

- 1) **int** A signed 32-bit integer.
- 2) **float** A 32-bit floating point number
- 3) **string** A string of arbitrary length, terminated by a NULL character.
- 4) **object** An integer that represents a particular object in the world.
- 5) **vector** A structure of three floats, useful for defining positions and orientation vectors.
To access the components of the vector, you can use the dot operator to get the x, y and z components of the vector.
A sample of how to do this is given below.

```
vector v = GetPosition(OBJECT_SELF) - GetPosition(oidTargetObject)
fDistanceToTarget = sqrt(v.x * v.x + v.y * v.y + v.z * v.z);
vector v = Vector(fX, fY, fZ); // Construct a vector
```

- 6) **command** A void-returning function, whether it is user-defined or engine-defined.
This is used by the [DelayCommand\(\)](#) and [AssignCommand\(\)](#) functions.

Mark B: "[Command](#)" isn't really a variable. In the current implementation, a command parameter takes the state of the running script and saves it for execution by another object (in the case of [AssignCommand](#)) or at a later time (in the case of [DelayCommand](#)).

However, it is on my list of things I am considering adding to the language (time permitting, and if it can be done without breaking the scripts that the designers have already done!)

One of my concerns would be when the parameters are evaluated. Does [OBJECT_SELF](#) always refer to the object that created the command, or should it refer to the object that is eventually running the command? And if you assign a command to someone, should it check their locals, or your locals. What about variables that fall out-of-scope between the evaluation and their execution?

- 7) **event** All of the "events" that run scripts within the scripting language have their own "event" creation functions, which you can give to any object. So, if you want to tell an object (oidTarget) to behave as if they have been attacked by the object running the script, you would write:

```
event evAttacked = EventOnAttacked(OBJECT_SELF);
SignalEvent(oidTarget, evAttacked);
```

- 8) **effect** Each effect that you're allowed to place on an object actually has a constructor within the scripting language that you are then allowed to use in [ApplyEffectToObject\(\)](#).

For example:

```
effect eConPenalty = EffectAbilityDecrease(ABILITY_CONSTITUTION, 1);
ApplyEffectToObject(DURATION_TYPE_PERMANENT, eConPenalty, oTargetPC);
```

- 9) **location** A location has three components, the object ID of the area, a vector representing the position within that area, and a floating point number representing the facing.

```
location loc = Location(objArea, vecPosition, fFacing); // Construct a location
```

So, instead of a rather complicated function like this:

```
JumpToPoint(object oidArea, vector vPosition, float fFacing);
```

we now have:

```
JumpToLocation(location locJumpTo);
```

- 10) **talent** A talent represents a feat, skill, or spell:

```
talent tHeavyArmor = TalentFeat(FEAT_ARMOR_PROFICIENCY_HEAVY);
talent tSkillHide = TalentSkill(SKILL_HIDE);

talent tSpellCurse = TalentSpell(SPELL_BESTOW_CURSE);
talent tTurnUndead = TalentSpell(SPELLABILITY_TURN_UNDEAD);
```

CREATURE EVENTS	MODULE EVENTS	AREA EVENTS	PLACEABLE OBJECT EVENTS
OnHeartbeat OnPerception OnSpellCastAt OnPhysicalAttacked OnDamaged OnDisturbed OnCombatRoundEnd OnConversation OnSpawn OnRested OnDeath OnBlocked OnUserDefined	OnAcquireItem OnActivateItem OnClientEnter OnClientLeave OnHeartbeat OnModuleLoad OnPlayerDeath OnPlayerDying OnPlayerRespawn OnPlayerRest OnUnAcquireItem OnUserDefined	OnEnter OnExit OnHeartbeat OnUserDefined	OnClosed OnDamaged OnDeath OnHeartbeat OnDisturbed OnLock OnPhysicalAttacked OnOpen OnSpellCastAt OnUnlock OnUsed OnUserDefined

TRIGGER EVENTS	DOOR EVENTS	ENCOUNTER EVENTS
OnClick OnEnter OnExit OnHeartbeat OnUserDefined	OnAreaTransitionClick OnClose OnDamaged OnDeath OnFailToOpen OnHeartbeat OnLock OnPhysicalAttacked OnOpen() OnSpellCastAt OnUnlock OnUserDefined	OnEnter OnExhausted OnExit OnHeartbeat OnUserDefined

Events

Scripts are written to respond to specific events provided by the game engine.

Let's say we want to have a special type of lizardman creature that upon their death causes a lightning bolt to strike the last character that attacked it. We'll need a script -- we'll call it "DeathStrikeLBolt.nss" and we'll attach that to the "Creature Event" called "OnDeath." Then, anytime one of our "special" lizardmen is killed, you don't want to be the last one to hit him.

```
void main() // DeathStrikeLBolt.nss
{
    object oAttacker = GetLastAttacker();
    ActionCastSpellAtObject(SPELL_LIGHTNING_BOLT, oAttacker);
}
```

Couldn't really be much simpler, could it? 😊

Suppose we wanted to have a king's throne, and if a Dispel Magic is cast on the throne, we want it to turn into a Treasure Chest.

Let's create a script called "ThroneToTreasure.nss." We'd then attach it to the "Placeable Object Event" called "OnSpellCastAt."

```
void main() // ThroneToTreasure.nss
{
    int nSpell = GetLastSpell();
    if (nSpell == SPELL_DISPEL_MAGIC)
    {
        location loc = GetLocation(OBJECT_SELF);
        DestroyObject(OBJECT_SELF); // Destroy the throne

        ActionDoCommand( PlayVisualAreaEffect(location, VFX_INP_POLYMORPH_S) ); // Let's add a little polymorph visual effect

        ActionDoCommand( CreateChest(loc) ); // Create the treasure chest in it's place.
    }
}

// This is simply because CreateObject returns an Object and therefore can't be used directly in an ActionDoCommand() statement.
void CreateChest(location loc)
{
    CreateObject(OBJECT_TYPE_ITEM, "THRONE_CHEST", loc);
}
```

TRUE, FALSE**ABILITY_**

CHARISMA, CONSTITUTION, DEXTERITY,
INTELLIGENCE, STRENGTH, WISDOM

AC_

ARMOUR_ENCHANTMENT_BONUS,
DEFLECTION_BONUS, DODGE_BONUS,
NATURAL_BONUS, SHIELD_ENCHANTMENT_BONUS

ACTION_

ANIMALEMPATHY, ATTACKOBJECT, CASTSPELL,
CLOSEDOOR, COUNTERSPELL, DIALOGOBJECT,
DISABLETRAP, DROPITEM, EXAMINETRAP, FLAGTRAP,
FOLLOW, HEAL, INVALID, ITEMCASTSPELL, LOCK,
MOVETOPOINT, OPENDOOR, OPENLOCK, PICKPOCKET,
PICKUPITEM, RECOVERTRAP, REST, SETTRAP, SIT,
TAUNT, USEOBJECT, WAIT

