in the game. Points to a “Sound” value in the sounds.txt file.
HD Night Ambience - Play this sound as an ambient sound while it is currently nighttime in the game while playing in the new graphics mode. Points to a “Sound” value in the sounds.txt file.

Day Event - Play this sound at a random range and variance in the background when it is currently daytime in the game. Points to a “Sound” value in the sounds.txt file.
HD Day Event - Play this sound at a random range and variance in the background when it is currently daytime in the game while playing in the new graphics mode. Points to a “Sound” value in the sounds.txt file.
Night Event - Play this sound at a random range and variance in the background when it is currently nighttime in the game. Points to a “Sound” value in the sounds.txt file.
HD Night Event - Play this sound at a random range and variance in the background when it is currently nighttime in the game while playing in the new graphics mode. Points to a “Sound” value in the sounds.txt file.
Event Delay - Controls the baseline number of frames to wait before playing the “Day Event” or “Night Event” sound, depending on the time of day. This only applies when the game is being played in SD mode. This value is used in the following calculation to get a random time to play the next event sound: [“Event Delay”] - [“Event Delay”] / 3 + RANDOM(0, ([“Event Delay”] / 3 * 2 + 1))
HD Event Delay - Controls the baseline number of frames to wait before playing the “Day Event” or “Night Event” sound, depending on the time of day. This only applies when the game is being played in the new graphics mode. This value is used in the following calculation to get a random time to play the next event sound: [“Event Delay”] - [“Event Delay”] / 3 + RANDOM(0, ([“Event Delay”] / 3 * 2 + 1))

Indoors - Boolean Field. If equals 1 then, if the current sound being played in the area level with this Sound Environment is “event_thunder_1”, then the sound will be obstructed. If equals 0, then ignore this.
Material 1 & Material 2 - Controls the material of the Sound Environment, which affects which footstep sounds are played. Uses a code to define a specific material.

Code	Description
0	None
1	Dirt
2	Indoor Stone
3	Outdoor Stone
4	Sand
5	Snow
6	Wood

HD Material 1 & HD Material 2 - Controls the material of the Sound Environment, which affects which footstep sounds are played. Uses a code to define a specific material. This only applies when the game is being played in the new graphics mode. See “Material 1 & Material 2” for the code descriptions.

The following are sound reverberation settings for special effects sounds
SFX EAX Environ - Determines an environment preset for default sound reverberation settings.

Code	Description
0	Generic
1	Padded Cell
2	Room
3	Bathroom
4	Livingroom
5	Stone Room
6	Auditorium
7	Concert Hall
8	Cave
9	Arena
10	Hanger
11	Carpeted Hallway
12	Hallway
13	Stone Corridor
14	Alley
15	Forest
16	City
17	Mountains
18	Quarry
19	Plain
20	Parking Lot
21	Sewer Pipe
22	Underwater
23	Drugged
24	Dizzy
25	Psychotic
26	Programmer Test (A long distant echo)

SFX EAX Room Vol - Room effect level at mid frequencies.
SFX EAX Room HF - Relative room effect level at high frequencies.
SFX EAX Decay Time - Reverberation decay time at mid frequencies.
SFX EAX Decay HF - High-frequency to mid-frequency decay time ratio.
SFX EAX Reflect - Early reflections level relative to room effect.
SFX EAX Reflect Delay - Initial reflection delay time.
SFX EAX Reverb - Late reverberation level relative to room effect.
SFX EAX Rev Delay - Late reverberation delay time relative to initial reflection.

The following are sound reverberation settings for Voice sounds.
VOX EAX Environ - Determines an environment preset for default sound reverberation settings.

Code	Description
0	Generic
1	Padded Cell
2	Room
3	Bathroom
4	Livingroom
5	Stone Room
6	Auditorium
7	Concert Hall
8	Cave
9	Arena
10	Hanger
11	Carpeted Hallway
12	Hallway
13	Stone Corridor
14	Alley
15	Forest
16	City
17	Mountains
18	Quarry
19	Plain
20	Parking Lot
21	Sewer Pipe
22	Underwater
23	Drugged
24	Dizzy
25	Psychotic
26	Programmer Test (A long distant echo)

VOX EAX Room Vol - Room effect level at mid frequencies.
VOX EAX Room HF - Relative room effect level at high frequencies.
VOX EAX Decay Time - Reverberation decay time at mid frequencies.
VOX EAX Decay HF - High-frequency to mid-frequency decay time ratio.
VOX EAX Reflect - Early reflections level relative to room effect.
VOX EAX Reflect Delay - Initial reflection delay time.
VOX EAX Reverb - Late reverberation level relative to room effect.
VOX EAX Rev Delay - Late reverberation delay time relative to initial reflection.

InheritEnvironment – Boolean field. If equals 1, then this sound environment will inherit certain values from the existing environment and overwrite other values with its own.

Overwritten values
Song
Day Ambience
HD Day Ambience
Night Ambience
HD Night Ambience
Day Event
HD Day Event
Night Event
HD Night Event
Event Delay
HD Event Delay

states.txt

Overview

This file defines the different states used by the game and controls how they function. States are basically passive behaviors applied to units that can apply various effects.

