Entities

This chapter contains required and recommended changes to entities.

Game Challenges

Co-op

```
entity {
	entityDef coop_game_challenge {
	inherit = "info/game_challenge/battlearena";
	class = "idGameChallenge_PVP";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true;
	disableAIPooling = false;
	edit = {
		networkSerializeTransforms = false;
		modeFlags = {
			allowRespawning = true;
			allowBulletPenetration = false;
		}
		difficultySettings = {
			playerIncomingDamageScale = "campaign/playerincomingdamage";
			aiIncomingDamageScale = "campaign/aiincomingdamage";
			healthPickupScale = "campaign/healthpickup";
			armorPickupScale = "campaign/armorpickup";
			ammoPickupScale = "campaign/ammopickup";
			bfgAmmoPickupScale = "campaign/bfgammopickup";
			healthDropScale = "campaign/healthdrop";
			armorDropScale = "campaign/armordrop";
			ammoDropScale = "campaign/ammodrop";
			bfgAmmoDropScale = "campaign/bfgammodrop";
		}
		scoreLimit = 99;
		numLives = 0;
		numRounds = 99;
		actorModifierListDecl = "actormodifiers_pvp";
		gcGameEventCallouts = {
			slayerPowerWeapon = "pvp/combat_events/slayer_power_weapon";
```

```
demonSummonedCallout = "pvp/combat_events/general_card_event";
demonEffectCallout = "pvp/combat_events/general_card_event";
demonEffectStatusTimer = "pvp/status_timers/demon_effect";
demonQuickUse1Callout = "pvp/combat_events/general_card_event";
demonQuickUse2Callout = "pvp/combat_events/general_card_event";
damageBoostCallout = "pvp/combat_events/general_card_event";
damageBoostStatusTimer = "pvp/status_timers/timer_damage_boost";
hasteCallout = "pvp/combat_events/general_card_event";
hasteStatusTimer = "pvp/status_timers/timer_haste";
mitigationCallout = "pvp/combat_events/general_card_event";
mitigationStatusTimer = "pvp/status_timers/timer_damage_mitigation";
invulnerableCallout = "pvp/combat_events/general_card_event";
invulnerableStatusTimer = "pvp/status_timers/timer_invulnerable";
berserkCallout = "pvp/combat_events/general_card_event";
berserkStatusTimer = "pvp/status_timers/timer_berserk";
regenCallout = "pvp/combat_events/general_card_event";
regenStatusTimer = "pvp/status_timers/timer_regeneration";
lootBlockedCallout = "pvp/voiced/loot_blocked";
lootBlockedStatusTimer = "pvp/status_timers/timer_loot_blocked";
extraLifeCallout = "pvp/combat_events/general_card_event";
demonCriticalHealth = "";
demonCriticalRecovery = "pvp/voiced/demon_healed";
slayerCriticalHealth = "pvp/voiced/slayer_health_critical";
everyoneCritical = "main_heavy";
}
hitConfirmSoundsInfo = "default";
characterStatusEventText = {
invulnerableText = {
textId = "#str_decl_powerup_statuseffect_GHOST45053";
color = {
r = 0.87450999;
g = 0.87450999;
b = 0.87450999;
}
}
toughenedText = {
textId = "#str_decl_powerup_statuseffect_GHOST45054";
color = {
r = 0.87450999;
```

```
        g = 0.87450999;
        b = 0.87450999;
      }
    }
    vulnerableText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45055";
      color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
      }
    }
    strengthenedText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45056";
      color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
      }
    }
    hastedText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45057";
      color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
      }
    }
    slowedText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45058";
      color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
      }
    }
    berserkingText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45059";
      color = {
```

```
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
    }
}
lootBlockedText = {
    textId = "#str_decl_powerup_statuseffect_GHOST45060";
    color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
    }
}
}
aiSpawnPoolDecl = "maps/game/pvp/battlemode";
enableBrinkOfDeath = true;
desummonKillDamage = "damage/hazard/pvp_round_kill";
slayerHighlightDecl = "pvp/demon_view_slayer_outline";
demonHighlightDecl = "pvp/demon_view_slayer_outline";
teammateHighlightDecl = "pvp/demon_view_slayer_outline";
slayerHighlightLOSBoxDecl = "highlight_los_slayer";
demonSpawnTargetEntity = "online/summon_target";
demonSpawnTargetEntityOneHit = "online/summon_target_onehit";
playerSpawnDef = "player";
teamSuperSettings = {
    damageFactorAI = 9.99999975e-05;
    tickFactor = 0.00329999998;
}
playerAlwaysFullBodyGibs = true;
maxPlayersPerTeam = {
    ptr = {
        ptr[0] = 1;
        ptr[1] = 2;
    }
}
demonOutlineColor = {
    g = 0.501960993;
}
demonAllyOutlineColor = {
```

```
    r = 1;
    g = 0.501960993;
    b = 0;
  }
  fillColorDemonSees = {
    r = 0.199999988;
    g = 0.199999988;
    b = 0.199999988;
    a = 0.850000024;
  }
  fillColorSlayerSees = {
    r = 0.199999988;
    g = 0.199999988;
    b = 0.199999988;
    a = 0.850000024;
  }
  fillColorHitFlash = {
    g = 0;
    b = 0;
    a = 0.501960993;
  }
  slayerHighlightOptions = {
    allyDemon = "EHM_ALWAYS";
    enemyDemon = "EHM_ALWAYS";
    allySlayer = "EHM_ALWAYS";
    enemySlayer = "EHM_ALWAYS";
  }
  demonHighlightOptions = {
    allyDemon = "EHM_ALWAYS";
    enemyDemon = "EHM_ALWAYS";
    allySlayer = "EHM_ALWAYS";
    enemySlayer = "EHM_ALWAYS";
  }
  raceToStyle = true;
  pvpGameEventCallouts = {
    roundOne = "";
    roundTwo = "";
    roundThree = "";
    roundFour = "";
```

```
    roundFinal = "";
    preMatchFiveSecondsRemaining = "";
    preMatchFourSecondsRemaining = "";
    preMatchThreeSecondsRemaining = "";
    preMatchTwoSecondsRemaining = "";
    preMatchOneSecondRemaining = "";
    roundStart = "";
    finalRound = "";
    demonsRoundLost = "";
    slayerRoundLost = "";
    slayerKilled = "pvp/voiced/slayer_killed";
    demonKilled = "pvp/voiced/slayer_killed";
    demonsKilled = "pvp/voiced/slayer_killed";
    slayerVictory = "";
    demonVictory = "";
    demonRespawningSoon = "";
    respawnFiveSecondsRemaining = "";
    respawnFourSecondsRemaining = "";
    respawnThreeSecondsRemaining = "";
    respawnTwoSecondsRemaining = "";
    respawnOneSecondRemaining = "";
    demonRespawned = "pvp/voiced/demon_resurrected";
    delayDuringSync = {
        num = 1;
        item[0] = "pvp/voiced/slayer_killed";
    }
}
jockeyTimeDuration = {
    value = 0;
}
roundCalloutTimeDuration = {
    value = 0.1;
}
roundStartCalloutDelaySec = {
    value = 0;
}
roundEndCalloutDelaySec = {
    value = 0;
}
```

```
    upgradeUIDelaySec = {
        value = 0;
    }
    upgradeUIDuration = {
        value = 0;
    }
    matchEndCalloutDelaySec = {
        value = 0;
    }
    matchEndResultsDelaySec = {
        value = 0;
    }
    pvpProgressionScoringDecl = "battle_arena";
    pvpLifecycleManager = {
        podiumAvatarEntDefs = {
            num = 7;
            item[0] = "podiums/avatars/archvile";
            item[1] = "podiums/avatars/doom_marine";
            item[2] = "podiums/avatars/mancubus";
            item[3] = "podiums/avatars/pain_elemental";
            item[4] = "podiums/avatars/revenant";
            item[5] = "podiums/avatars/marauder";
            item[6] = "podiums/avatars/dreadknight";
        }
        podiumLayers = {
            num = 1;
            item[0] = "game/pvp/podium_stage";
        }
        slayerPodiumEntities = "pvp_match_slayer_podium_";
        demonPodiumEntities = "pvp_match_demon_podium_slot_";
        characterAnimBlendMS = 0;
        playerAppearSound = "play_pvp_staging_spawnin";
    }
    slayerPVPLoadoutDecl = "pvploadout/default";
    demonPVPLoadoutDecl = "pvpdemonloadout/default";
    respawnTimeSec = {
        branchPairs = {
            num = 1;
            item[0] = {
```

```
        branchKey = "CONTROLLERPAD_DECL";
        branchResult = {
          value = 0.1;
        }
      }
    }
  }
  respawnTimeCapSec = {
    defaultValue = {
      value = 0.1;
    }
    branchPairs = {
      num = 1;
      item[0] = {
        branchKey = "CONTROLLERPAD_DECL";
        branchResult = {
          value = 0.1;
        }
      }
    }
  }
  respawnStatusEffects = {
    num = 0;
  }
  idealRespawnDistanceFromSlayer = 0;
  slayerLayersToActivate = {
    num = 1;
    item[0] = "game/pvp/slayer_team";
  }
  demonLayersToActivate = {
    num = 1;
    item[0] = "game/pvp/demon_team";
  }
  layersToDeactivate = {
    num = 1;
    item[0] = "game/pvp/permanently_hidden";
  }
  musicWinState = "music_ghost_states/pvp_lose";
  musicLoseState = "music_ghost_states/pvp_lose";
```

```
musicHealthCriticalState = "music_ghost_states/main_heavy";
closeCallHealthThreshold = 50;
clutchThreshold = 5;
comebackThreshold = 3;
surpriseTimeLimit = {
value = 3;
}
surviveThreshold = 4;
requiredPlayerCount = 2;
respawnHealthScalar = 0.5;
powerUpgradeRound = 99;
soundOcclusionBypass = false;
gameDifficulty = "DIFFICULTY_NIGHTMARE";
spawnPosition = {
x = 1;
y = 1;
z = 1;
}
}
}
}
```