ALIGNMENT_

ALL, CHAOTIC, EVIL, GOOD, LAWFUL, NEUTRAL

ANIMAL_COMPANION_CREATURE_TYPE_

BADGER, BEAR, BOAR, DIREWOLF, HAWK, NONE,
PANTHER, SPIDER, WOLF

AOE_MOB_

BLINDING, CIRCCHAOS, CIRCEVIL, CIRCGOOD,
CIRCLAW, DRAGON_FEAR, ELECTRICAL, FEAR, FIRE,
FROST, INVISIBILITY_PURGE, MENACE, PROTECTION,
SILENCE, STUN, TYRANT_FOG, UNEARTHLY,
UNNATURAL

AOE_PER_

CREEPING_DOOM, DARKNESS,
DELAY_BLAST_FIREBALL, ENTANGLE,
EVARDS_BLACK_TENTACLES, FOGACID, FOGFIRE,
FOGGHOUL, FOGKILL, FOGMIND, FOGSTINK, GREASE,
INVIS_SPHERE, STORM, WALLBLADE, WALLFIRE,
WALLWIND, WEB

AREA_TRANSITION_

CITY, CRYPT, FOREST, RANDOM, RURAL,
USER_DEFINED

ASSOCIATE_COMMAND_

ATTACKNEAREST, FOLLOWMASTER, GUARDMASTER,
HEALMASTER, LEAVEPARTY, MASTERATTACKEDOTHER,
MASTERFAILEDLOCKPICK,
MASTERGOINGTOBEATTACKED, MASTERSAWTRAP,
MASTERUNDERATTACK, RELEASEDOMINATION,
STANDGROUND, UNPOSSESSFAMILIAR,
UNSUMMONANIMALCOMPANION, UNSUMMONFAMILIAR,
UNSUMMONSUMMONED

ASSOCIATE_TYPE_

ANIMALCOMPANION, DOMINATED, FAMILIAR,
HENCHMAN, SUMMONED

ATTACK_BONUS_

MISC, OFFHAND, ONHAND

ATTITUDE_

AGGRESSIVE, DEFENSIVE, NEUTRAL, SPECIAL

BASE_ITEM_

AMULET, ARMOR, ARROW, BASTARDSWORD,
BATTLEAXE, BELT, BOLT, BOOK, BOOTS, BRACER,
BULLET, CBLUDGWEAAPON, CLOAK, CLUB,
CPIERCWEAPON, CREATUUREITEM, CSLASHWEAPON,
CSLSHPRCWEAP, DAGGER, DART, DIREMACE,
DOUBLEAXE, GEM, GLOVES, GOLD, GREATAXE,
GREATSWORD, HALBERD, HANDAXE, HEALERSKIT,
HEAVYCROSSBOW, HEAVYFLAIL, HELMET, INVALID,
KAMA, KATANA, KEY, KUKRI, LARGEBOX, LARGESHIELD,
LIGHTCROSSBOW, LIGHTFLAIL, LIGHTHAMMER,
LIGHTMACE, LONGBOW, LONGSWORD, MAGICROD,
MAGICSTAFF, MAGICWAND, MISCLARGE, MISCMEDIUM,
MISCSMALL, MISCTALL, MISCTHIN, MISCSWIDE,
MORNINGSTAR, POTIONS, QUARTERSTAFF, RAPIER,
RING, SCIMITAR, SCROLL, SCYTHE, SHORTBOW,
SHORTSPEAR, SHORTSWORD, SHURIKEN, SICKLE,
SLING, SMALLSHIELD, SPELLSCROLL, THIEVESTOOLS,
THROWINGAXE, TORCH, TOWERSHIELD, TRAPKIT,
TWOBLADEDWORD, WARHAMMER

BODY_NODE_

CHEST, HAND

CAMERA_MODE_

CHASE_CAMERA, STIFF_CHASE_CAMERA, TOP_DOWN

CLASS_TYPE_

ABERRATION, ANIMAL, BARBARIAN, BARD, BEAST,
CLERIC, COMMONER, CONSTRUCT, DRAGON, DRUID,
ELEMENTAL, FEY, FIGHTER, GIANT, HUMANOID,
INVALID, MAGICAL_BEAST, MONK, MONSTROUS,
OUTSIDER, PALADIN, RANGER, ROGUE,
SHAPECHANGER, SORCERER, UNDEAD, VERMIN,
WIZARD

COMBAT_MODE_

FLURRY_OF_BLOWS, IMPROVED_POWER_ATTACK,
INVALID, PARRY, POWER_ATTACK, RAPID_SHOT

CREATURE_SIZE_

HUGE, INVALID, LARGE, MEDIUM, SMALL, TINY

CREATURE_TYPE_

CLASS, DOES_NOT_HAVE_SPELL_EFFECT,
HAS_SPELL_EFFECT, IS_ALIVE, PERCEPTION,
PLAYER_CHAR, RACIAL_TYPE, REPUTATION

DAMAGE_BONUS_

1, 1d10, 1d4, 1d6, 1d8, 2, 2d6, 3, 4, 5

DAMAGE_POWER_

ENERGY, NORMAL, PLUS_FIVE, PLUS_FOUR, PLUS_ONE,
PLUS_THREE, PLUS_TWO

DAMAGE_TYPE_

ACID, BLUDGEONING, COLD, DIVINE, ELECTRICAL,
FIRE, MAGICAL, NEGATIVE, PIERCING, POSITIVE,
SLASHING, SONIC

DIRECTION_

EAST, NORTH, SOUTH, WEST

DISEASE_

BLINDING_SICKNESS, BURROW_MAGGOTS,
CACKLE_FEVER, DEMON_FEVER, DEVIL_CHILLS,
DREAD_BLISTERS, FILTH_FEVER, GHOUL_ROT,
MINDFIRE, MUMMY_ROT, RED_ACHE,
RED_SLAAD_EGGS, SHAKES, SLIMY_DOOM,
SOLDIER_SHAKES, VERMIN_MADNESS, ZOMBIE_CREEP