This file is used by the following data files: cubemain.txt, MonProp.txt, Overlay.txt, Runes.txt, Sets.txt, SetItems.txt, skills.txt, UniqueItems.txt

Any column field name starting with “*” is considered a comment field and is not used by the game

Data Fields

state - Defines the unique name ID for the state
group - Assigns the state to a group ID value. This means that only 1 state with that group ID can be active at any time on a unit. If this value is empty, then ignore this.

remhit - Boolean field. If equals 1, then this state will be removed when the unit is hit. If equals 0, then ignore this.
nosend - Boolean field. If equals 1, then this state change will not be sent to the client. If equals 0, then ignore this.

transform - Boolean field. If equals 1, then this state will be flagged to change the unit's appearance and reset its animations when it is applied. If equals 0, then ignore this.
aura - Boolean field. If equals 1, then this state will be treated as an aura. If equals 0, then ignore this.
curable - Boolean field. If equals 1, then this state can be cured (This can be checked by NPC healing or the Paladin Cleansing skill). If equals 0, then ignore this.
curse - Boolean field. If equals 1, then this state will be flagged as a curse. If equals 0, then ignore this.
active - Boolean field. If equals 1, then the state will be classified as an active state which enables the "cltactivefunc" and "srvactivefunc" fields. If equals 0, then ignore this.
restrict - Boolean field. If equals 1, then this state will restrict the usage of certain skills (This connects with the "restrict" field from the skills.txt file). If equals 0, then ignore this.
disguise - Boolean field. If equals 1, then this state will be flagged as a disguise, meaning that the unit's appearance is changed, which can affect how the animations are treated when being used. If equals 0, then ignored this.

attblue - Boolean field. If equals 1, then the state will make the related Attack Rating value in the character screen be colored blue. If equals 0, then ignore this.
damblue - Boolean field. If equals 1, then the state will make related Damage value in the character screen be colored blue. If equals 0, then ignore this.
armblue - Boolean field. If equals 1, then the state will make Defense value (Armor) in the character screen be colored blue. If equals 0, then ignore this.
rfblue - Boolean field. If equals 1, then the state will make Fire Resistance value in the character screen be colored blue. If equals 0, then ignore this.
rlblue - Boolean field. If equals 1, then the state will make Lightning Resistance value in the character screen be colored blue. If equals 0, then ignore this.
rcblue - Boolean field. If equals 1, then the state will make Cold Resistance value in the character screen be colored blue. If equals 0, then ignore this.
stambarblue - Boolean field. If equals 1, then the state will make the Stamina Bar UI in the HUD be colored blue. If equals 0, then ignore this.
rpblue - Boolean field. If equals 1, then the state will make Poison Resistance value in the character screen be colored blue. If equals 0, then ignore this.

attred - Boolean field. If equals 1, then the state will make the related Attack Rating value in the character screen be colored red. If equals 0, then ignore this.
damred - Boolean field. If equals 1, then the state will make related Damage value in the character screen be colored red. If equals 0, then ignore this.
armred - Boolean field. If equals 1, then the state will make Defense value (Armor) in the character screen be colored red. If equals 0, then ignore this.
rfred - Boolean field. If equals 1, then the state will make Fire Resistance value in the character screen be colored red. If equals 0, then ignore this.
rlred - Boolean field. If equals 1, then the state will make Lightning Resistance value in the character screen be colored red. If equals 0, then ignore this.
rcred - Boolean field. If equals 1, then the state will make Cold Resistance value in the character screen be colored red. If equals 0, then ignore this.
rpred - Boolean field. If equals 1, then the state will make Poison Resistance value in the character screen be colored red. If equals 0, then ignore this.

exp - Boolean field. If equals 1, then a unit with this state will give exp when killed or will gain exp when killing another unit. If equals 0, then ignore this.
plrstaydeath - Boolean field. If equals 1, then the state will persist on the player after that player is killed. If equals 0, then ignore this. state stays after death
monstaydeath - Boolean field. If equals 1, then the state will persist on the monster (non-boss) after that monster is killed. If equals 0, then ignore this.
bossstaydeath - Boolean field. If equals 1, then the state will persist on the boss after that boss is killed. If equals 0, then ignore this.
hide - Boolean field. If equals 1, then the state will hide the unit when dead (corpse and death animations will not be drawn). If equals 0, then ignore this.
hidedead - Boolean field. If equals 1, then the state will be used to destroy units with invisible corpses. If equals 0, then ignore this.
shatter - Boolean field. If equals 1, then the state causes ice shatter missiles to create when the unit dies. If equals 0, then ignore this.
udead - Boolean field. If equals 1, then the state flags the unit as a used dead corpse and the unit cannot be targeted for corpse skills. If equals 0, then ignore this.

life - Boolean field. If equals 1, then this state will cancel out the monster's normal life regeneration. If equals 0, then ignore this.
green - Boolean field. If equals 1, then the state overrides the color changes the unit and the unit will be colored green. If equals 0, then ignore this.
pgsv - Boolean field. If equals 1, then the state is flagged as part of a progressive skill which relates to charge-up skill functionalities. If equals 0, then ignore this.
nooverlays - Boolean field. If equals 1, then the standard way for states to add overlays will be disabled. If equals 0, then ignore this.
noclear - Boolean field. If equals 1, then when this state is applied on the unit, it will not clear stats that have this state from the state's previous application. If equals 0, then ignore this.
bossinv - Boolean field. If equals 1, then the unit with this state will use the state's source unit's (in this case, the unit's boss) inventory for generating the unit's equipped item graphics. If equals 0, then ignore this.
meleeonly - Boolean field. If equals 1, then the state will make the all the unit's attack become melee attacks. If equals 0, then ignore this.
notondead - Boolean field. If equals 1, then the state will not play its On function (function that happens when the state is applied) if the unit is dead. If equals 0, then ignore this.

overlay1 (to overlay4) - Controls which overlay to use for normally displaying the state (Uses the "overlay" field from the Overlay.txt file). The usage depends on the specific state defined and/or the function using the state. Typically, states use "overlay1" for the Front overlay and "overlay2" for the Back overlay. Other cases can have states use each overlay field as the Front Start, Front End, Back Start, and Back End, respectively.
pgsvoverlay - Controls which overlay to use when the state has progressive charges on the unit, such as for the charge-up stat when using Assassin Martial Arts charge-up skills (Uses the "overlay" field from the Overlay.txt file)
castoverlay - Controls which overlay to use when the state is initially applied on the unit (Uses the "overlay" field from the Overlay.txt file)
removerlay - Controls which overlay to use when the state is removed from the unit (Uses the "overlay" field from the Overlay.txt file)

stat - Controls the stat associated with the stat. This is also used when determining how to add the progressive overlay (Uses the "Stat" field from ItemStatCost.txt)
setfunc - Controls the client side set functions for when the state is initially applied on the unit