Slayer vs Slayer

```
entity {
entityDef deathmatch_game_challenge {
inherit = "info/game_challenge/battlearena";
class = "idGameChallenge_PVP";
expandInheritance = false;
poolCount = 0;
poolGranularity = 2;
networkReplicated = true;
disableAIPooling = false;
edit = {
networkSerializeTransforms = false;
modeFlags = {
allowRespawning = true;
allowBulletPenetration = false;
}
```

```
  scoreLimit = 3;
  numLives = 0;
  numRounds = 5;
  actorModifierListDecl = "actormodifiers_pvp";
  gcGameEventCallouts = {
    slayerPowerWeapon = "pvp/combat_events/slayer_power_weapon";
    demonSummonedCallout = "pvp/combat_events/general_card_event";
    demonEffectCallout = "pvp/combat_events/general_card_event";
    demonEffectStatusTimer = "pvp/status_timers/demon_effect";
    demonQuickUse1Callout = "pvp/combat_events/general_card_event";
    demonQuickUse2Callout = "pvp/combat_events/general_card_event";
    damageBoostCallout = "pvp/combat_events/general_card_event";
    damageBoostStatusTimer = "pvp/status_timers/timer_damage_boost";
    hasteCallout = "pvp/combat_events/general_card_event";
    hasteStatusTimer = "pvp/status_timers/timer_haste";
    mitigationCallout = "pvp/combat_events/general_card_event";
    mitigationStatusTimer = "pvp/status_timers/timer_damage_mitigation";
    invulnerableCallout = "pvp/combat_events/general_card_event";
    invulnerableStatusTimer = "pvp/status_timers/timer_invulnerable";
    berserkCallout = "pvp/combat_events/general_card_event";
    berserkStatusTimer = "pvp/status_timers/timer_berserk";
    regenCallout = "pvp/combat_events/general_card_event";
    regenStatusTimer = "pvp/status_timers/timer_regeneration";
    lootBlockedCallout = "pvp/voiced/loot_blocked";
    lootBlockedStatusTimer = "pvp/status_timers/timer_loot_blocked";
    extraLifeCallout = "pvp/combat_events/general_card_event";
    demonCriticalHealth = "pvp/voiced/demon_health_critical";
    demonCriticalRecovery = "pvp/voiced/demon_healed";
    slayerCriticalHealth = "pvp/voiced/slayer_health_critical";
  }
  hitConfirmSoundsInfo = "default";
  characterStatusEventText = {
    invulnerableText = {
      textId = "#str_decl_powerup_statuseffect_GHOST45053";
      color = {
        r = 0.87450999;
        g = 0.87450999;
        b = 0.87450999;
      }
    }
```

```
    toughenedText = {
        textId = "#str_decl_powerup_statuseffect_GHOST45054";
        color = {
            r = 0.87450999;
            g = 0.87450999;
            b = 0.87450999;
        }
    }
    vulnerableText = {
        textId = "#str_decl_powerup_statuseffect_GHOST45055";
        color = {
            r = 0.87450999;
            g = 0.87450999;
            b = 0.87450999;
        }
    }
    strengthenedText = {
        textId = "#str_decl_powerup_statuseffect_GHOST45056";
        color = {
            r = 0.87450999;
            g = 0.87450999;
            b = 0.87450999;
        }
    }
    hastedText = {
        textId = "#str_decl_powerup_statuseffect_GHOST45057";
        color = {
            r = 0.87450999;
            g = 0.87450999;
            b = 0.87450999;
        }
    }
    slowedText = {
        textId = "#str_decl_powerup_statuseffect_GHOST45058";
        color = {
            r = 0.87450999;
            g = 0.87450999;
            b = 0.87450999;
        }
    }
```

```
berserkingText = {
textId = "#str_decl_powerup_statuseffect_GHOST45059";
color = {
r = 0.87450999;
g = 0.87450999;
b = 0.87450999;
}
}
lootBlockedText = {
textId = "#str_decl_powerup_statuseffect_GHOST45060";
color = {
r = 0.87450999;
g = 0.87450999;
b = 0.87450999;
}
}
}
aiSpawnPoolDecl = "maps/game/pvp/battlemode";
enableBrinkOfDeath = false;
desummonKillDamage = "damage/hazard/pvp_round_kill";
slayerHighlightDecl = "pvp/slayer_view_demon_outline";
demonHighlightDecl = "pvp/demon_view_teammate_outline";
teammateHighlightDecl = "pvp/demon_view_teammate_outline";
slayerHighlightLOSBoxDecl = "highlight_los_slayer";
demonSpawnTargetEntity = "online/summon_target";
demonSpawnTargetEntityOneHit = "online/summon_target_onehit";
playerSpawnDef = "player";
teamSuperSettings = {
damageFactorAI = 9.99999975e-05;
tickFactor = 0.00329999998;
}
playerAlwaysFullBodyGibs = true;
maxPlayersPerTeam = {
ptr = {
ptr[0] = 1;
ptr[1] = 2;
}
}
demonOutlineColor = {
g = 0.501960993;
```

```
    }
    demonAllyOutlineColor = {
        r = 1;
        g = 0.501960993;
        b = 0;
    }
    fillColorDemonSees = {
        r = 0.199999988;
        g = 0.199999988;
        b = 0.199999988;
        a = 0.850000024;
    }
    fillColorSlayerSees = {
        r = 0.199999988;
        g = 0.199999988;
        b = 0.199999988;
        a = 0.850000024;
    }
    fillColorHitFlash = {
        g = 0;
        b = 0;
        a = 0.501960993;
    }
    slayerHighlightOptions = {
        allyDemon = "EHM_NONE";
        enemyDemon = "EHM_NONE";
        allySlayer = "EHM_ALWAYS";
        enemySlayer = "EHM_NONE";
    }
    demonHighlightOptions = {
        allyDemon = "EHM_ALWAYS";
        enemyDemon = "EHM_ALWAYS";
        allySlayer = "EHM_ALWAYS";
        enemySlayer = "EHM_ALWAYS";
    }
    raceToStyle = true;
    pvpGameEventCallouts = {
        roundOne = "pvp/voiced/round_one";
        roundTwo = "pvp/voiced/round_two";
        roundThree = "pvp/voiced/round_three";
```

```
roundFour = "pvp/voiced/round_four";
roundFinal = "pvp/voiced/final_round";
preMatchFiveSecondsRemaining = "pvp/chimes/five";
preMatchFourSecondsRemaining = "pvp/chimes/four";
preMatchThreeSecondsRemaining = "pvp/chimes/three";
preMatchTwoSecondsRemaining = "pvp/chimes/two";
preMatchOneSecondRemaining = "pvp/chimes/one";
roundStart = "pvp/voiced/fight";
finalRound = "pvp/voiced/final_round";
demonsRoundLost = "pvp/voiced/demons_round_lost";
slayerRoundLost = "pvp/voiced/slayer_round_lost";
slayerKilled = "pvp/voiced/slayer_killed";
demonKilled = "";
demonsKilled = "pvp/voiced/demons_killed";
slayerVictory = "pvp/voiced/slayer_won_the_match";
demonVictory = "pvp/voiced/demons_won_the_match";
demonRespawningSoon = "pvp/voiced/demon_resurrecting_soon";
respawnFiveSecondsRemaining = "pvp/chimes/resurrect_five";
respawnFourSecondsRemaining = "pvp/chimes/resurrect_four";
respawnThreeSecondsRemaining = "pvp/chimes/resurrect_three";
respawnTwoSecondsRemaining = "pvp/chimes/resurrect_two";
respawnOneSecondRemaining = "pvp/chimes/resurrect_one";
demonRespawned = "pvp/voiced/demon_resurrected";
delayDuringSync = {
num = 3;
item[0] = "pvp/voiced/slayer_killed";
item[1] = "pvp/voiced/demons_won_the_match";
item[2] = "pvp/voiced/slayer_round_lost";
}
}
jockeyTimeDuration = {
value = 5;
}
roundCalloutTimeDuration = {
value = 3;
}
roundStartCalloutDelaySec = {
value = 2;
}
upgradeUIDelaySec = {
```

```
        value = 0;
    }
    matchEndCalloutDelaySec = {
        value = 0;
    }
    matchEndResultsDelaySec = {
        value = 0;
    }
    pvpProgressionScoringDecl = "battle_arena";
    pvpLifecycleManager = {
        podiumAvatarEntDefs = {
            num = 7;
            item[0] = "podiums/avatars/archvile";
            item[1] = "podiums/avatars/doom_marine";
            item[2] = "podiums/avatars/mancubus";
            item[3] = "podiums/avatars/pain_elemental";
            item[4] = "podiums/avatars/revenant";
            item[5] = "podiums/avatars/marauder";
            item[6] = "podiums/avatars/dreadknight";
        }
        podiumLayers = {
            num = 1;
            item[0] = "game/pvp/podium_stage";
        }
        slayerPodiumEntities = "pvp_match_slayer_podium_";
        demonPodiumEntities = "pvp_match_demon_podium_slot_";
        characterAnimBlendMS = 0;
        playerAppearSound = "play_pvp_staging_spawnin";
    }
    slayerPVPLoadoutDecl = "pvploadout/default";
    demonPVPLoadoutDecl = "pvpdemonloadout/default";
    respawnTimeSec = {
        branchPairs = {
            num = 1;
            item[0] = {
                branchKey = "CONTROLLERPAD_DECL";
                branchResult = {
                    value = 0;
                }
            }
```

```
        }
    }
    respawnTimeCapSec = {
        defaultValue = {
            value = 0;
        }
        branchPairs = {
            num = 1;
            item[0] = {
                branchKey = "CONTROLLERPAD_DECL";
                branchResult = {
                    value = 0;
                }
            }
        }
    }
    respawnStatusEffects = {
        num = 0;
    }
    idealRespawnDistanceFromSlayer = 100;
    slayerLayersToActivate = {
        num = 1;
        item[0] = "game/pvp/slayer_team";
    }
    demonLayersToActivate = {
        num = 1;
        item[0] = "game/pvp/demon_team";
    }
    layersToDeactivate = {
        num = 1;
        item[0] = "game/pvp/permanently_hidden";
    }
    musicWinState = "music_ghost_states/pvp_win";
    musicLoseState = "music_ghost_states/pvp_lose";
    closeCallHealthThreshold = 50;
    clutchThreshold = 5;
    comebackThreshold = 3;
    surpriseTimeLimit = {
        value = 3;
    }
```

```
    surviveThreshold = 4;
    requiredPlayerCount = 2;
    soundOcclusionBypass = false;
  }
}
}
```