DOOR_ACTION_

BASH, IGNORE, KNOCK, OPEN, UNLOCK

DURATION_TYPE_

INSTANT, PERMANENT, TEMPORARY

EFFECT_TYPE_

ABILITY_DECREASE, ABILITY_INCREASE,
AC_DECREASE, AC_INCREASE,
ARCANE_SPELL_FAILURE, AREA_OF_EFFECT,
ATTACK_DECREASE, ATTACK_INCREASE, BEAM,
BLINDNESS, CHARMED, CONCEALMENT, CONFUSED,
CURSE, DAMAGE_DECREASE,
DAMAGE_IMMUNITY_DECREASE,
DAMAGE_IMMUNITY_INCREASE, DAMAGE_INCREASE,
DAMAGE_REDUCTION, DAMAGE_RESISTANCE,
DARKNESS, DAZED, DEAF, DISEASE, DISPELMAGICALL,
DISPELMAGICBEST, DOMINATED, ELEMENTALSHIELD,
ENEMY_ATTACK_BONUS, ENTANGLE, FRIGHTENED,
HASTE, IMMUNITY, IMPROVEDINVISIBILITY,
INVALIDEFFECT, INVISIBILITY, INVULNERABLE,
MISS_CHANCE, MOVEMENT_SPEED_DECREASE,
MOVEMENT_SPEED_INCREASE, NEGATIVELEVEL,
PARALYZE, POISON, POLYMORPH, REGENERATE,
RESURRECTION, SANCTUARY,
SAVING_THROW_DECREASE,
SAVING_THROW_INCREASE, SEEINVISIBLE, SILENCE,
SKILL_DECREASE, SKILL_INCREASE, SLEEP, SLOW,
SPELL_IMMUNITY, SPELL_RESISTANCE_DECREASE,
SPELL_RESISTANCE_INCREASE,
SPELLLEVELABSORPTION, STUNNED,
TEMPORARY_HITPOINTS, TIMESTOP, TRUESEEING,
TURNED, ULTRAVISION

ENCOUNTER_DIFFICULTY_

EASY, HARD, IMPOSSIBLE, NORMAL, VERY_EASY

FAMILIAR_CREATURE_TYPE_

BAT, CRAGCAT, FIREMEPHIT, HELLHOUND, ICEMEPHIT,
IMP, NONE, PIXIE, RAVEN

FEAT_

AIR_DOMAIN_POWER, ALERTNESS, AMBIDEXTERITY,
ANIMAL_COMPANION, ANIMAL_DOMAIN_POWER,
ARMOR_PROFICIENCY_HEAVY,
ARMOR_PROFICIENCY_LIGHT,
ARMOR_PROFICIENCY_MEDIUM, AURA_OF_COURAGE,
BARBARIAN_ENDURANCE, BARBARIAN_RAGE,
BARD_SONGS, BARDIC_KNOWLEDGE,
BATTLE_TRAINING_VERSUS_GIANTS,
BATTLE_TRAINING_VERSUS_GOBLINS,
BATTLE_TRAINING_VERSUS_ORCS,
BATTLE_TRAINING_VERSUS_REPTILIANS,
CALLED_SHOT, CLEAVE, COMBAT_CASTING,
CRIPPLING_STRIKE, DAMAGE_REDUCTION,
DARKVISION, DEATH_DOMAIN_POWER,
DEFENSIVE_ROLL, DEFLECT_ARROWS,
DESTRUCTION_DOMAIN_POWER, DIAMOND_BODY,
DIAMOND_SOUL, DISARM, DIVINE_GRACE,
DIVINE_HEALTH, DODGE, EARTH_DOMAIN_POWER,
ELEMENTAL_SHAPE, EMPOWER_SPELL, EMPTY_BODY,
EVASION, EVIL_DOMAIN_POWER, EXTEND_SPELL,
EXTRA_TURNING

FAVORED_ENEMY_

ABERRATION, ANIMAL, BEAST, CONSTRUCT,
DRAGON, DWARF, ELEMENTAL, ELF, FEY, GIANT,
GNOME, GOBLINOID, HALFPEL, HALFLING,
HALFROC, HUMAN, MAGICAL_BEAST, MONSTROUS,
ORC, OUTSIDER, REPTILIAN, SHAPECHANGER,
UNDEAD, VERMIN

FEARLESS, FIRE_DOMAIN_POWER, FLURRY_OF_BLOWS,
GOOD_AIM, GOOD_DOMAIN_POWER,
GREAT_FORTITUDE

HARDINESS_VERSUS_

ENCHANTMENTS, ILLUSIONS, POISONS, SPELLS
HEALING_DOMAIN_POWER, IMMUNITY_TO_SLEEP

IMPROVED_CRITICAL_

BASTARD_SWORD, BATTLE_AXE, CLUB, CREATURE,
DAGGER, DART, DIRE_MACE, DOUBLE_AXE,
GREAT_AXE, GREAT_SWORD, HALBERD,
HAND_AXE, HEAVY_CROSSBOW, HEAVY_FLAIL,
KAMA, KATANA, KUKRI, LIGHT_CROSSBOW,
LIGHT_FLAIL, LIGHT_HAMMER, LIGHT_MACE,
LONG_SWORD, LONGBOW, MORNING_STAR,
RAPIER, SCIMITAR, SCYTHE, SHORT_SWORD,
SHORTBOW, SHURIKEN, SICKLE, SLING, SPEAR,
STAFF, THROWING_AXE, TWO_BLADED_SWORD,
UNARMED_STRIKE, WAR_HAMMER

IMPROVED_

DISARM, EVASION, KNOCKDOWN, PARRY,
POWER_ATTACK, TWO_WEAPON_FIGHTING,
UNARMED_STRIKE

IRON_WILL, KEEN_SENSE, KI_STRIKE, KNOCKDOWN,
KNOWLEDGE_DOMAIN_POWER, LAY_ON_HANDS,
LIGHTNING_REFLEXES, LOWLIGHTVISION,
LUCK_DOMAIN_POWER, LUCKY,
MAGIC_DOMAIN_POWER, MAXIMIZE_SPELL, MOBILITY,
MONK_AC_BONUS, MONK_ENDURANCE,
NATURE_SENSE, OPPORTUNIST,
PARTIAL_SKILL_AFFINITY_LISTEN,
PARTIAL_SKILL_AFFINITY_SEARCH,
PARTIAL_SKILL_AFFINITY_SPOT, PERFECT_SELF,
PLANT_DOMAIN_POWER, POINT_BLANK_SHOT,
POWER_ATTACK, PROTECTION_DOMAIN_POWER,
PURITY_OF_BODY, QUICK_TO_MASTER,
QUICKEN_SPELL, QUIVERING_PALM, RAPID_SHOT,
REMOVE_DISEASE, RESIST_NATURES_LURE, SAP,
SHIELD_PROFICIENCY, SILENCE_SPELL

SKILL_AFFINITY_

CONCENTRATION, LISTEN, LORE, MOVE_SILENTLY,
SEARCH, SPOT

SKILL_FOCUS_

ANIMAL_EMPATHY, CONCENTRATION,
DISABLE_TRAP, DISCIPLINE, HEAL, HIDE, LISTEN,
LORE, MOVE_SILENTLY, OPEN_LOCK, PARRY,
PERFORM, PERSUADE, PICK_POCKET, SEARCH,
SET_TRAP, SPELLCRAFT, SPOT, TAUNT,
USE_MAGIC_DEVICE