Code	Parameters	Description
0		Do nothing
1	stat	Creates the overlay used for a progressive state. Can only be used if the "pgsv" flag is enabled and the "pgsvoverlay" field has a value.
2		Changes the area level's room lighting based on a skill's "auralencalc" field from the skills.txt file. Gets the skill by looking at the "modifierlist_skill" stat defined in the ItemStatsCost.txt file
3		Updates a skill's level. Gets the skill by looking at the "modifierlist_skill" stat defined in the ItemStatsCost.txt file.
4		Sets the source unit for the state. Gets the "source_unit_type" and "source_unit_id" stats defined in the ItemStatsCost.txt file.
5		Changes the monster's class type to another monster's class type. Gets the "shortparam1" stat defined in the ItemStatsCost.txt file and uses stat's parameter to get the class type that the unit should change to. Only works for monster units. Has a special case where if the class the monster changed from was the "baalthrone" monster (defined in monstats.txt), then also set the path of the monster to move a direction.
6		Gets the skill by looking at the "modifierlist_skill" stat defined in the ItemStatsCost.txt file and then creates the overlays defined in the "castoverlay" and "overlay#" fields

7		Plays a sound from the “prgsound” field of a skill from the skills.txt file. Gets the skill by looking at the “modifierlist_skill” stat defined in the ItemStatsCost.txt file.
8		Gets the skill by looking at the “modifierlist_skill” stat defined in the ItemStatsCost.txt file and then creates the overlays defined in the “overlay1”, “overlay2” and “overlay3” fields
9		Calls the updates passive skills function which updates the values of any skill with a “passivestate” field defined in the skills.txt file
10	skill	Creates the missile defined in the “skill” parameter’s “cltmissile” field and hides the targeted unit
11		Hides the unit, by disabling the drawing of its visuals
12		Hides the unit, by disabling the drawing of its visuals and shadows
13		Initializes the particle for attaching to the unit by getting the offset of the source unit’s Special component
14		Tells the unit to use the “SKILL1” command and resets its direction
15		Sets the monster mode to neutral and sets its flag to a pet
16		Sets up the overlays for a charge up skill. Gets the skill by looking at the “modifierlist_skill” stat defined in the ItemStatsCost.txt file. Applies all the overlays defined the “overlay#” fields, based on the number of skill charges on the unit.
17	missile	Creates the missile defined in the “missile” parameter
18	missile	Creates blood on the targeted unit and creates the missile defined in the “missile” parameter
19		Sets the global skill cooldown to 0

remfunc - Controls the client side remove functions for when the state is removed from the unit

Code	Description
0	Do nothing
1	Removes the “pgsvoverlay” overlay. This function relies on the “pgsv” being enabled.
2	Removes the state’s source unit
3	Removes the overlays defined in the “castoverlay” and all of the “overlay#” fields
4	Removes the “cltprgsound” from the related skill. To get the skill, this looks at the “modifierlist_skill” stat defined in the ItemStatsCost.txt file.
5	Removes the overlays defined in the “castoverlay”, “overlay1”, “overlay2”, and “overlay3” fields
6	Calls the updates passive skills function which updates the values of any skill with a “passivestate” field defined in the skills.txt file
7	Checks that the related unit is a monster and that the skill used is the “Nest” skill defined in the skills.txt file. If true, then it removes the related unit’s collision pattern.
8	Hides the unit, by disabling the drawing of its visuals and shadows
9	Removes particles attached to the unit or the position of the unit
10	Gets the related unit’s position and creates a “monstercorpseexplode” and “pain worm appear” missile defined from the Missiles.txt file
11	Removes the overlays defined in all of the “overlay#” fields
12	Sets the global skill cooldown to 0

missile - Used as a possible parameter for the “setfunc” field (Uses the “Missile” field from Missiles.txt)
skill - Used as a possible parameter for the “setfunc” field (Uses the “skill” field from skills.txt)

itemtype - Defines a potential Item Type (see ItemTypes.txt) that can be affected by the state’s color change
itemtrans - Controls the color change of the item when the unit has this state (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

colorpri - Defines the priority of the state’s color change, when compared to other current sates on the unit. The current state that has the highest color priority on the unit will be used and other state colors will be ignored. If multiple current states share the same color priority value, then the game will choose the state with the lower ID value (based on where in the list of states in the data file that the state is defined)
colorshift - Controls which index of the color shift palette to use.

ID	Description
0 (or empty)	Do nothing
1	First Hue Rotation
25	First Hue Rotation and Darken

49	First Hue Rotation and Lighten
73	Color to Grey
74	Color to Black
75	First No Red Rotation
100	Color to Red
101	Color to Orange
102	Color to Yellow
103	Color to Grass
104	Color to Green (There is a special case to not turn the player unit green)
105	Color to Teal
106	Color to Aqua
107	Color to Light Blue
108	Color to Blue
109	Color to Purple
110	Color to Magenta
111	Color to Some Funky Red
112	Color to RGB Red
113	Color to RGB Green
114	Color to RGB Blue

light-r - Controls the state's change of the red color value of the Light radius (Uses a value from 0 to 255)
light-g - Controls the state's change of the green color value of the Light radius (Uses a value from 0 to 255)
light-b - Controls the state's change of the blue color value of the Light radius (Uses a value from 0 to 255)

onsound - Plays a sound when the state is initially applied to the unit. Links to a "Sound" from the sounds.txt file.
offsound - Plays a sound when the state is removed from the unit. Links to a "Sound" from the sounds.txt file.

gfxtype - Controls the how to handle the unit graphics transformation based on the unit type (This relies on the "disguise" field being enabled). If equals 1, then use this on a monster type unit. If equals 2, then use this on a player type unit. Otherwise, ignore this.
gfxclass - Control's the unit class used for handling the unit graphics transformation. This field relies on what unit type was used in the "gfxtype" field. If "gfxtype" equals 1 for monster type units, then this field will rely on the "hcIdx" field from the monstats.txt data file. If "gfxtype" equals 2, then this field will use the character class numeric ID.