Pickups

Pickup

> For non-respawning pickups, only use the idProp2/idMover entities.

idProp2

> Remove the **bindInfo** if this is a self-respawning pickup.

```
entity {
	entityDef deathmatch_pickup_mega_health_1 {
	inherit = "pickup/health/large";
	class = "idProp2";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true; // Must be true
	disableAIPooling = false;
	edit = {
		bindInfo = { // remove if self-respawning pickup
			bindParent = "deathmatch_pickup_mega_health_1_offset_mover"; // fixes entity offset
		}
		renderModelInfo = {
			model = "art/pickups/health/megaHealth_head_a.lwo";
			contributesToLightProbeGen = false;
			noAmbient = true;
			scale = {
				x = 1;
				y = 1;
				z = 1;
			}
			emissiveColor = {
				r = 0;
				g = 0.913725019;
```

```
		}
		emissiveScale = 3;
	}
	spawn_statIncreases = {
		num = 1;
		item[0] = {
			stat = "STAT_ITEMS_SPAWNED";
			increase = 1;
		}
	}
	equipOnPickup = true;
	lootStyle = "LOOT_TOUCH";
	triggerDef = "trigger/props/megahealth";
	isStatic = true;
	canBePossessed = true;
	fxDecl = "pickups/megahealth_head";
	useableComponentDecl = "mega_health";
	sound_spawn = "play_pickup_loop_05";
	sound_stop = "stop_pickup_loop_05";
	pickup_statIncreases = {
		num = 1;
		item[0] = {
			stat = "STAT_HEALTH_PICKUP";
			increase = 1;
		}
	}
	spawnPosition = { // Be sure to set the spawn position
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	spawnOrientation = { // Be sure to set the spawn orientation
		mat = {
			mat[0] = {
				x = -1;
				y = 0;
				z = 0;
			}
			mat[1] = {
```

```
        x = 0;
        y = -1;
        z = 0;
    }
}
}
targets = {
num = 1;
item[0] = "deathmatch_pickup_mega_health_1_respawn_relay"; // Activates relay to start
respawn process
}
isLootDrop = true; // Must be true
}
}
}
```

idMover

> Optional, but will fix the entity offset for the pickup. However, the pickup's spawn orientation will revert to its default value.

```
entity {
entityDef deathmatch_pickup_mega_health_1_offset_mover {
inherit = "func/mover";
class = "idMover";
expandInheritance = false;
poolCount = 0;
poolGranularity = 2;
networkReplicated = true;
disableAIPooling = false;
edit = {
spawnPosition = { // Be sure to set the spawn position
x = -11.11;
y = -11.11;
z = -11.11;
}
spawnOrientation = { // The spawn orientation does not matter
mat = {
```

```
            mat[0] = {
                x = -1;
                y = 0;
                z = 0;
            }
            mat[1] = {
                x = 0;
                y = -1;
                z = 0;
            }
        }
    }
    renderModelInfo = {
        model = NULL;
    }
    clipModelInfo = {
        clipModelName = NULL;
    }
}
}
```

Respawning Pickups

Target Spawn

```
entity {
entityDef deathmatch_pickup_mega_health_1_respawn {
inherit = "target/spawn";
class = "idTarget_Spawn";
expandInheritance = false;
poolCount = 0;
poolGranularity = 2;
networkReplicated = false; // Must be false
disableAIPooling = false;
edit = {
flags = {
```

```
		noFlood = true;
	}
	spawnConditions = {
		maxCount = 0;
		reuseDelaySec = 0;
		doBoundsTest = false;
		boundsTestType = "BOUNDSTEST_NONE";
		fovCheck = 0;
		minDistance = 0;
		maxDistance = 0;
		neighborSpawnerDistance = -1;
		LOS_Test = "LOS_NONE";
		playerToTest = "PLAYER_SP";
		conditionProxy = "";
	}
	spawnEditableShared = {
		groupName = "";
		deathTrigger = "";
		coverRadius = 0;
		maxEnemyCoverDistance = 0;
	}
	entityDefs = {
	num = 1;
	item[0] = {
		name = "deathmatch_pickup_mega_health_1"; // Activates respawning pickup
	}
}
	conductorEntityAIType = "SPAWN_AI_TYPE_ANY";
	initialEntityDefs = {
		num = 0;
	}
	spawnEditable = {
		spawnAt = "";
		copyTargets = false;
		additionalTargets = {
			num = 0;
		}
		overwriteTraversalFlags = true;
		traversalClassFlags = "CLASS_A";
```

```
		combatHintClass = "CLASS_ALL";
		spawnAnim = "";
		aiStateOverride = "AIOVERRIDE_TELEPORT";
		initialTargetOverride = "";
	}
	portal = "";
	targetSpawnParent = "";
	disablePooling = false;
	spawnPosition = { // Be sure to set the spawn position
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	spawnOrientation = { // Be sure to set the spawn orientation
		mat = {
			mat[0] = {
				x = -1;
				y = 0;
				z = 0;
			}
			mat[1] = {
				x = 0;
				y = -1;
				z = 0;
			}
		}
	}
}
}
```

Relay

```
entity {
	entityDef deathmatch_pickup_mega_health_1_respawn_relay {
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
```

```
	poolGranularity = 2;
	networkReplicated = false; // Must be false
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1; // Activates on one count
		spawnPosition = { // Spawn position is not important
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "deathmatch_pickup_mega_health_1_respawn"; // Activates target spawn
		}
        delay = 15; // Must have a delay
		repeat = true; // Can activate again
	}
}
}
```

Chainsaw Pickups

Info

> Chainsaw fuel pickups cannot be picked-up after a client has already picked-up 2 of them in the same map.