SKILL_MASTERY, SLIPPERY_MIND, SMITE_EVIL,
SNEAK_ATTACK

SPELL_FOCUS_

ABJURATION, CONJURATION, DIVINATION,
ENCHANTMENT, EVOCATION, ILLUSION,
NECROMANCY, TRANSMUTATION

SPELL_PENETRATION, STILL_MIND, STILL_SPELL,
STONECUNNING, STRENGTH_DOMAIN_POWER,
STUNNING_FIST, SUMMON_FAMILIAR,
SUN_DOMAIN_POWER, TOUGHNESS, TRACKLESS_STEP,
TRAVEL_DOMAIN_POWER, TRICKERY_DOMAIN_POWER,
TURN_UNDEAD, TWO_WEAPON_FIGHTING,
UNCANNY_DODGE_1, UNCANNY_DODGE_2,
UNCANNY_DODGE_3, UNCANNY_DODGE_4,
UNCANNY_DODGE_5, UNCANNY_DODGE_6,
UNCANNY_REFLEX, VENOM_IMMUNITY,
WAR_DOMAIN_POWER, WATER_DOMAIN_POWER,
WEAPON_FINESSE

WEAPON_FOCUS_

BASTARD_SWORD, BATTLE_AXE, CLUB, CREATURE,
DAGGER, DART, DIRE_MACE, DOUBLE_AXE,
GREAT_AXE, GREAT_SWORD, HALBERD,
HAND_AXE, HEAVY_CROSSBOW, HEAVY_FLAIL,
KAMA, KATANA, KUKRI, LIGHT_CROSSBOW,
LIGHT_FLAIL, LIGHT_HAMMER, LIGHT_MACE,
LONG_SWORD, LONGBOW, MORNING_STAR,
RAPIER, SCIMITAR, SCYTHE, SHORT_SWORD,
SHORTBOW, SHURIKEN, SICKLE, SLING, SPEAR,
STAFF, THROWING_AXE, TWO_BLADED_SWORD,
UNARMED_STRIKE, WAR_HAMMER

WEAPON_PROFICIENCY_

CREATURE, DRUID, ELF, EXOTIC, MARTIAL, MONK,
ROGUE, SIMPLE, WIZARD

WEAPON_SPECIALIZATION_

BASTARD_SWORD, BATTLE_AXE, CLUB, CREATURE,
DAGGER, DART, DIRE_MACE, DOUBLE_AXE,
GREAT_AXE, GREAT_SWORD, HALBERD,
HAND_AXE, HEAVY_CROSSBOW, HEAVY_FLAIL,
KAMA, KATANA, KUKRI, LIGHT_CROSSBOW,
LIGHT_FLAIL, LIGHT_HAMMER, LIGHT_MACE,
LONG_SWORD, LONGBOW, MORNING_STAR,
RAPIER, SCIMITAR, SCYTHE, SHORT_SWORD,
SHORTBOW, SHURIKEN, SICKLE, SLING, SPEAR,

STAFF, THROWING_AXE, TWO_BLADED_SWORD, UNARMED_STRIKE, WAR_HAMMER
WHOLENESS_OF_BODY, WILD_SHAPE, WOODLAND_STRIDE

GAME_DIFFICULTY_
CORE_RULES, DIFFICULT, EASY, NORMAL, VERY_EASY

GENDER_
BOTH, FEMALE, MALE, NONE, OTHER

GUI_
PANEL_PLAYER_DEATH

IMMUNITY_TYPE_
ABILITY_DECREASE, AC_DECREASE, ATTACK_DECREASE, BLINDNESS, CHARM, CONFUSED, CRITICAL_HIT, CURSED, DAMAGE_DECREASE, DAMAGE_IMMUNITY_DECREASE, DAZED, DEAFNESS, DEATH, DISEASE, DOMINATE, ENTANGLE, FEAR, KNOCKDOWN, MIND_SPELLS, MOVEMENT_SPEED_DECREASE, NEGATIVE_LEVEL, NONE, PARALYSIS, POISON, SAVING_THROW_DECREASE, SILENCE, SKILL_DECREASE, SLEEP, SLOW, SNEAK_ATTACK, SPELL_RESISTANCE_DECREASE, STUN, TRAP

INVENTORY_
DISTURB_
TYPE_ADDED, TYPE_REMOVED, TYPE_STOLEN

SLOT_
ARMS, ARROWS, BELT, BOLTS, BOOTS, BULLETS, CARMOUR, CHEST, CLOAK, CWEAPON_B, CWEAPON_L, CWEAPON_R, HEAD, LEFTHAND, LEFTRING, NECK, RIGHTHAND, RIGHTRING

INVISIBILITY_TYPE_
DARKNESS, IMPROVED, NORMAL

ITEM_PROPERTY_
ABILITY_BONUS, AC_BONUS, AC_BONUS_VS_ALIGNMENT_GROUP, AC_BONUS_VS_DAMAGE_TYPE, AC_BONUS_VS_RACIAL_GROUP, AC_BONUS_VS_SPECIFIC_ALIGNMENT, ATTACK_BONUS, ATTACK_BONUS_VS_ALIGNMENT_GROUP, ATTACK_BONUS_VS_RACIAL_GROUP, ATTACK_BONUS_VS_SPECIFIC_ALIGNMENT, BASE_ITEM_WEIGHT_REDUCTION, BONUS_FEAT, BONUS_SPELL_SLOT_OF_LEVEL_N, BOOMERANG, CAST_SPELL, DAMAGE_BONUS, DAMAGE_BONUS_VS_ALIGNMENT_GROUP, DAMAGE_BONUS_VS_RACIAL_GROUP, DAMAGE_BONUS_VS_SPECIFIC_ALIGNMENT, DAMAGE_REDUCTION, DAMAGE_RESISTANCE, DAMAGE_VULNERABILITY, DANCING, DARKVISION, DECREASED_ABILITY_SCORE, DECREASED_AC, DECREASED_ATTACK_MODIFIER, DECREASED_DAMAGE, DECREASED_ENHANCEMENT_MODIFIER, DECREASED_SAVING_THROWS, DECREASED_SAVING_THROWS_SPECIFIC, DECREASED_SKILL_MODIFIER, DOUBLE_STACK, ENHANCED_CONTAINER_BONUS_SLOTS, ENHANCED_CONTAINER_REDUCED_WEIGHT, ENHANCEMENT_BONUS, ENHANCEMENT_BONUS_VS_ALIGNMENT_GROUP, ENHANCEMENT_BONUS_VS_RACIAL_GROUP, ENHANCEMENT_BONUS_VS_SPECIFIC_ALIGNMENT, EXTRA_MELEE_DAMAGE_TYPE, EXTRA_RANGED_DAMAGE_TYPE, FREEDOM_OF_MOVEMENT, HASTE, HOLY_AVENGER, IMMUNITY_DAMAGE_TYPE, IMMUNITY_MISCELLANEOUS, IMMUNITY_SPECIFIC_SPELL, IMMUNITY_SPELL_SCHOOL, IMMUNITY_SPELLS_BY_LEVEL, IMPROVED_EVASION, KEEN, LIGHT, MASSIVE_CRITICALS, MIGHTY, MIND_BLANK, MONSTER_DAMAGE, NO_DAMAGE, ON_HIT_PROPERTIES, ON_MONSTER_HIT, POISON, REGENERATION, REGENERATION_VAMPIRIC, SAVING_THROW_BONUS, SAVING_THROW_BONUS_SPECIFIC, SKILL_BONUS, SPELL_RESISTANCE, THIEVES_TOOLS, TRAP, TRUE_SEEING, TURN_RESISTANCE, UNLIMITED_AMMUNITION, USE_LIMITATION_ALIGNMENT_GROUP, USE_LIMITATION_CLASS, USE_LIMITATION_RACIAL_TYPE, USE_LIMITATION_SPECIFIC_ALIGNMENT, USE_LIMITATION_TILESET, VORPAL, WOUNDING