ID	Description
0	Amazon
1	Sorceress
2	Necromancer
3	Paladin
4	Barbarian
5	Druid
6	Assassin

cltevent - Controls the event to check on the client side to determine when to use the function defined in the "clteventfunc" field (Uses an event defined in the Events.txt file)
clteventfunc - Controls the client Unit event function that is called when the event is determined in the "cltevent" field. These functions are equal to the funuctions used

ID	Description
0	Do nothing
1	Sorceress Apply Chilling Armor • Requires on the "hitbymissile" event defined in the "cltevent" field • Uses the related skill with this state and it's related missile fields to fire a missile at a target

cltactivefunc - Controls the Client Do function that is called every frame while the state is active (see the "cltdofunc" field in skills.txt). This relies on the "active" field being enabled.
srvactivefunc - Controls the Server Do function that is called every frame while the state is active (see the "srvdofunc" field in skills.txt). This relies on the "active" field being enabled.

canstack - Boolean Field. If equals 1, then this state can stack with duplicate forms of itself (This is only usable with the "poison" state). If equals 0, then ignore this.

sunderfull - Boolean Field. If equals 1, then this state will reapply any negative resistance stats at full potential when calculating pierce immunity if the immunity was broken. If equals 0, then reapply at the normal reduced efficiency (currently 1/5).

sunder-res-reduce – Boolean Field. If equals 1, then this state will apply pierce resistance at reduced effectiveness (currently 1/5) when calculating pierce resistance if an immunity was broken. If equals 0, then apply pierce resistance at normal effectiveness.

SuperUniques.txt

Overview

This file defines the Super Unique monsters and their properties. Super Unique monsters are considered the special boss monsters that have static encounters in the game.

This file uses the following data files: monsounds.txt, monstats.txt, monumod.txt, TreasureClassEx.txt

Data Fields

Superunique - Defines the unique name ID for the Super Unique monster
Name - Uses a string for the Super Unique monster's name
Class - Defines the baseline monster type for the Super Unique monster, which this monster will use for default values. This uses the "Id" field from the monstats.txt file.
hcIdx - Defines the unique numeric ID for the Super Unique monster. The existing IDs are hardcoded for specific scripts with the specified Super Unique monsters.

MonSound - Defines what set of sounds to use for the Super Unique monster. Uses the "Id" field from the monsounds.txt file. If this field is empty, then the Super Unique monster will default to using the monster class sounds.

Mod1 (to Mod3) - Controls which monster modifier to assign to the Super Unique monster. Uses the "id" field from the monumod.txt file (See that file for details on the available modifiers).

MinGrp - Controls the min amount of Minion monsters that will spawn with the Super Unique monster.
MaxGrp - Controls the max amount of Minion monsters that will spawn with the Super Unique monster. This value must be equal to or higher than "MinGrp". If this value is greater than "MinGrp" then a random number will be chosen between the "MinGrp" and "MaxGrp" values.

AutoPos - Boolean Field. If equals 1, then the Super Unique monster will randomly spawn within a radius of its designated position. If equals 0, then the Super Unique monster will spawn at exact coordinates of its designated position.
Stacks - Boolean Field. If equals 1, then this Super Unique monster can spawn more than once in the same game. If equals 0, then this Super Unique monster can only spawn once in the same game.
Replaceable - Boolean Field. If equals 1, then the room where the Super Unique monster spawns in can be replaced during the creation of a level preset. If equals 0, then the room cannot be replaced and will remain static.

Utrans & Utrans(N) & UTrans(H) - Modifies the color transform for the unique monster respectively in Normal, Nightmare, or Hell difficulty. If this value is greater than or equal to 30, then the value will default to 2, which is the monster's default color palette shift. If the value is 0 or is empty, then a random value will be chosen.

TC & TC(N) & TC(H) - Controls the Treasure Class to use when the Super Unique monster is killed respectively in Normal, Nightmare, or Hell difficulty. This linked to the "Treasure Class" ID from the TreasureClassEx.txt file.
TC Desecrated & TC(N) Desecrated & TC(H) Desecrated - Controls the Treasure Class to use when the Super Unique monster is desecrated (Terrorized) and killed respectively in Normal, Nightmare, or Hell difficulty. This linked to the "Treasure Class" ID from the TreasureClassEx.txt file.

TreasureClassEx.txt

Overview

This file controls the Treasure Class linked to a monster drop. Treasure Classes are groups of item types and their chances of dropping from a monster.

This is used by the following data files: monstats.txt, SuperUniques.txt

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

Treasure Class - Defines the unique Treasure Class ID, that is referenced in other files.

group - Assigns the Treasure Class to a group ID value, which will connect this Treasure Class with other Treasure Classes, as a potential Treasure Class to use for an itemdrop. When determining which Treasure Class to use for an item drop, the game will iterate through all Treasure Classes that share the same group. This field works with the "level" field to determine an ideal Treasure Class to use for the monster drop. Treasure Classes that share the same group should be in contiguous order.
level - Defines the level of a Treasure Class. Monsters who have a Treasure Class will pick the Treasure Class that a level value that is less than or equal to the monster's level. This is ignored for Boss monsters unless that Boss monster is desecrated.