> This method will give ALL Slayers +1 fuel pickup when one Slayer tries to pickup a fuel canister.

idDynamicEntity

```
entity {
	entityDef coop_pickup_ammo_gas_1_fake {
	inherit = "func/dynamic";
	class = "idDynamicEntity"; // This is the fuel canister model
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		spawnPosition = { // Be sure to set the spawn position
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = { // Be sure to set the spawn orientation
			mat = {
				mat[0] = {
					x = -1;
					y = 0;
					z = 0;
				}
				mat[1] = {
					x = 0;
```

```
        y = -1;
        z = 0;
    }
}
renderModelInfo = {
    model = "art/pickups/ammo/ammo_chainsaw_01.lwo";
    contributesToLightProbeGen = false;
    ignoreDesaturate = true;
    emissiveScale = 0.2;
    scale = {
        x = 2;
        y = 2;
        z = 2;
    }
}
clipModelInfo = {
    type = "CLIPMODEL_NONE";
}
dormancy = {
    canBecomeDormant = true;
}
}
}
```

idTrigger

```
entity {
entityDef coop_pickup_ammo_gas_1_trigger {
inherit = "trigger/trigger";
class = "idTrigger";
expandInheritance = false;
poolCount = 0;
poolGranularity = 2;
networkReplicated = false;
disableAIPooling = false;
edit = {
```

```
	spawnPosition = { // Set the spawn position to same as the idDynamicEntity
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	renderModelInfo = {
		model = NULL;
	}
	clipModelInfo = {
		type = "CLIPMODEL_BOX";
		size = {
			x = 2;
			y = 2;
			z = 2;
		}
	}
	targets = {
		num = 1;
		item[0] = "coop_pickup_ammo_gas_1_fake_timeline"; // Targets the removal or respawn timeline
	}
	playerCanActivate = true;
	clientCanActivate = true;
	triggerForAllClients = true;
	triggerOnEnter = true;
	triggerOnce = true; // Set to false for respawning pickup
	wait = 0; // Add a wait timer for respawning pickup
}
}
}
```

Global SFX / Give Fuel

A separate relay entity must exist once to activate the sound effect & give fuel ammo command.

```
entity {
	entityDef multiplayer_chainsaw_refill_relay {
	inherit = "target/relay";
	class = "idTarget_Count";
```

```
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 2;
			item[0] = "multiplayer_chainsaw_refill"; // Ammo refill
			item[1] = "multiplayer_chainsaw_refill_sfx"; // Sound effect
		}
		repeat = true; // Must be true
	}
}
}
```

```
entity {
	entityDef multiplayer_chainsaw_refill_sfx {
	inherit = "sound/soundentity";
	class = "idSoundEntity";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		temporarySoundEvent = true;
		spawnPosition = { // The spawn position does not matter
			x = 1;
```

```
		y = 1;
		z = 1;
	}
	soundOcclusionBypass = true;
	startEvents = {
		num = 1;
		item[0] = "play_pickup_ammo_chainsaw";
	}
	stopEvents = {
		num = 1;
		item[0] = "play_pickup_ammo_chainsaw";
	}
}
}
```

```
entity {
	entityDef multiplayer_chainsaw_refill {
	class = "idTarget_Command";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		commandText = "give ammo/sharedammopool/fuel"; // Be sure to set the ammo count to 1 for this
decl
		spawnPosition = { // The spawn position does not matter
			x = 1;
			y = 1;
			z = 1;
		}
	}
}
}
```

Removal Timeline / Relay

```
entity {
	entityDef coop_pickup_ammo_gas_1_fake_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true; // Must be true
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 2;
				item[0] = {
					entity = "coop_pickup_ammo_gas_1_fake_remove"; // Remove relay
					events = {
						num = 1;
						item[0] = {
							eventTime = 100;
							eventCall = {
								eventDef = "activate";
								args = {
									num = 1;
									item[0] = {
										entity = "";
									}
								}
							}
```

```
        }
    }
    item[1] = {
        entity = "multiplayer_chainsaw_refill_relay"; // SFX & give fuel relay
        events = {
            num = 1;
            item[0] = {
                eventTime = 100;
                eventCall = {
                    eventDef = "activate";
                    args = {
                        num = 1;
                        item[0] = {
                            entity = "";
                        }
                    }
                }
            }
        }
    }
    allowClientsToStart = true;
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
}
}
```

```
entity {
    entityDef coop_pickup_ammo_gas_1_fake_remove {
    inherit = "target/remove";
    class = "idTarget_Remove";
```

```
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "coop_pickup_ammo_gas_1_fake"; // Removes the chainsaw canister model
		}
	}
}
}
```

Respawn Timeline / Relay

```
entity {
	entityDef coop_pickup_ammo_gas_1_fake_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true; // Must be true
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
```

```
networkSerializeTransforms = false;
spawnPosition = {
x = 1;
y = 1;
z = 1;
}
componentTimeLine = {
entityEvents = {
num = 2;
item[0] = {
entity = "coop_pickup_ammo_gas_1_fake_relay"; // Respawn relay
events = {
num = 2;
item[0] = {
eventTime = 100;
eventCall = {
eventDef = "activate";
args = {
num = 1;
item[0] = {
entity = "";
}
}
}
}
item[1] = {
eventTime = 30000; // Show fuel cansiter render model after 30 seconds
eventCall = {
eventDef = "activate";
args = {
num = 1;
item[0] = {
entity = "";
}
}
}
}
}
}
item[1] = {
```

```
        entity = "multiplayer_chainsaw_refill_fuck_you_relay"; // SFX & give fuel relay
        events = {
            num = 1;
            item[0] = {
                eventTime = 100;
                eventCall = {
                    eventDef = "activate";
                    args = {
                        num = 1;
                        item[0] = {
                            entity = "";
                        }
                    }
                }
            }
        }
    }
    allowClientsToStart = true;
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
}
}
```

```
entity {
	entityDef coop_pickup_ammo_gas_1_fake_relay {
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false
```

```
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "coop_pickup_ammo_gas_1_fake"; // Show/Hide fuel canister render model
		}
		repeat = true; // Must be true
	}
}
}
```

Resetting Encounter Managers

> If you want Encounter Managers to reset when a round ends (all players die on either team), you will need to setup a unique exit trigger for every **idEncounterManager** entity. The **round_end** entity must target the exit trigger.

idEncounterTrigger_Exit

```
entity {
	entityDef multiplayer_reset_encounter_on_round_end {
	inherit = "encounter/trigger/exit";
	class = "idEncounterTrigger_Exit";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		spawnPosition = { // The spawn position does not matter
			x = 0;
			y = 0;
			z = 0;
		}
		triggerOnce = false; // Must be false
		resetScriptOnExit = true; // Must be true
	}
}
}
```

round_end

> This entity will always activate so long as the entityDef name is unchanged.

```
entity {
	entityDef round_end { // Do not change the entityDef name
```

```
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		count = 1;
		delay = 0;
		call = "";
		repeat = true; // Must be true
		checkLiveActivator = false;
		adjustFor16msFrames = false;
		selfActivateDelay = -1;
		waitInterval = 0;
		delayMin = -1;
		delayMax = -1;
		spawnPosition = { // The spawn position does not matter
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 2;
			item[0] = "multiplayer_reset_encounter_on_round_end"; // Exit trigger
			item[1] = "invasion_ambient_music_1_new"; // Recommended to target a new ambient music
timeline
		}
	}
}
}
```

idEncounterManager

```
entity {
	entityDef coop_encounter_manager_test_1 {
	inherit = "encounter/manager";
	class = "idEncounterManager";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		enableAIHighlightOnFinish = false;
		disabledAITypeForHighlight = "AI_MONSTER_SPECTRE AI_MONSTER_BUFF_POD AI_MONSTER_TENTACLE";
		playerMetricDef = "encounter/player_metrics";
		chargeCombatGrouping = "encounter/combat_role/charge_command";
		aiTypeDefAssignments = "actorpopulation/default/default_no_bosses";
		spawnPosition = { // The spawn position does not matter
			x = 0;
			y = 0;
			z = 0;
		}
		combatRatingScale = "COMBAT_RATING_SCALE_IGNORE";
		encounterComponent = {
			entityEvents = {
				num = 1;
				item[ 0] = {
					entity = "coop_encounter_manager_test_1"; // Must match the entityDef name
					events = {
						num = 1;
						item[ 0] = {
							eventCall = {
								eventDef = "designerComment";
								args = {
									num = 2;
									item[ 0] = {
										string = "This is a test encounter";
									}
									item[ 1] = {
										bool = true;
									}
```

```
□□□□□□□□}
□□□□□□□□}
□□□□□□□}
□□□□□□}
□□□□□}
□□□□}
□□}
□□exitTriggers = {
□□□num = 1; // Must be present in addition to the encounter's normal exit trigger
□□□item[0] = "multiplayer_reset_encounter_on_round_end";
□□}
□}
}
}
```

Music

Event Call

```
item[0] = {
	eventCall = {
		eventDef = "activateTarget";
		args = {
			num = 2;
			item[0] = {
				entity = "heavy_music_timeline_1";
			}
			item[1] = {
				string = "activate heavy music";
			}
		}
	}
}
```