METAMAGIC_
ANY, EMPOWER, EXTEND, MAXIMIZE, NONE, QUICKEN, SILENT, STILL

NUM_
INVENTORY_SLOTS

OBJECT_TYPE_
ALL, AREA_OF_EFFECT, CREATURE, DOOR, INVALID, ITEM, PLACEABLE, STORE, TRIGGER, WAYPOINT

PANEL_BUTTON_
CHARACTER, INVENTORY, JOURNAL, MAP, OPTIONS, PLAYER_VERSUS_PLAYER, REST, SPELLS

PERCEPTION_
HEARD, HEARD_AND_NOT_SEEN, NOT_HEARD, NOT_SEEN, NOT_SEEN_AND_NOT_HEARD, SEEN, SEEN_AND_HEARD, SEEN_AND_NOT_HEARD

PERSISTENT_
ZONE_ACTIVE, ZONE_FOLLOW

PI

PLACEABLE_ACTION_
BASH, KNOCK, UNLOCK, USE

PLAYER_CHAR_
IS_PC, NOT_PC

POISON_
ARANEA_VENOM, ARSENIC, BEBILITH_VENOM, BLACK_ADDER_VENOM, BLACK_LOTUS_EXTRACT, BLADE_BANE, BLOODROOT, BLUE_WHINNIS, BURNT_OTHUR_FUMES, CARRION_CRAWLER_BRAIN_JUICE, CHAOS_MIST, COLOSSAL_SPIDER_VENOM, DARK_REAVER_POWDER, DEATHBLADE, DRAGON_BILE, ETTERCAP_VENOM, GARGANTUAN_SPIDER_VENOM, GIANT_WASP_POISON, GREENBLOOD_OIL, HUGE_SPIDER_VENOM, ID_MOSS, IRON_GOLEM, LARGE_SCORPION_VENOM, LARGE_SPIDER_VENOM, LICH_DUST, MALYSS_ROOT_PASTE, MEDIUM_SPIDER_VENOM, NIGHTSHADE, NITHARIT, OIL_OF_TAGGIT, PHASE_SPIDER_VENOM, PIT_FIEND_ICHOR, PURPLE_WORM_POISON, QASIT_VENOM, SASSONE_LEAF_RESIDUE, SHADOW_ESSENCE, SMALL_CENTIPED_POISON, SMALL_SPIDER_VENOM, STRIPED_TOADSTOOL, TERINAV_ROOT, TINY_SPIDER_VENOM, UNGOL_DUST, WRAITH_SPIDER_VENOM, WYVERN_POISON

POLYMORPH_TYPE_
BADGER, BALOR, BOAR, BROWN_BEAR, COW, DEATH_SLAAD, DOOM_KNIGHT, ELDER_AIR_ELEMENTAL, ELDER_EARTH_ELEMENTAL, ELDER_FIRE_ELEMENTAL, ELDER_WATER_ELEMENTAL, FIRE_GIANT, GIANT_SPIDER, HUGE_AIR_ELEMENTAL, HUGE_EARTH_ELEMENTAL, HUGE_FIRE_ELEMENTAL, HUGE_WATER_ELEMENTAL, IMP, IRON_GOLEM, PANTHER, PENGUIN, PIXIE, QUASIT, RED_DRAGON, SUCCUBUS, TROLL, UMBER_HULK, WERECAT, WERERAT, WEREWOLF, WOLF, YUANTI, ZOMBIE

PROJECTILE_PATH_TYPE_
ACCELERATING, BALLISTIC, DEFAULT, HIGH_BALLISTIC, HOMING

RACIAL_TYPE_
ABERRATION, ALL, ANIMAL, BEAST, CONSTRUCT, DRAGON, DWARF, ELEMENTAL, ELF, FEY, GIANT, GNOME, HALFELF, HALFLING, HALFORG, HUMAN, HUMANOID_GOBLINOID, HUMANOID_MONSTROUS, HUMANOID_ORC, HUMANOID_REPTILIAN, INVALID, MAGICAL_BEAST, OUTSIDER, SHAPECHANGER, UNDEAD, VERMIN

RADIUS_SIZE_
COLOSSAL, GARGANTUAN, HUGE, LARGE, MEDIUM, SMALL

REPUTATION_TYPE_
ENEMY, FRIEND, NEUTRAL

REST_EVENTTYPE_REST_
CANCELLED, FINISHED, INVALID, STARTED

SAVING_THROW_
ALL, FORT, REFLEX, WILL

TYPE_
ACID, ALL, CHAOS, COLD, DEATH, DISEASE, DIVINE, ELECTRICITY, EVIL, FEAR, FIRE, GOOD, LAW, MIND_SPELLS, NEGATIVE, NONE, POISON, POSITIVE, SONIC, SPELL, TRAP

SHAPE_
CONE, CUBE, SPELLCONE, SPELLCYLINDER, SPHERE

SKILL_
ALL_SKILLS, ANIMAL_EMPATHY, CONCENTRATION, DISABLE_TRAP, DISCIPLINE, HEAL, HIDE, LISTEN, LORE, MOVE_SILENTLY, OPEN_LOCK, PARRY, PERFORM, PERSUADE, PICK_POCKET, SEARCH, SET_TRAP, SPELLCRAFT, SPOT, TAUNT, USE_MAGIC_DEVICE

SPECIAL_ATTACK_
CALLED_SHOT_ARM, CALLED_SHOT_LEG, DISARM, FLURRY_OF_BLOWS, IMPROVED_DISARM, IMPROVED_KNOCKDOWN, INVALID, KNOCKDOWN, RAPID_SHOT, SAP, STUNNING_FIST