Picks - Controls how to handle the calculations for item drops. If this value is positive, then this value will control how many item drop chances will be rolled for the Treasure Class using the "Prob#" fields as probability values. If this value is negative, then this value functions as the total guaranteed quantity of item drops from the Treasure Class, and each "Prob#" field now defines the quantity of items generated from its related "Item#" field. If this field is empty, then default to a value of 1.

Unique - Modifies the item ratio drop for a Unique Quality item. A higher value means a better chance of being chosen. (See itemratio.txt for an explanation for how the Item Quality is chosen)
Set - Modifies the item ratio drop for a Set Quality item. A higher value means a better chance of being chosen. (See itemratio.txt for an explanation for how the Item Quality is chosen)
Rare - Modifies the item ratio drop for a Rare Quality item. A higher value means a better chance of being chosen. (See itemratio.txt for an explanation for how the Item Quality is chosen)
Magic - Modifies the item ratio drop for a Magic Quality item. A higher value means a better chance of being chosen. (See itemratio.txt for an explanation for how the Item Quality is chosen)

NoDrop - Controls the probability of no item dropping by the Treasure Class. The higher this value, then the more likely no item will drop from the monster. This can be automatically be affected by the number of players currently in the game.

Item1 (to Item10) - Defines a potential Item Type (see ItemTypes.txt) or other Treasure Class that can drop from this Treasure Class. Linking another Treasure Class in this field means that there is a chance to use that Treasure Class group of items which the game will then calculate a selection from that Treasure Class, and so on.
Prob1 (to Prob10) - The individual probability for each related “Item#” drop. The higher this value, then the more likely the “Item#” field will be chosen. The chance a drop is picked is calculated by summing all “Prob#” field values and the “NoDrop” value for a total denominator value, and then having each “Prob#” value and the “NoDrop” value rolling their chance out of the total denominator value for a drop.

firstLadderSeason - Integer field. If this has a value, it will only roll in ladder games starting at the season specified (inclusive). If blank or 0 then it will roll in ladder and non-ladder.
lastLadderSeason - Integer field. The last ladder season the treasure class is ladder-only (inclusive). Must be used in conjunction with firstLadderSeason.
noAlwaysDrop - Boolean field. If equals 1, then this treasure class will roll normally when being forced to always drop, like with the Find Item skill. If 0 then this treasure class will always drop an item when forced to.

UniqueAppellation.txt

Overview

This file controls the list of strings that are randomly selected to be used as an extra suffix when generating unique monster names

The game has a 50% chance to randomly use Unique Appellation when generating the Unique monster name.
- If a Unique Appellation is not added, then the game will generate a unique monster name using the string called Monster1Format (ID: 1721)
- If a Unique Appellation is added, then the game will generate a unique monster name using the string called Monster2Format (ID: 1722)

Data Fields

Name - A string key, which is used as a potential selection for generating a unique monster’s name

UniqueItems.txt

Overview

This file defines each Unique item and controls their item modifiers.

The row order of items should not be changed because it defines their ID value.

Any column field name starting with “*” is considered a comment field and is not used by the game.

Data Fields

index - Points to a string key value to use as the Unique item’s name
version - Defines which game version to create this item (<100 = Classic mode | 100 = Expansion mode)
enabled - Boolean Field. If equals 1, then this item can be rolled as a choice when randomly dropping a unique. If equals 0, then this item cannot be dropped randomly, but can still be drop explicitly from a treasure class.
firstLadderSeason - Integer field. The first ladder season the unique item can be Dropped or created on (inclusive). If blank or 0 then it is available in non-ladder.
lastLadderSeason - Integer field. The last ladder season the unique item is ladder-only (inclusive). Must be used in conjunction with firstLadderSeason.
rarity - Modifies the chances that this Unique item will spawn compared to the other Unique items. This value acts as a numerator and a denominator. Each “rarity” value gets summed together to give a total denominator, used for the random roll for the item. For example, if there are 3 possible Unique items, and their “rarity” values are 3, 5, 7, then their chances to be chosen are 3/15, 5/15, and 7/15 respectively. (The minimum “rarity” value equals 1) (Only works for games in Expansion mode)
nolimit - Boolean Field. Requires the “quest” field from the misc.txt file to be enabled. If equals 1, then this item can be created and will automatically be identified. If equals 0, then ignore this.

lvl - The item level for the item, which controls what object or monster needs to be in order to drop this item
lvl req - The minimum character level required to equip the item

code - Defines the baseline item code to use for this Unique item (must match the “code” field value from weapons.txt, armor.txt, or misc.txt)
carry1 - Boolean Field. If equals 1, then players can only carry one of these items in their inventory. If equals 0, then ignore this.

cost mult - Multiplicative modifier for the Unique item’s buy, sell, and repair costs
cost add - Flat integer modification to the Unique item’s buy, sell, and repair costs. This is added after the “cost mult” has modified the costs.

chrtransform - Controls the color change of the item when equipped on a character or dropped on the ground. If empty, then the item will have the default item color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black

lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

invtransform - Controls the color change of the item in the inventory UI. If empty, then the item will have the default item color. (Uses Color Codes from the reference file colors.txt)

Code	Color
	No color change
whit	White
lgry	Light Grey
dgry	Dark Grey
blac	Black
lblu	Light Blue
dblu	Dark Blue
cblu	Crystal Blue
lred	Light Red
dred	Dark Red
cred	Crystal Red
lgrn	Light Green
dgrn	Dark Green
cgrn	Crystal Green
lyel	Light Yellow
dyel	Dark Yellow
lgld	Light Gold
dgld	Dark Gold
lpur	Light Purple
dpur	Dark Purple
oran	Orange
bwht	Bright White

invfile - An override for the “invfile” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Unique item will use what was defined by the baseline item from the “item” field.
flippyfile - An override for the “flippyfile” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Unique item will use what was defined by the baseline item from the “item” field.