Entities

```
entity {
	entityDef heavy_music_timeline_1 {
		inherit = "target/timeline";
		class = "idTarget_Timeline";
		expandInheritance = false;
		poolCount = 0;
		poolGranularity = 2;
		networkReplicated = true;
		disableAIPooling = false;
		edit = {
			flags = {
				noFlood = true;
			}
			networkSerializeTransforms = false;
			spawnPosition = {
				x = 1;
```

```
		y = 1;
		z = 1;
	}
	componentTimeLine = {
		entityEvents = {
			num = 1;
			item[0] = {
				entity = "heavy_music_relay_1";
				events = {
					num = 2;
					item[0] = {
						eventTime = 250;
						eventCall = {
							eventDef = "activate";
							args = {
								num = 1;
								item[0] = {
									entity = "";
								}
							}
						}
					}
				}
			}
		}
	}
	allowOnClients = true;
	allowClientsToStart = true;
	renderModelInfo = {
		scale = {
			x = 1;
			y = 1;
			z = 1;
		}
	}
}
}
```

```
entity {
	entityDef heavy_music_relay_1 {
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "heavy_music_1";
		}
		repeat = true;
	}
}
}
```

```
entity {
	entityDef heavy_music_1 {
	inherit = "sound/musicentity";
	class = "idMusicEntity";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
```

```
  spawnPosition = {
    x = 1;
    y = 1;
    z = 1;
  }
  initialState = "music_ghost_states/main_heavy";
  initialSwitchGroup = "music_ghost_switch";
  initialSwitchState = "doom_hunter_music";
 }
}
}
```

Doors

Door

```
entity {
	entityDef interact_doors_cultist_wide_01_1 {
	inherit = "interact/doors/cultist_wide_01";
	class = "idInteractable_Obstacle";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		soundOffset = {
			z = 1.5;
		}
		whenToSave = "SGT_CHECKPOINT";
		flags = {
			skipRenderModelReplication = true;
		}
		dormancy = {
			distance = 100;
			playerDistance = 120;
			playerRearwardDistance = 80;
		}
		fxDecl = "gameplay/invasion/demon_door";
		interaction = {
			initalState = "interactables/door/normal/locked";
			interactionGraph = "interactables/door";
			stateData = {
				num = 1;
				item[0] = {
					stateName = "interactables/door/normal/locked";
				}
			}
			transitionData = {
```

```
    num = 12;
    item[0] = {
        transitionName = "needs_key_To_unlocked_open";
        additionalWaitMs = 1000;
        playSoundOnTransition = "play_doomhunter_widedoor_open_unlocked";
    }
    item[1] = {
        transitionName = "unlocked_To_unlocked_open";
        playSoundOnTransition = "play_doomhunter_widedoor_open";
    }
    item[2] = {
        transitionName = "unlocked_open_To_unlocked";
        playSoundOnTransition = "play_doomhunter_widedoor_close";
    }
    item[3] = {
        transitionName = "locked_To_locked_open";
        playSoundOnTransition = "play_doomhunter_widedoor_open";
    }
    item[4] = {
        transitionName = "locked_open_To_locked";
        playSoundOnTransition = "play_doomhunter_widedoor_close";
    }
    item[5] = {
        transitionName = "unlocked_open_To_locked";
        playSoundOnTransition = "play_doomhunter_widedoor_close";
    }
    item[6] = {
        transitionName = "locked_To_locked_fail";
    }
    item[7] = {
        transitionName = "needs_key_To_needs_key_failed";
    }
    item[8] = {
        transitionName = "locked_To_unlocked";
    }
    item[9] = {
        transitionName = "unlocked_To_locked";
    }
    item[10] = {
```

```
        transitionName = "start_locked_To_locked";
    }
    item[11] = {
        transitionName = "locked_To_stay_open";
    }
}
animWebDecl = "animweb/interact/doors/cultist_wide_01";
}
touchData = {
retouchDelaySec = 0.5;
triggerDef = "trigger/interact/cultist_wide_01_square";
}
stateColors = {
num = 8;
item[0] = {
colorScale = 5;
materialSwapList = {
num = 2;
item[0] = {
from = "art/tile/glow/doorglow_a";
to = "art/tile/glow/doorglow_red";
}
item[1] = {
from = "art/tile/glow/doorglow_b";
to = "art/tile/glow/doorglow_red";
}
}
state = "start_locked";
}
item[1] = {
colorScale = 5;
materialSwapList = {
num = 2;
item[0] = {
from = "art/tile/glow/doorglow_a";
to = "art/tile/glow/doorglow_red";
}
item[1] = {
from = "art/tile/glow/doorglow_b";
```

```
                to = "art/tile/glow/doorglow_red";
            }
        }
        state = "activate_to_unlock";
    }
    item[2] = {
        colorScale = 5;
        materialSwapList = {
            num = 2;
            item[0] = {
                from = "art/tile/glow/doorglow_a";
                to = "art/tile/glow/doorglow_red";
            }
            item[1] = {
                from = "art/tile/glow/doorglow_b";
                to = "art/tile/glow/doorglow_red";
            }
        }
        state = "locked";
    }
    item[3] = {
        colorScale = 5;
        materialSwapList = {
            num = 2;
            item[0] = {
                from = "art/tile/glow/doorglow_a";
                to = "art/tile/glow/doorglow_red";
            }
            item[1] = {
                from = "art/tile/glow/doorglow_b";
                to = "art/tile/glow/doorglow_red";
            }
        }
        state = "needs_key";
    }
    item[4] = {
        colorScale = 5;
        materialSwapList = {
            num = 2;
```

```
        item[0] = {
            from = "art/tile/glow/doorglow_a";
            to = "art/tile/glow/doorglow_green";
        }
        item[1] = {
            from = "art/tile/glow/doorglow_b";
            to = "art/tile/glow/doorglow_blue";
        }
    }
    state = "unlocked";
}
item[5] = {
    colorScale = 5;
    materialSwapList = {
        num = 2;
        item[0] = {
            from = "art/tile/glow/doorglow_a";
            to = "art/tile/glow/doorglow_green";
        }
        item[1] = {
            from = "art/tile/glow/doorglow_b";
            to = "art/tile/glow/doorglow_blue";
        }
    }
    state = "locked_open";
}
item[6] = {
    colorScale = 5;
    materialSwapList = {
        num = 2;
        item[0] = {
            from = "art/tile/glow/doorglow_a";
            to = "art/tile/glow/doorglow_green";
        }
        item[1] = {
            from = "art/tile/glow/doorglow_b";
            to = "art/tile/glow/doorglow_blue";
        }
    }
```

```
        state = "unlocked_open";
      }
      item[7] = {
        colorScale = 5;
        materialSwapList = {
          num = 2;
          item[0] = {
            from = "art/tile/glow/doorglow_a";
            to = "art/tile/glow/doorglow_green";
          }
          item[1] = {
            from = "art/tile/glow/doorglow_b";
            to = "art/tile/glow/doorglow_blue";
          }
        }
        state = "stay_open";
      }
    }
    collisionPieces = {
      num = 4;
      item[0] = {
        meshName = "collision";
        jointName = "collision_joint";
      }
      item[1] = {
        meshName = "collision1";
        jointName = "collision1_joint";
      }
      item[2] = {
        meshName = "collision2";
        jointName = "collision2_joint";
      }
      item[3] = {
        meshName = "collision3";
        jointName = "collision3_joint";
      }
    }
    blockerBoundsExpandDists = {
      x = 0.100000001;
```

```
    y = 1;
    z = 0.100000001;
  }
  stateObstacleData = {
    num = 16;
    item[0] = {
      statePath = "interactables/door/normal/activate_to_unlock";
      obstacleReachability = true;
    }
    item[1] = {
      statePath = "interactables/door/normal/locked";
      obstacleReachability = true;
    }
    item[2] = {
      statePath = "interactables/door/normal/locked_fail";
      obstacleReachability = true;
    }
    item[3] = {
      statePath = "interactables/door/normal/locked_open";
    }
    item[4] = {
      statePath = "interactables/door/normal/needs_key";
      obstacleReachability = true;
    }
    item[5] = {
      statePath = "interactables/door/normal/needs_key_failed";
      obstacleReachability = true;
    }
    item[6] = {
      statePath = "interactables/door/normal/start_locked";
      obstacleReachability = true;
    }
    item[7] = {
      statePath = "interactables/door/normal/stay_open";
    }
    item[8] = {
      statePath = "interactables/door/normal/unlocked";
    }
    item[9] = {
```

```
        statePath = "interactables/door/normal/unlocked_open";
    }
    item[10] = {
        statePath = "interactables/door/oneWay/oneWay_closed";
        obstacleReachability = true;
    }
    item[11] = {
        statePath = "interactables/door/oneWay/oneWay_lock_fail";
        obstacleReachability = true;
    }
    item[12] = {
        statePath = "interactables/door/oneWay/oneWay_locked";
        obstacleReachability = true;
    }
    item[13] = {
        statePath = "interactables/door/oneWay/oneWay_open";
    }
    item[14] = {
        statePath = "interactables/door/oneWay/oneWay_open_fail";
        obstacleReachability = true;
    }
    item[15] = {
        statePath = "interactables/door/oneWay/oneWay_opened";
    }
  }
  dissolveDistance = 20;
  renderModelInfo = {
    model = "md6def/objects/doors/cultist_wide_door_01/cultist_wide_door_01.md6";
  }
  guiData = {
    swfFile = "swf/worldgui/guientity/1_interact_lock_default.swf";
    scale = 0.0850000009;
    initialCommands = {
      num = 1;
      item[0] = {
        commandType = "SPRITE_HIDE";
      }
    }
    positionOffset = {
```

```
		x = 8;
		z = 25;
	}
}
	umbraVisBlocking = "CAN_BLOCK";
	umbraSoundBlocking = "CAN_BLOCK";
	demonPlayerRenderModel = "art/objects/door/demon/cultist_wideDoor_01_demon.lwo";
	spawnPosition = {
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	demonPlayersPassThrough = false;
}
}
}
```