SPELL_
ACID_FOG, AID, ANIMATE_DEAD, AURA_OF_VITALITY, AWAKEN, BARKSKIN, BESTOW_CURSE, BLADE_BARRIER, BLESS, BLESS_WEAPON, BLINDNESS_AND_DEAFNESS, BULLS_STRENGTH, BURNING_HANDS, CALL_LIGHTNING, CATS_GRACE, CHAIN_LIGHTNING, CHARM_MONSTER, CHARM_PERSON, CHARM_PERSON_OR_ANIMAL, CIRCLE_OF_DEATH, CIRCLE_OF_DOOM, CLAIRAUDIENCE_AND_CLAIRVOYANCE, CLARITY, CLOAK_OF_CHAOS, CLOUDKILL, COLOR_SPRAY, CONE_OF_COLD, CONFUSION, CONTAGION, CONTROL_UNDEAD, CREATE_GREATER_UNDEAD, CREATE_UNDEAD, CREEPING_DOOM, CURE_CRITICAL_WOUNDS, CURE_LIGHT_WOUNDS, CURE_MINOR_WOUNDS, CURE_MODERATE_WOUNDS, CURE_SERIOUS_WOUNDS, DARKNESS, DARKVISION, DAZE, DEATH_WARD, DELAYED_BLAST_FIREBALL, DESTRUCTION, DISMISSAL, DISPEL_MAGIC, DIVINE_POWER, DOMINATE_ANIMAL, DOMINATE_MONSTER, DOMINATE_PERSON, DOOM, EAGLE_SPLEDOR, ELEMENTAL_SHIELD, ELEMENTAL_SWARM, ENDURANCE, ENDURE_ELEMENTS, ENERGY_BUFFER, ENERGY_DRAIN, ENERVATION, ENTANGLE, ETHEREAL_VISAGE, EVARDS_BLACK_TENTACLES, FEAR, FEEBLEMIND, FIND_TRAPS, FINGER_OF_DEATH, FIRE_STORM, FIREBALL, FLAME_ARROW, FLAME_LASH, FLAME_STRIKE, FOXS_CUNNING, FREEDOM_OF_MOVEMENT, GATE, GHOSTLY_VISAGE, GHOUL_TOUCH, GLOBE_OF_INVULNERABILITY, GREASE, GREATER_BULLS_STRENGTH, GREATER_CATS_GRACE, GREATER_DISPELLING, GREATER_EAGLE_SPLENDOR, GREATER_ENDURANCE, GREATER_FOXS_CUNNING, GREATER_MAGIC_WEAPON, GREATER_OWLS_WISDOM, GREATER_PLANAR_BINDING, GREATER_RESTORATION, GREATER_SHADOW_CONJURATION_ACID_ARROW, GREATER_SHADOW_CONJURATION_MINOR_GLOBE, GREATER_SHADOW_CONJURATION_MIRROR_IMAGE, GREATER_SHADOW_CONJURATION_SUMMON_SHADOW, GREATER_SHADOW_CONJURATION_WEB, GREATER_SPELL_BREACH, GREATER_SPELL_MANTLE, GREATER_STONESKIN, HAMMER_OF_THE_GODS, HARM, HASTE, HEAL, HEALING_CIRCLE, HOLD_ANIMAL, HOLD_MONSTER, HOLD_PERSON, HOLY_AURA, HOLY_SWORD, HORRID_WILTING, ICE_STORM, IDENTIFY, IMPLOSION, IMPROVED_INVISIBILITY, INCENDIARY_CLOUD, INVISIBILITY, INVISIBILITY_PURGE, INVISIBILITY_SPHERE, KNOCK, LEGEND_LORE, LESSER_DISPEL, LESSER_MIND_BLANK, LESSER_PLANAR_BINDING, LESSER_RESTORATION, LESSER_SPELL_BREACH, LESSER_SPELL_MANTLE, LIGHT, LIGHTNING_BOLT, MAGE_ARMOR, MAGIC_CIRCLE_AGAINST_CHAOS, MAGIC_CIRCLE_AGAINST_EVIL, MAGIC_CIRCLE_AGAINST_GOOD, MAGIC_CIRCLE_AGAINST_LAW, MAGIC_MISSILE, MAGIC_VESTMENT, MAGIC_WEAPON, MASS_BLINDNESS_AND_DEAFNESS, MASS_CHARM, MASS_HASTE, MASS_HEAL, MELFS_ACID_ARROW, METEOR_SWARM, MIND_BLANK, MIND_FOG, MINOR_GLOBE_OF_INVULNERABILITY, MORDENKAINENS_DISJUNCTION, MORDENKAINENS_SWORD, NATURES_BALANCE, NEGATIVE_ENERGY_BURST, NEGATIVE_ENERGY_PROTECTION, NEGATIVE_ENERGY_RAY, NEUTRALIZE_POISON, OWLS_WISDOM, PHANTASMAL_KILLER, PLANAR_BINDING, POISON, POLYMORPH_SELF, POWER_WORD_KILL, POWER_WORD_STUN, PRAYER, PREMONITION, PRISMATIC_SPRAY, PROTECTION_FROM_CHAOS, PROTECTION_FROM_ELEMENTS, PROTECTION_FROM_EVIL, PROTECTION_FROM_GOOD, PROTECTION_FROM_LAW, PROTECTION_FROM_SPELLS, RAISE_DEAD, RAY_OF_ENFEEBLEMENT, RAY_OF_FROST, REGENERATE, REMOVE_BLINDNESS_AND_DEAFNESS, REMOVE_CURSE, REMOVE_DISEASE, REMOVE_FEAR, REMOVE_PARALYSIS, RESIST_ELEMENTS, RESISTANCE, RESTORATION, RESURRECTION, SANCTUARY, SCARE, SCHOOL_ABJURATION, SCHOOL_CONJURATION, SCHOOL_DIVINATION, SCHOOL_ENCHANTMENT, SCHOOL_EVOCATION, SCHOOL_GENERAL, SCHOOL_ILLUSION, SCHOOL_NECROMANCY, SCHOOL_TRANSMUTATION, SEARING_LIGHT, SEE_INVISIBILITY, SHADES_CONE_OF_COLD, SHADES_FIREBALL, SHADES_STONESKIN, SHADES_SUMMON_SHADOW, SHADES_WALL_OF_FIRE, SHADOW_CONJURATION_DARKNESS, SHADOW_CONJURATION_INVISIBILITY, SHADOW_CONJURATION_MAGE_ARMOR, SHADOW_CONJURATION_MAGIC_MISSILE, SHADOW_CONJURATION_SUMMON_SHADOW, SHADOW_SHIELD, SHAPECHANGER, SHIELD_OF_LAW, SILENCE, SLAY_LIVING, SLEEP, SLOW, SOUND_BURST, SPELL_MANTLE, SPELL_RESISTANCE,

SPHERE_OF_CHAOS, STINKING_CLOUD, STONESKIN,
 STORM_OF_VENGEANCE, SUMMON_CREATURE_I,
 SUMMON_CREATURE_II, SUMMON_CREATURE_III,
 SUMMON_CREATURE_IV, SUMMON_CREATURE_IX,
 SUMMON_CREATURE_V, SUMMON_CREATURE_VI,
 SUMMON_CREATURE_VII, SUMMON_CREATURE_VIII,
 SUNBEAM, TENSERS_TRANSFORMATION, TIME_STOP,
 TRUE_SEEING, UNHOLY_AURA, VAMPIRIC_TOUCH,
 VIRTUE, WAIL_OF_THE_BANSHEE, WALL_OF_FIRE,
 WAR_CRY, WEB, WEIRD, WORD_OF_FAITH