dropsound - An override for the “dropsound” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Unique item will use what was defined by the baseline item from the “item” field.
dropsfxframe - An override for the “dropsfxframe” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Unique item will use what was defined by the baseline item from the “item” field.
usesound - An override for the “usesound” field from the weapon.txt, armor.txt, or misc.txt files. By default, the Unique item will use what was defined by the baseline item from the “item” field.

prop1 (to prop12) - Controls the item properties for the Unique item (Uses the “code” field from Properties.txt)
par1 (to par12) - The stat’s “parameter” value associated with the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
min1 (to min12) - The stat’s “min” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)
max1 (to max12) - The stat’s “max” value to assign to the related property (prop#). Usage depends on the property function (See the “func” field on Properties.txt)

diablocloneweight - The amount of weight added to the diablo clone progress when this item is sold. When offline, selling this item will instead immediately spawn diablo clone.

UniquePrefix.txt

Overview

This file controls the list of strings that are randomly selected to be used as the prefix when generating unique monster names

This is always added to every unique monster name

Data Fields

Name - A string key, which is used as a potential selection for generating a unique monster’s name

UniqueSuffix.txt

Overview

This file controls the list of strings that are randomly selected to be used as the suffix when generating unique monster names

This is always added to every unique monster name

Data Fields

Name - A string key, which is used as a potential selection for generating a unique monster's name

weapons.txt

Overview

This file controls the functionalities for weapons type items

This file is loaded together with other similar files in the following order: weapons.txt, armor.txt, misc.txt
These combined files form the items structure. Technically these files share the same fields, but some fields are exclusive for specific item types, so they are not displayed in the data files that do not need them.

Any column field name starting with "*" is considered a comment field and is not used by the game

Data Fields

name - This is a reference field to define the item
version - Defines which game version to create this item (0 = Classic mode | 100 = Expansion mode)
compactsave - Boolean Field. If equals 1, then only the item's base stats will be stored in the character save, but not any modifiers or additional stats. If equals 0, then all of the items stats will be saved.
rarity - Determines the chance that the item will randomly spawn (1/#). The higher the value then the rarer the item will be. This field depends on the "spawnable" field being enabled, the "quest" field being disabled, and the item level being less than or equal to the area level. This value is also affected by the relative Act number that the item is dropping in, where the higher the Act number, then the more common the item will drop.
spawnable - Boolean Field. If equals 1, then this item can be randomly spawned. If equals 0, then this item will never randomly spawn.

speed - If the item type is an armor, then this will affect the Walk/Run Speed reduction when wearing the armor. If the item type is a weapon, then this will affect the Attack Speed reduction when wearing the weapon.
reqstr - Defines the amount of the Strength attribute needed to use the item
reqdex - Defines the amount of the Dexterity attribute needed to use the item

durability - Defines the base durability amount that the item will spawn with.
nodurability - Boolean Field. If equals 1, then the item will not have durability. If equals 0, then the item will have durability.

level - Controls the base item level. This is used for determining when the item is allowed to drop, such as making sure that the item level is not greater than the monster's level or the area level.
ShowLevel - Boolean Field. If equals 1, then display the item level next to the item name. If equals 0, then ignore this.
levelreq - Controls the player level requirement for being able to use the item

cost - Defines the base gold cost of the item when being sold by an NPC. This can be affected by item modifiers and the rarity of the item.
gamble cost - Defines the gambling gold cost of the item on the Gambling UI

code - Defines a unique 3 letter/number code for the item. This is used as an identifier to reference the item.
namestr - String Key that is used for the base item name

magic lvl - Defines the magic level of the item, which can affect how magical item modifiers that can appear on the item (See automagic.txt)
auto prefix - Automatically picks an item affix name from a designated "group" value from the automagic.txt file, instead of using random prefixes. This is only used when the item is Magical quality.

alternategfx - Uses a unique 3 letter/number code similar to the defined "code" fields to determine what in-game graphics to display on the player character when the item is equipped

normcode - Links to a "code" field to determine the normal version of the item
ubercode - Links to a "code" field to determine the Exceptional version of the item
ultracode - Links to a "code" field to determine the Elite version of the item

component - Determines the layer of player animation when the item is equipped. This uses a code referenced from the Composit.txt file.

Code	Description
0	Head
1	Torso
2	Legs

3	Right Arm
4	Left Arm
5	Right Hand
6	Left Hand
7	Shield
8	Special 1
9	Special 2
10	Special 3
11	Special 4
12	Special 5
13	Special 6
14	Special 7
15	Special 8
16	Do not display anything

invwidth & **invheight -** Defines the width and height of grid cells that the item occupies in the player inventory

hasinv - Boolean Field. If equals 1, then the item will have its own inventory allowing for the capability to socket gems, runes, or jewels. If equals 0, then the item cannot have sockets.
gemsockets - Controls the maximum number of sockets allowed on this item. This is limited by the item's size based on the "invwidth" and "invheight" fields. This also compares with the "MaxSock1", "MaxSock25" and "MaxSock40" fields from the ItemTypes.txt file.
gemapplytype - Determines which affect from a gem or rune will be applied when it is socketed into this item (See gems.txt)

Code	Description
0	Weapon
1	Armor or Helmet
2	Shield

flippyfile - Controls which DC6 file to use for displaying the item in the game world when it is dropped on the ground (uses the file name as the input)
invfile - Controls which DC6 file to use for displaying the item graphics in the inventory (uses the file name as the input)
uniqueinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Unique quality item (uses the file name as the input)
setinvfile - Controls which DC6 file to use for displaying the item graphics in the inventory when it is a Set quality item (uses the file name as the input)

useable - Boolean Field. If equals 1, then the item can be used with the right-click mouse button command (this only works with specific belt items or quest items). If equals 0, then ignore this.

stackable - Boolean Field. If equals 1, then the item will use a quantity field and handle stacking functionality. This can depend on if the item type is throwable, is a type of ammunition, or is some other kind of miscellaneous item. If equals 0, then the item cannot be stacked.
minstack - Controls the minimum stack count or quantity that is allowed on the item. This field depends on the "stackable" field being enabled.
maxstack - Controls the maximum stack count or quantity that is allowed on the item. This field depends on the "stackable" field being enabled.
spawnstack - Controls the stack count or quantity that the item can spawn with. This field depends on the "stackable" field being enabled.