Open

```
entity {
	entityDef interact_doors_cultist_wide_01_1_open {
	inherit = "target/interact_action_door";
	class = "idTarget_InteractionAction";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		renderModelInfo = {
			materialRemap = {
				num = 1;
				item[0] = {
					to = "art/tile/common/nodraw";
				}
			}
		}
	}
	actionFilter = "interactionAction/door";
```

```
		spawnPosition = {
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		targets = {
			num = 1;
			item[0] = "interact_doors_cultist_wide_01_1";
		}
		action = "IA_DOOR_OPEN";
	}
	}
	}
```

Replication

```
entity {
	entityDef interact_doors_cultist_wide_01_1_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 2;
				item[0] = {
```

```
        entity = "interact_doors_cultist_wide_01_1_relay";
        events = {
          num = 1;
          item[0] = {
            eventTime = 250;
            eventCall = {
              eventDef = "activate";
              args = {
                num = 1;
                item[0] = {
                  entity = "";
                }
              }
            }
          }
        }
      }
      item[1] = {
        entity = "interact_doors_cultist_wide_01_1_remove";
        events = {
          num = 1;
          item[0] = {
            eventTime = 2500;
            eventCall = {
              eventDef = "activate";
              args = {
                num = 1;
                item[0] = {
                  entity = "";
                }
              }
            }
          }
        }
      }
    }
    allowClientsToStart = true;
    renderModelInfo = {
      scale = {
```

```
		x = 1;
		y = 1;
		z = 1;
	}
}
}
}
```

```
entity {
	entityDef interact_doors_cultist_wide_01_1_relay {
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "interact_doors_cultist_wide_01_1_open";
		}
		repeat = true;
	}
}
}
```

```
entity {
	entityDef interact_doors_cultist_wide_01_1_remove {
	inherit = "target/remove";
	class = "idTarget_Remove";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "interact_doors_cultist_wide_01_1";
		}
		repeat = true;
	}
}
}
```

Shootables

Shootable

Disable network replication

```
entity {
	entityDef funky_shootable_1 {
	inherit = "func/shootable";
	class = "idTrigger_TakeDamage";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		renderModelInfo = {
			model = "art/kit/gameplay/breakable_glow_b.lwo";
			scale = {
				x = 0.25;
				y = 0.25;
				z = 0.25;
			}
		}
		clipModelInfo = {
			type = "CLIPMODEL_BOX";
			size = {
				x = 1.5;
				y = 1.5;
				z = 1.5;
			}
		}
		spawnPosition = {
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
```

```
	spawnOrientation = {
		mat = {
			mat[0] = {
				x = 0;
				y = -1;
				z = 0;
			}
			mat[1] = {
				x = 1;
				y = 0;
				z = 0;
			}
		}
	}
	wait = 0;
	triggerOnce = true;
	triggerForAllClients = true;
	playerCanActivate = false;
	deadAiCanActivate = false;
	allowZeroDamage = true;
	soundActivated = "play_shootable_activate";
	soundReady = "play_shootable_reset";
	fxDecl = "gameplay/shootable";
	weaponDamageType = "PLAYER_WEAPON_SHOTGUN_ PLAYER_WEAPON_SHOTGUN_FULL_AUTO_
PLAYER_WEAPON_SHOTGUN_FULL_AUTO_MASTERY_ PLAYER_WEAPON_SHOTGUN_STICKY_BOMB_
PLAYER_WEAPON_DOUBLE_BARRELED_SHOTGUN_ PLAYER_WEAPON_PLASMA_ PLAYER_WEAPON_PLASMA_HEATWAVE_
PLAYER_WEAPON_PLASMA_SUPERCHARGE_ PLAYER_WEAPON_PLASMA_ICE_BOMB_ PLAYER_WEAPON_HAR_
PLAYER_WEAPON_HAR_SCOPE_ PLAYER_WEAPON_HAR_MICROMISSLE_ PLAYER_WEAPON_ROCKETLAUNCHER_
PLAYER_WEAPON_ROCKETLAUNCHER_DETONATE_ PLAYER_WEAPON_ROCKETLAUNCHER_LOCKON_
PLAYER_WEAPON_BALLISTA_ PLAYER_WEAPON_BALLISTA_ARBALEST_ PLAYER_WEAPON_BALLISTA_DESTROYER_
PLAYER_WEAPON_CHAINGUN_ PLAYER_WEAPON_CHAINGUN_TURRET_ PLAYER_WEAPON_CHAINGUN_GATLING_
PLAYER_WEAPON_MEAT_HOOK_NAPALM_ PLAYER_WEAPON_BFG_ PLAYER_WEAPON_FRAG_
PLAYER_WEAPON_FRAG_GRENADE_ PLAYER_WEAPON_FRAG_GRENADE_CLUSTER_ PLAYER_WEAPON_FLAME_BELCH_
PLAYER_WEAPON_FLAME_BELCH_NAPALM_ PLAYER_WEAPON_PISTOL_ PLAYER_WEAPON_PISTOL_HAND_CANNON_
PLAYER_WEAPON_UNMAYKR_ PLAYER_WEAPON_MICROWAVE_MOD_";
	targets = {
		num = 1;
		item[0] = "funky_shootable_1_timeline";
	}
```

```
	dormancy = {
		allowDormancy = false;
		allowPvsDormancy = false;
	}
	keepAfterTriggerOnce = true;
}
}
}
```

Replication

```
entity {
	entityDef funky_shootable_1_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 2;
				item[0] = {
					entity = "funky_shootable_1_relay";
					events = {
						num = 1;
						item[0] = {
							eventTime = 250;
```

```
                eventCall = {
                    eventDef = "activate";
                    args = {
                        num = 1;
                        item[0] = {
                            entity = "";
                        }
                    }
                }
            }
        }
        item[1] = {
            entity = "funky_shootable_1_remove";
            events = {
                num = 1;
                item[0] = {
                    eventTime = 250;
                    eventCall = {
                        eventDef = "activate";
                        args = {
                            num = 1;
                            item[0] = {
                                entity = "";
                            }
                        }
                    }
                }
            }
        }
    }
    allowClientsToStart = true;
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
```

```
	}
}
}
```

```
entity {
	entityDef funky_shootable_1_relay {
	inherit = "target/relay";
	class = "idTarget_Count";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "something_to_activate";
		}
		dormancy = {
			allowPvsDormancy = false;
		}
		delay = 0;
	}
}
}
```

```
entity {
	entityDef funky_shootable_1_remove {
	inherit = "target/remove";
```

```
	class = "idTarget_Remove";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		count = 1;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		targets = {
			num = 1;
			item[0] = "funky_shootable_1";
		}
		dormancy = {
			allowPvsDormancy = false;
		}
		delay = 0;
	}
}
}
```

Breakable Walls

Breakable Wall

```
entity {
	entityDef destructible_mall_wall_b_1 {
	inherit = "destructible/e2m1_nest/mall_wall_b";
	class = "idDestructible";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		whenToSave = "SGT_CHECKPOINT";
		flags = {
			skipRenderModelReplication = true;
		}
		soundOffset = {
			z = 1.5;
		}
		renderModelInfo = {
			emissiveColor = {
				r = 0;
				b = 0.0470587984;
			}
			emissiveScale = 6;
			model = "art/kit/hell_earth/walls/crack_mall_b_whole.lwo";
			acceptsDecals = true;
		}
		clipModelInfo = {
			type = "CLIPMODEL_BOX";
			size = {
				x = 7.19999981;
				y = 1;
				z = 7.0999999;
			}
```

```
		numSides = 6;
	}
	soundOcclusionBypass = true;
	targetingDecl = NULL;
	fxDecl = "gameplay/invasion/demon_door";
	destructible = {
		idleCommands = {
			num = 1;
			item[0] = {
				time = 10000;
				command = "IDLE_COMMAND_BECOME_STATIC";
			}
		}
		decl = "destructible/e2m1_nest/mall_wall_b";
	}
	effectiveDamageTypes = {
		num = 1;
		item[0] = "damage/special/no_damage";
	}
	meleeOnDash = false;
	dissolveDistance = 20;
	demonPlayerRenderModel = "art/kit/hell_earth/walls/crack_mall_a_demon.lwo";
	umbraVisBlocking = "CAN_BLOCK";
	umbraSoundBlocking = "CAN_BLOCK";
	spawnPosition = {
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	dormancy = {
		playerDistance = 20;
		playerRearwardDistance = 20;
		allowDistanceDormancy = false;
		allowDormancy = true;
		allowPvsDormancy = false;
	}
	targets = {
		num = 1;
		item[0] = "destructible_mall_wall_b_1_timeline";
```

```
	}
	demonPlayersPass Through = false;
	}
}
}
```