SPELLABILITY_

ACTIVATE_ITEM, BARBARIAN_RAGE, BATTLE_MASTERY,
 DETECT_EVIL, DIVINE_PROTECTION,
 DIVINE_STRENGTH, DIVINE_TRICKERY,
 ELEMENTAL_SHAPE, EMPTY_BODY, FEROCITY_1,
 FEROCITY_2, FEROCITY_3, GOLEM_BREATH_GAS,
 HELL_HOUND_FIREBREATH, KRENSHAR_SCARE,
 LAY_ON_HANDS, LESSER_BODY_ADJUSTMENT,
 MEPHIT_SALT_BREATH, MEPHIT_STEAM_BREATH,
 MUMMY_BOLSTER_UNDEAD,
 NEGATIVE_PLANE_AVATAR, QUIVERING_PALM,
 RAGE_3, RAGE_4, RAGE_5, REMOVE_DISEASE,
 ROGUES_CUNNING, SMITE_EVIL, SMOKE_CLAW,
 TRUMPET_BLAST, TURN_UNDEAD, TYRANT_FOG_MIST,
 WHOLENES_OF_BODY, WILD_SHAPE

AURA_

BLINDING, COLD, ELECTRICITY, FEAR, FIRE,
 MENACE, OF_COURAGE, PROTECTION, STUN,
 UNEARTHLY_VISAGE, UNNATURAL

BOLT_ABILITY_

DRAIN_CHARISMA, DRAIN_CONSTITUTION,
 DRAIN_DEXTERITY, DRAIN_INTELLIGENCE,
 DRAIN_STRENGTH, DRAIN_WISDOM

BOLT_

ACID, CHARM, COLD, CONFUSE, DAZE, DEATH,
 DISEASE, DOMINATE, FIRE, KNOCKDOWN,
 LEVEL_DRAIN, LIGHTNING, PARALYZE, POISON,
 SHARDS, SLOW, STUN, WEB

CONE_

ACID, COLD, DISEASE, FIRE, LIGHTNING, POISON,
 SONIC

DRAGON_

FEAR, WING_BUFFET

DRAGON_BREATH_

ACID, COLD, FEAR, FIRE, GAS, LIGHTNING,
 PARALYZE, SLEEP, SLOW, WEAKEN

GAZE_

CHARM, CONFUSION, DAZE, DEATH,
 DESTROY_CHAOS, DESTROY_EVIL,
 DESTROY_GOOD, DESTROY_LAW, DOMINATE,
 DOOM, FEAR, PARALYSIS, STUNNED

HOWL_

CONFUSE, DAZE, DEATH, DOOM, FEAR, PARALYSIS,
 SONIC, STUN

INTENSITY_

1, 2, 3

PULSE_ABILITY_

DRAIN_CHARISMA, DRAIN_CONSTITUTION,
 DRAIN_DEXTERITY, DRAIN_INTELLIGENCE,
 DRAIN_STRENGTH, DRAIN_WISDOM

PULSE_

COLD, DEATH, DISEASE, DROWN, FIRE, HOLY,
 LEVEL_DRAIN, LIGHTNING, NEGATIVE, POISON,
 SPORES, WHIRLWIND

SUMMON_

ANIMAL_COMPANION, CELESTIAL, FAMILIAR,
 MEPHIT, SLAAD, TANARRI

STANDARD_FACTION_

COMMONER, DEFENDER, HOSTILE, MERCHANT

SUBTYPE_

EXTRAORDINARY, MAGICAL, SUPERNATURAL

TALENT_CATEGORY_

DRAGONS_BREATH, PERSISTENT_AREA_OF_EFFECT

BENEFICIAL_

CONDITIONAL_AREAEFFECT,
 CONDITIONAL_POTION, CONDITIONAL_SINGLE,
 ENHANCEMENT_AREAEFFECT,
 ENHANCEMENT_POTION, ENHANCEMENT_SELF,
 ENHANCEMENT_SINGLE, HEALING_AREAEFFECT,
 HEALING_POTION, HEALING_TOUCH,
 OBTAIN_ALLIES, PROTECTION_AREAEFFECT,
 PROTECTION_POTION, PROTECTION_SELF,
 PROTECTION_SINGLE

HARMFUL_

AREAEFFECT_DISCRIMINANT,
 AREAEFFECT_INDISCRIMINANT, RANGED, TOUCH

TALENT_TYPE_

FEAT, SKILL, SPELL

TALKVOLUME_

SHOUT, SILENT_SHOUT, SILENT_TALK, TALK, WHISPER

TILE_MAIN_LIGHT_COLOR_

AQUA, BLACK, BLUE, BRIGHT_WHITE, DARK_AQUA,
 DARK_BLUE, DARK_GREEN, DARK_ORANGE,
 DARK_PURPLE, DARK_RED, DARK_YELLOW,
 DIM_WHITE, GREEN, ORANGE, PALE_AQUA,
 PALE_BLUE, PALE_DARK_AQUA, PALE_DARK_BLUE,
 PALE_DARK_GREEN, PALE_DARK_ORANGE,
 PALE_DARK_PURPLE, PALE_DARK_RED,
 PALE_DARK_YELLOW, PALE_GREEN, PALE_ORANGE,
 PALE_PURPLE, PALE_RED, PALE_YELLOW, PURPLE, RED,
 WHITE, YELLOW

VOICE_CHAT_

ATTACK, BADIDEA, BATTLECRY1, BATTLECRY2,
 BATTLECRY3, BORED, CANDO, CANTDO, CHEER, CUSS,
 DEATH, ENCUMBERED, ENEMIES, FLEE, FOLLOWME,
 GATTACK1, GATTACK2, GATTACK3, GOODBYE,
 GOODIDEA, GROUP, GUARDME, HEALME, HELLO, HELP,
 HIDE, HOLD, LAUGH, LOOKHERE, MOVEOVER,
 NEARDEATH, NO, PAIN1, PAIN2, PAIN3, PICKLOCK,
 POISONED, REST, SEARCH, SELECTED, SPELLFAILED,
 STOP, TALKTOME, TASKCOMPLETE, TAUNT, THANKS,
 THREATEN, WEAPONSUCKS, YES

WEATHER_

CLEAR, RAIN, SNOW, USE_AREA_SETTINGS

ANIMATION_FIREFORGET_

BOW, DRINK, GREETING, HEAD_TURN_LEFT, HEAD_TURN_RIGHT, PAUSE_BORED, PAUSE_SCRATCH_HEAD, READ, SALUTE, STEAL, TAUNT, VICTORY1, VICTORY2, VICTORY3