Transmogrify - Boolean Field. If equals 1, then the item will use the transmogrify function. If equals 0, then ignore this. This field depends on the "useable" field being enabled.
TMogType - Links to a "code" field to determine which item is chosen to transmogrify this item to.
TMogMin - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the "TMogType" field, and that the transmogrify item has quantity.
TMogMax - Controls the minimum quantity that the transmogrify item will have. This depends on what item was chosen in the "TMogType" field, and that the transmogrify item has quantity.

type - Points to an Item Type defined in the ItemTypes.txt file, which controls how the item functions
type2 - Points to a secondary Item Type defined in the ItemTypes.txt file, which controls how the item functions. This is optional but can add more functionalities and possibilities with the item.

dropsound - Points to sound defined in the sounds.txt file. Used when the item is dropped on the ground.
dropsfxframe - Defines which frame in the "flippyfile" animation to play the "dropsound" sound when the item is dropped on the ground.
usesound - Points to sound defined in the sounds.txt file. Used when the item is moved in the inventory or used.

unique - Boolean Field. If equals 1, then the item can only spawn as a Unique quality type. If equals 0, then the item can spawn as other quality types.

transparent - Boolean Field. If equals 1, then the item will be drawn transparent on the player model (similar to ethereal models). If equals 0, then the item will appear solid on the player model.
transtbl - Controls what type of transparency to use, based on the "transparent" field being enabled.

Code	Description
0	Transparency at 25%
1	Transparency at 50%
2	Transparency at 75%
3	Black Alpha Transparency
4	White Alpha Transparency
5	No Transparency
6	Dark Transparency (Unused)
7	Highlight Transparency (Used when mousing over the unit)
8	Blended

lightradius - Controls the value of the light radius that this item can apply on the monster. This only affects monsters with this item equipped, not other types of units. This is ignored if the item's component on the monster is "lit", "med", or "hvy".

belt - Controls which belt type to use for belt items only. This field determines what index entry in the belts.txt file to use.

quest - Controls what quest class is tied to the item which can enable certain item functionalities for a specific quest. Any value greater than 0 will also mean the item is flagged as a quest item, which can affect how it is displayed in tooltips, how it is traded with other players, its item rarity, and how it cannot be sold to an NPC. If equals 0, then the item will not be flagged as a quest item.

Code	Description
0	Not a quest item
1	Act 1 Prologue
2	Den of Evil
3	Sisters' Burial Grounds
4	Tools of the Trade
5	The Search for Cain
6	The Forgotten Tower
7	Sisters to the Slaughter
8	Act 2 Prologue
9	Radament's Lair
10	The Horadric Staff
11	The Tainted Sun
12	The Arcane Sanctuary
13	The Summoner
14	The Seven Tombs
15	Act 2 Traversed
16	Lam Esen's Tome
17	Khalim's Will
18	Blade of the Old Religion
19	The Golden Bird
20	The Blackened Temple
21	The Guardian
22	Act 4 Prologue
23	The Fallen Angel
24	Terror's End
25	The Hellforge
26	Rogue Warning
27	Guard in Town Warning
28	Guard in Desert Warning
29	Dark Wanderer Seen
30	Angel Warning
31	Respec from Akara Complete Act 5 Prologue
32	Siege on Harrogath
33	Rescue on Mount Arreat
34	Prison of Ice
35	Betrayal of Harrogath
36	Rite of Passage
37	Eve of Destruction

questdiffcheck - Boolean Field. If equals 1 and the "quest" field is enabled, then the game will check the current difficulty setting and will tie that difficulty setting to the quest item. This means that the player can have more than 1 of the same quest item as long each they are obtained per difficulty mode (Normal / Nightmare / Hell). If equals 0 and the "quest" field is enabled, then the player can only have 1 count of the quest item in the inventory, regardless of difficulty.

missiletype - Points to the "Id" field from the Missiles.txt file, which determines what type of missile is used when using the throwing weapons
durwarning - Controls the threshold value for durability to display the low durability warning UI. This is only used if the item has durability.
qntwarning - Controls the threshold value for quantity to display the low quantity warning UI. This is only used if the item has stacks.

mindam - The minimum physical damage provided by the item
maxdam - The maximum physical damage provided by the item

StrBonus - The percentage multiplier that gets multiplied the player's current Strength attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.
DexBonus - The percentage multiplier that gets multiplied the player's current Dexterity attribute value to modify the bonus damage percent from the equipped item. If this equals 1, then default the value to 100.

gemoffset - Determines the starting index offset for reading the gems.txt file when determining what effects gems or runes will have the item based on the "gemapplytype" field. For example, if this value equals 9, then the game will start with index 9 ("Chipped Emerald") and ignore the previously defined gems in the gems.txt file, which can mean that those ignored gems will not apply modifiers when socketed into the item.
bitfield1 - Controls different flags that can affect the item. Uses an integer value to check against different bit fields by using the "&" operator. For example, if the value equals 5 (binary = 101) then that returns true for both the 4 (binary = 100) and 1 (binary = 1) bit field values.