Replication

```
entity {
	entityDef destructible_mall_wall_b_1_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 1;
				item[0] = {
					entity = "destructible_mall_wall_b_1_relay";
					events = {
						num = 1;
						item[0] = {
							eventTime = 250;
							eventCall = {
								eventDef = "activate";
								args = {
```

```
                num = 1;
                item[0] = {
                    entity = "";
                }
            }
        }
    }
    allowClientsToStart = true;
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
}
}
```

```
entity {
    entityDef destructible_mall_wall_b_1_relay {
    inherit = "target/relay";
    class = "idTarget_Count";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        flags = {
            noFlood = true;
        }
        networkSerializeTransforms = false;
        count = 1;
        spawnPosition = {
```

```
    x = 1;
    y = 1;
    z = 1;
  }
  targets = {
    num = 1;
    item[0] = "destructible_mall_wall_b_1";
  }
  repeat = false;
 }
}
}
```

Bounce Pads

Dynamic Entity (Backer)

```
entity {
	entityDef invasion_bouncepad_back_1 {
	inherit = "func/dynamic";
	class = "idDynamicEntity";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		spawnPosition = { // Set the position roughly -0.2 (on the z axis) behind the emitter FX
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = { // Backer should be laying on the ground
			mat = {
				mat[0] = {
					x = 0.707107;
					y = -0.707107;
					z = 0;
				}
				mat[1] = {
					x = 0.707107;
					y = 0.707107;
					z = 0;
				}
				mat[2] = {
					x = 0;
					y = 0;
					z = 1;
				}
			}
```

```
	}
	renderModelInfo = {
		model = "art/kit/hell/bridge/bridge_archPillar_a_divider.lwo";
		scale = { // Do not change the scale, it will break collision
			x = 0.66;
			y = 0.66;
			z = 0.5;
		}
	}
	clipModelInfo = {
		clipModelName = "art/kit/hell/bridge/bridge_archPillar_a_divider.lwo"; // Adds collision
	}
}
}
}
```

Emitter FX (Slayer Team)

```
entity {
	layers {
		"game/pvp/slayer_team" // Only the Slayer can see this FX
	}
	entityDef invasion_bouncepad_fx_slayer_1 {
	inherit = "func/fx";
	class = "idEntityFx";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			canBecomeDormant = true;
		}
		dormancy = {
			delay = 5;
			distance = 78.029007;
		}
```

```
	spawnPosition = { // Position should be identical to the FX on the demon team
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	fxEffect = "gameplay/invasion/bounce_pad_slayer"; // Emitter is red
	spawnOrientation = { // Orientation should not be changed
		mat = {
			mat[0] = {
				x = 1;
				y = 0;
				z = 0;
			}
			mat[1] = {
				x = 0;
				y = 1;
				z = 0;
			}
			mat[2] = {
				x = 0;
				y = 0;
				z = 1;
			}
		}
	}
	renderModelInfo = {
		scale = {
			x = 1;
			y = 1;
			z = 1;
		}
	}
}
}
```

Emitter FX (Demon Team)

```
entity {
	layers {
		'game/pvp/demon_team" // Only Demon Players can see this FX
	}
	entityDef invasion_bouncepad_fx_slayer_1 {
	inherit = "func/fx";
	class = "idEntityFx";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			canBecomeDormant = true;
		}
		dormancy = {
			delay = 5;
			distance = 78.029007;
		}
		spawnPosition = { // Position should be identical to the FX on the slayer team
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
        automapPropertiesDecl = "invasion_bounce_demon";
		fxEffect = "gameplay/invasion/bounce_pad_demon"; // Emitter is purple
		spawnOrientation = { // Orientation should not be changed
			mat = {
				mat[0] = {
					x = 1;
					y = 0;
					z = 0;
				}
				mat[1] = {
					x = 0;
					y = 1;
					z = 0;
```

```
        }
        mat[2] = {
            x = 0;
            y = 0;
            z = 1;
        }
    }
  }
  renderModelInfo = {
    scale = {
        x = 1;
        y = 1;
        z = 1;
    }
  }
}
}
}
```

Bouncepad

```
entity {
 entityDef invasion_bouncepad_from_1 {
 inherit = "trigger/bounce_pad";
 class = "idTrigger_BouncePad";
 expandInheritance = false;
 poolCount = 0;
 poolGranularity = 2;
 networkReplicated = false;
 disableAIPooling = false;
 edit = {
  flags = {
    canBecomeDormant = true;
  }
  launchFX = "fx/bounce_pad/basic";
  spawnPosition = { // Position should be roughtly +1 (z axis) above the FX
    x = -11.11;
    y = -11.11;
```

```
		z = -11.11;
	}
	renderModelInfo = {
		model = NULL;
		scale = {
			x = 1;
			y = 1;
			z = 1;
		}
	}
	clipModelInfo = {
		clipModelName = "maps/game/sp/e1m3_cult/e1m3_cult/invasion_x_bouncepad_from_1";
		size = {
			x = 1.5;
			y = 1.5;
			z = 1;
		}
	}
	playerCanActivate = false; // Slayer cannot activate
	demonPlayerCanActivate = true; // Demon players can activate
	destination = "invasion_bouncepad_dest_1"; // Fly to this destination when activated
	launchSpeed = 32; // Speed demons will fly upwards
}
}
```

Destination

```
entity {
	layers {
		"game/pvp/demon_team" // Only demon players can be launched
	}
	entityDef invasion_bouncepad_dest_1 {
	inherit = "info/bounce_destination";
	class = "idInfo_BounceDestination";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
```

```
disableAIPooling = false;
edit = {
spawnPosition = { // Increase z axis position close to the destination
x = -11.11;
y = -11.11;
z = -11.11;
}
spawnOrientation = { // Orientation does not matter
mat = {
mat[0] = {
x = 1;
y = 0;
z = 0;
}
mat[1] = {
x = 0;
y = 1;
z = 0;
}
mat[2] = {
x = 0;
y = 0;
z = 1;
}
}
}
renderModelInfo = {
scale = {
x = 1;
y = 1;
z = 1;
}
}
}
}
}
```

Teleporters

Timeline

> The timeline is only required for teleporter cooldowns.

```
entity {
	layers {
		"game/pvp/demon_team" // Applies only to Demon Players
	}
	entityDef invasion_portal_timeline_1 {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false so cooldown is not shared globally
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 2;
				item[0] = {
					entity = "invasion_portal_fx_demon_1";
					events = {
						num = 2;
						item[0] = {
```

```
        eventCall = {
            eventDef = "hide"; // Hide 1st effect upon activation
            args = {
                num = 0;
            }
        }
    }
    item[1] = {
        eventTime = 5000;
        eventCall = {
            eventDef = "show"; // Show 1st effect after 5 seconds
            args = {
                num = 0;
            }
        }
    }
}
item[1] = {
    entity = "invasion_portal_fx_demon_2";
    events = {
        num = 2;
        item[0] = {
            eventCall = {
                eventDef = "hide"; // Hide 2nd effect upon activation
                args = {
                    num = 0;
                }
            }
        }
        item[1] = {
            eventTime = 5000;
            eventCall = {
                eventDef = "show"; // Show 2nd effect after 5 seconds
                args = {
                    num = 0;
                }
            }
        }
    }
```

```
		}
	}
	}
	allowClientsToStart = true; // Must be true
	shouldForceTimelineFinish = true; // Must be true
	}
}
}
```

Dynamic Entity (Backer)

```
entity {
	entityDef invasion_portal_back_1 {
	inherit = "func/dynamic";
	class = "idDynamicEntity";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		spawnPosition = { // Set the spawn position roughly -0.2 (on the x or y axis) behind the
portal FX
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = { // Backer should be standing upright
			mat = {
				mat[0] = {
					x = 0;
					y = 0;
					z = -1;
				}
				mat[1] = {
					x = -1;
					y = 0;
```

```
        z = 0;
    }
    mat[2] = {
        x = 0;
        y = 1;
        z = 0;
    }
  }
  renderModelInfo = {
    model = "art/kit/hell/bridge/bridge_archPillar_a_divider.lwo";
    scale = { // Do not change the scale, it will break collision
      x = 0.66;
      y = 0.66;
      z = 0.5;
    }
  }
  clipModelInfo = {
    clipModelName = "art/kit/hell/bridge/bridge_archPillar_a_divider.lwo"; // Adds collision
  }
 }
}
}
```

Portal FX (Slayer Team)

```
entity {
 layers {
  'game/pvp/slayer_team" // Only the Slayer can see this FX
 }
 entityDef invasion_portal_fx_slayer_1 {
 inherit = "func/fx";
 class = "idEntityFx";
 expandInheritance = false;
 poolCount = 0;
 poolGranularity = 2;
 networkReplicated = false;
 disableAIPooling = false;
```

```
edit = {
	flags = {
		canBecomeDormant = true;
	}
	dormancy = {
		delay = 5;
		distance = 78.029007;
	}
	spawnPosition = { // Position should be identical to the FX on the demon team
		x = -11.11;
		y = -11.11;
		z = -11.11;
	}
	spawnOrientation = { // Orientation should be identical to the FX on the demon team
		mat = {
			mat[0] = {
				x = 0;
				y = 1;
				z = 0;
			}
			mat[1] = {
				x = -1;
				y = 0;
				z = 0;
			}
			mat[2] = {
				x = 0;
				y = 0;
				z = 1;
			}
		}
	}
	renderModelInfo = {
		scale = {
			x = 1;
			y = 1;
			z = 1;
		}
	}
```

```
		flags = {
			hide = false;
		}
		fxEffect = "gameplay/invasion/teleporter_flat_demon"; // FX is red
	}
}
}
```

Portal FX (Demon Team)

```
entity {
	layers {
		"game/pvp/demon_team" // Only Demon Players can see this FX
	}
	entityDef invasion_portal_fx_demon_1 {
	inherit = "func/fx";
	class = "idEntityFx";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			canBecomeDormant = true;
		}
		dormancy = {
			delay = 5;
			distance = 78.029007;
		}
		spawnPosition = { // Position should be identical to the FX on the slayer team
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = { // Orientation should be identical to the FX on the slayer team
			mat = {
				mat[0] = {
```

```
                x = 0;
                y = 1;
                z = 0;
            }
            mat[1] = {
                x = -1;
                y = 0;
                z = 0;
            }
            mat[2] = {
                x = 0;
                y = 0;
                z = 1;
            }
        }
    }
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
    flags = {
        hide = false;
    }
    fxEffect = "gameplay/invasion/teleporter_flat_slayer"; // FX is blue
  }
}
}
```

Teleporter

```
entity {
  layers {
    "game/pvp/demon_team" // Only demon players can use the teleporter
  }
  entityDef invasion_portal_from_1 {
```

```
	inherit = "trigger/teleporter";
	class = "idTrigger_Teleporter";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false; // Must be false so cooldown is not shared globally
	disableAIPooling = false;
	edit = {
		clientCanActivate = true; // Must be true
		fxExtraCondition = "FX_EXTRA_COND_MAX";
		sounds = {
			activated = "play_arena_teleporter_use";
		}
		spawnPosition = { // Position should be close to the idDynamicEntity backer
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		renderModelInfo = {
			model = NULL;
		}
		clipModelInfo = {
			type = "CLIPMODEL_CYLINDER";
			size = { // Scale based on portal size
				x = 1;
				y = 3;
				z = 2;
			}
			clipModelName = NULL;
		}
		targets = {
			num = 1;
			item[0] = "invasion_portal_timeline_1"; // Activate the timeline to show and hide FX
		}
		dormancy = {
			playerDistance = 6;
			playerRearwardDistance = 6;
		}
		flags = {
```

```
		hide = false;
	}
	wait = 5; // Deactivate both teleporters for 5 seconds
	playerCanActivate = false; // Slayer cannot activate
	demonPlayerCanActivate = true; // Demon players can activate
	otherTeleporter = "invasion_portal_from_2"; // Deactivate this teleporter
	destination = "invasion_portal_dest_1"; // Go to this destination when activated
}
}
}
```

Destination

```
entity {
	entityDef invasion_portal_dest_1 {
	inherit = "info/teleport_destination";
	class = "idInfo_TeleportDestination";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		spawnPosition = { // Position should be just foward of the opposing idTrigger_Teleporter
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = { // Orientation should be identical to the opposing idEntityFx
			mat = {
				mat[0] = {
					x = 0;
					y = 1;
					z = 0;
				}
				mat[1] = {
					x = -1;
					y = 0;
					z = 0;
```

```
□□□□}
□□□□mat[2] = {
□□□□□x = 0;
□□□□□y = 0;
□□□□□z = 1;
□□□□}
□□□}
□□}
□}
}
}
```

Triggers

Trigger

```
entity {
	entityDef trigger_volume_1 {
	inherit = "trigger/trigger";
	class = "idTrigger";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		demonPlayerCanActivate = true;
		removeFlag = "RMV_CHECKPOINT_ALLOW_MS";
		spawnPosition = {
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		renderModelInfo = {
			model = NULL;
			scale = {
				x = 1;
				y = 1;
				z = 1;
			}
		}
		clipModelInfo = {
			type = "CLIPMODEL_BOX";
			size = {
				x = 5;
				y = 5;
				z = 5;
			}
		}
```

```
	targets = {
		num = 1;
		item[0] = "trigger_volume_1_timeline";
	}
	wait = 0;
	triggerOnce = true;
}
}
}
```

Replication

```
entity {
	entityDef trigger_volume_1_timeline {
	inherit = "target/timeline";
	class = "idTarget_Timeline";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = true;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		networkSerializeTransforms = false;
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
		componentTimeLine = {
			entityEvents = {
				num = 1;
				item[0] = {
					entity = "trigger_volume_1_relay";
					events = {
						num = 1;
						item[0] = {
```

```
        eventTime = 250;
        eventCall = {
            eventDef = "activate";
            args = {
                num = 1;
                item[0] = {
                    entity = "";
                }
            }
        }
    }
    allowClientsToStart = true;
    renderModelInfo = {
        scale = {
            x = 1;
            y = 1;
            z = 1;
        }
    }
}
}
```

```
entity {
    entityDef trigger_volume_1_relay {
    inherit = "target/relay";
    class = "idTarget_Count";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false;
    disableAIPooling = false;
    edit = {
        flags = {
            noFlood = true;
```

```
    }
    networkSerializeTransforms = false;
    count = 1;
    spawnPosition = {
        x = 1;
        y = 1;
        z = 1;
    }
    targets = {
        num = 1;
        item[0] = "something_to_activate";
    }
    repeat = true;
}
}
}
```

Gorilla Bars

Entity

> Disable network replication.

```
entity {
	entityDef game_interact_vault_pipe_1 {
	inherit = "interact/vault/pipe";
	class = "idTrigger_GorillaBar";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		removeFlag = "RMV_CHECKPOINT_ALLOW_MS";
		renderModelInfo = {
			model = "art/test/monkey_bar_a.lwo";
		}
		netRelevancyFlags = "";
		triggerOnce = false;
		triggerOnEnter = true;
		testForValidGrab = true;
		spawnPosition = {
			x = -11.11;
			y = -11.11;
			z = -11.11;
		}
		spawnOrientation = {
			mat = {
				mat[0] = {
					x = 0;
					y = 1;
				}
				mat[1] = {
					x = -1;
```

```
        y = 0;
    }
}
  clipModelInfo = {
    clipModelName = "art/test/monkey_bar_a.lwo";
  }
  animOffsetAdjust = {
    x = -1;
  }
}
}
```

Startup Commands

Add this to "shell.entities" and trigger with a logic entity

```
entity {
	entityDef mapstart_commands {
	class = "idTarget_Command";
	expandInheritance = false;
	poolCount = 0;
	poolGranularity = 2;
	networkReplicated = false;
	disableAIPooling = false;
	edit = {
		flags = {
			noFlood = true;
		}
		commandText = "pvp_lobby_showFullMapList 1; net_gameserver_enable 0; net_maxRate 100;
g_havokHeapLimitMB 200; g_damageScaleAItoDP 4; g_damageScaleSlayertoSlayer 0;
g_damageScaleSlayertoDP 0; g_damageScaleDPtoSlayer 0; pm_disableWalkingOnTheEdgeOfLedge 0;
warning_disable general; warning_disable anim";
		spawnPosition = {
			x = 1;
			y = 1;
			z = 1;
		}
	}
}
}
```

Command details

- **pvp_lobby_showFullMapList 1** - Shows entire map list if non PVP maps are added
- **net_gameserver_enable 0** - Enables local hosting
- **net_maxRate 100** - Reduces likelihood of crashes
- **g_havokHeapLimitMB 200** - Reduces likelihood of crashes

- **g_damageScaleAItoDP 4** - Multiplies AI damage to Demon Players by x4 (x2 more than Nightmare difficulty, remove for Invasion/SVS)
- **g_damageScaleSlayertoSlayer 0** - Slayers cannot damage other Slayers (remove for Invasion/SVS)
- **g_damageScaleSlayertoDP 0** - Slayers cannot damage other Demon Players (remove for Invasion/SVS)
- **g_damageScaleDPtoSlayer 0** - Demon Players cannot damage other Slayers (remove for Invasion/SVS)
- **pm_disableWalkingOnTheEdgeOfLedge 0** - Movement is no longer negated when moving off a ledge
- **warning_disable general** - Removes warning spam in the console
- **warning_disable anim** - Removes warning spam in the console