Bit Field One Bits	Binary Equivalent Value	Description
1	1	Allow the item to be capable of having Magic quality
2	10	The item is classified as metal
4	100	The item is classified as a spellcaster item (currently does nothing)
8	1000	The item is classified as a skill based item (currently does nothing)

The following fields are separated per NPC in each Act:
[NPC]Min - Minimum amount of this item type in Normal rarity that the NPC can sell at once
[NPC]Max - Maximum amount of this item type in Normal rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicMin - Minimum amount of this item type in Magical rarity that the NPC can sell at once
[NPC]MagicMax - Maximum amount of this item type in Magical rarity that the NPC can sell at once. This must be equal to or greater than the minimum amount.
[NPC]MagicLvl - Maximum magic level allowed for this item type in Magical rarity

Where [NPC] is one of the following:

Charsi
Gheed
Akara
Fara
Lysander
Drognan
Hratli
Alkor
Ormus
Elzix
Asheara
Cain
Halbu
Jamella
Larzuk
Malah
Anya

Transform - Controls the color palette change of the item for the character model graphics
InvTrans - Controls the color palette change of the item for the inventory graphics

Code	Color
0	No color change
1	Grey
2	Grey 2
3	Gold
4	Brown
5	Grey Brown
6	Inventory Grey
7	Inventory Grey 2
8	Inventory Grey Brown

SkipName - Boolean Field. If equals 1 and the item is Unique rarity, then skip adding the item's base name in its title. If equals 0, then ignore this.

NightmareUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Nightmare difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.
HellUpgrade - Links to another item's "code" field. Used to determine which item will replace this item when being generated in the NPC's store while the game is playing in Hell difficulty. If this field's code equals "xxx", then this item will not change in this difficulty.

Nameable - Boolean Field. If equals 1, then the item's name can be personalized by Anya for the Act 5 Betrayal of Harrogath quest reward. If equals 0, then the item cannot be used for the personalized name reward.
PermStoreItem - Boolean Field. If equals 1, then this item will always appear on the NPC's store. If equals 0, then the item will randomly appear on the NPC's store when appropriate.
diablocloneweight - The amount of weight added to the diablo clone progress when this item is sold. When offline, selling this item will instead immediately spawn diablo clone.

The following fields are exclusive to the weapons.txt file because these fields only work with Weapon type items:

1or2handed - Boolean Field. If equals 1, then the item will be treated as a one-handed and two-handed weapon by the Barbarian class. If equals 0, then the Barbarian can only use this weapon as either one-handed or two-handed, but not both.
2handed - Boolean Field. If equals 1, then the item will be treated as two-handed weapon. If equals 0, then the item will be treated as one-handed weapon.
2handedwclass - Defines the two-handed weapon class, which controls what character animations are used when the weapon is equipped

Code	Description
1hs	One Handed Swing
1ht	One Handed Thrust
bow	Bow
2hs	Two Handed Swing
2ht	Two Handed Thrust
1js	Left Jab Right Swing
1jt	Left Jab Right Thrust
1ss	Left Swing Right Swing
1st	Left Swing Right Thrust
stf	Staff
xbw	Crossbow
ht1	One Hand-To-Hand
ht2	Two Hand-To-Hand

2handmindam - The minimum physical damage provided by the weapon if the item is two-handed. This relies on the "2handed" field being enabled.
2handmaxdam - The maximum physical damage provided by the weapon if the item is two-handed. This relies on the "2handed" field being enabled.

hit class - Defines the hit class of the weapon which is used to know what SFX to use when the weapon hits an enemy

Code	Hit Class
(empty)	None
hth	Hand To Hand
1hss	One Hand Swing vs. Small
1hsl	One Hand Swing vs. Large
2hss	Two Hand Swing vs. Small
2hsl	Two Hand Swing vs. Large

1ht	One Hand Thrust
2ht	Two Hand Thrust
club	Club
staf	Staff
bow	Bow
xbow	Crossbow
claw	Claw
over	Overlay

minmisdam - The maximum physical damage provided by the item if it is a throwing weapon
maxmisdam - The maximum physical damage provided by the item if it is a throwing weapon

rangeadder - Adds extra range in grid spaces for melee attacks while the melee weapon is equipped. The baseline melee range is 1, and this field adds to that range.
wclass - Defines the one-handed weapon class, which controls what character animations are used when the weapon is equipped

Code	Description
1hs	One Handed Swing
1ht	One Handed Thrust
bow	Bow
2hs	Two Handed Swing
2ht	Two Handed Thrust
1js	Left Jab Right Swing
1jt	Left Jab Right Thrust
1ss	Left Swing Right Swing
1st	Left Swing Right Thrust
stf	Staff
xbw	Crossbow
ht1	One Hand-To-Hand
ht2	Two Hand-To-Hand

wanderingmon.txt

Overview

This file controls the list of monsters that can be used as good NPC units to randomly place in certain area levels. The “actinfo.txt” file controls the statistics on when to spawn wandering monsters, and this file simply controls the list of possible monsters to choose from.

Data Fields

class - Uses a monster “Id” defined from the monstats.txt file. Monsters defined here are added to a list which is used to randomly pick a monster to spawn in an area level.

Reference Data Files

The following files are considered hardcoded reference files used for specific fields or as indices for other data files.

ArmType.txt
bodylocs.txt
colors.txt
compcode.txt
Composit.txt
cubemod.txt
ElemTypes.txt
events.txt
HitClass.txt
lowqualityitems.txt
misscalc.txt
MonMode.txt
MonPlace.txt
ObjMode.txt
ObjType.txt
PlayerClass.txt
PlrMode.txt
PlrType.txt
skillcalc.txt
StorePage.txt