ANIMATION_LOOPING_

GET_LOW, GET_MID, LISTEN, LOOK_FAR, MEDITATE, PAUSE, PAUSE_DRUNK, PAUSE_TIRED, PAUSE2, SIT_CHAIR, SIT_CROSS, TALK_FORCEFUL, TALK_LAUGHING, TALK_NORMAL, TALK_PLEADING, WORSHIP

ANIMATION_PLACEABLE_

ACTIVATE, CLOSE, DEACTIVATE, OPEN

VFX_

NONE

VFX_BEAM_

COLD, EVIL, FIRE, FIRE_LASH, HOLY, LIGHTNING, MIND, ODD

VFX_COM_

SPARKS_PARRY, UNLOAD_MODEL

BLOOD_

CRT_GREEN, CRT_RED, CRT_WIMP, CRT_YELLOW, LRG_GREEN, LRG_RED, LRG_WIMP, LRG_YELLOW, REG_GREEN, REG_RED, REG_WIMP, REG_YELLOW, SPARK_LARGE, SPARK_MEDIUM, SPARK_SMALL

CHUNK_

BONE_MEDIUM, GREEN_MEDIUM, GREEN_SMALL, RED_LARGE, RED_MEDIUM, RED_SMALL, YELLOW_MEDIUM, YELLOW_SMALL

HIT_

ACID, DIVINE, ELECTRICAL, FIRE, FROST, NEGATIVE, SONIC

SPECIAL_

BLUE_RED, PINK_ORANGE, RED_ORANGE, RED_WHITE, WHITE_BLUE, WHITE_ORANGE

VFX_DUR_

ANTI_LIGHT_10, BARD_SONG, BLACKOUT, BLIND, BLINDVISION, BLUR, CESSATE_NEGATIVE, CESSATE_NEUTRAL, CESSATE_POSITIVE, DARKNESS, DARKVISION, ELEMENTAL_SHIELD, ENTANGLE, ETHEREAL_VISAGE, FREEDOM_OF_MOVEMENT, GHOSTLY_PULSE, GHOSTLY_VISAGE, GLOBE_INVULNERABILITY, GLOBE_MINOR, INVISIBILITY, LIGHT, LOWLIGHTVISION, MAGIC_RESISTANCE, MAGICAL_SIGHT, MIRV_ACID, PARALYZE_HOLD, PARALYZED, SANCTUARY, SPELLTURNING, ULTRAVISION, WEB, WEB_MASS

AURA_

COLD, DISEASE, DRAGON_FEAR, FIRE, ODD, POISON, SILENCE

LIGHT_

BLUE_10, BLUE_15, BLUE_20, BLUE_5, GREY_10, GREY_15, GREY_20, GREY_5, ORANGE_10, ORANGE_15, ORANGE_20, ORANGE_5, PURPLE_10, PURPLE_15, PURPLE_20, PURPLE_5, RED_10, RED_15, RED_20, RED_5, WHITE_10, WHITE_15, WHITE_20, WHITE_5, YELLOW_10, YELLOW_15, YELLOW_20, YELLOW_5

MIND_AFFECTING_

DISABLED, DOMINATED, FEAR, NEGATIVE, POSITIVE

PROT_

BARKSKIN, GREATER_STONESKIN, PREMONITION, SHADOW_ARMOR, STONESKIN

PROTECTION_

ELEMENTS, EVIL_MAJOR, EVIL_MINOR, GOOD_MAJOR, GOOD_MINOR

VFX_FNF_

BLINDDEAF, DISPEL, DISPEL_DISJUNCTION, DISPEL_GREATER, FIREBALL, FIRESTORM, HORRID_WILTING, ICESTORM, IMPLOSION, MASS_HEAL, MASS_MIND_AFFECTING, METEOR_SWARM, NATURES_BALANCE, PWKILL, PWSTUN, SCREEN_BUMP, SCREEN_SHAKE, SMOKE_PUFF, SOUND_BURST, STORM, STRIKE_HOLY, SUMMON_CELESTIAL, SUMMON_GATE, SUMMON_MONSTER_1, SUMMON_MONSTER_2, SUMMON_MONSTER_3, SUMMON_UNDEAD, SUNBEAM, TIME_STOP, WAIL_O_BANSHEES, WEIRD, WORD

GAS_EXPLOSION_

ACID, EVIL, FIRE, GREASE, MIND, NATURE

HOWL_

MIND, ODD, WAR_CRY, WAR_CRY_FEMALE

LOS_

EVIL_10, EVIL_20, EVIL_30, HOLY_10, HOLY_20, HOLY_30, NORMAL_10, NORMAL_20, NORMAL_30

VFX_IMP_

AC_BONUS, ACID_L, ACID_S, BLIND_DEAF_M, BREACH, CHARM, CONFUSION_S, DAZED_S, DEATH, DEATH_L, DEATH_WARD, DESTRUCTION, DISEASE_S, DISPEL, DISPEL_DISJUNCTION, DIVINE_STRIKE_FIRE, DIVINE_STRIKE_HOLY, DOMINATE_S, DOOM, ELEMENTAL_PROTECTION, EVIL_HELP, FEAR_S, FLAME_M, FLAME_S, FORTITUDE_SAVING_THROW_USE, FROST_L, FROST_S, GLOBE_USE, GOOD_HELP, GREASE, HARM, HASTE, HEALING_G, HEALING_L, HEALING_M, HEALING_S, HEALING_X, HOLY_AID, IMPROVE_ABILITY_SCORE, KNOCK, LIGHTNING_M, LIGHTNING_S, MAGBLUE, MAGIC_PROTECTION, MAGIC_RESISTANCE_USE, MAGICAL_VISION, MIRV, MIRV_FLAME, NEGATIVE_ENERGY, POISON_L, POISON_S, POLYMORPH, RAISE_DEAD, REDUCE_ABILITY_SCORE, REFLEX_SAVE_THROW_USE, REMOVE_CONDITION, RESTORATION, RESTORATION_GREATER, RESTORATION_LESSER, SILENCE, SLEEP, SLOW, SONIC, SPELL_MANTLE_USE, SPIKE_TRAP, STUN, SUNSTRIKE, SUPER_HEROISM, UNSUMMON, WILL_SAVING_THROW_USE

AURA_

FEAR, HOLY, NEGATIVE_ENERGY, UNEARTHLY,

HEAD_

ACID, COLD, ELECTRICITY, EVIL, FIRE, HEAL, HOLY, MIND, NATURE, ODD, SONIC

PULSE_

COLD, FIRE, HOLY, NATURE, NEGATIVE, WATER, WIND

How Actions Work

The 'Action' prefix on a function means that it doesn't directly do anything, but rather adds it to the end of a creature's action queue. This allows you to run a script once to set up a series of actions on a creature that can keep him busy almost indefinitely if you'd like.

For instance:

```
ActionMoveToObject( oPC );
ActionSpeakString( "I have you now!" );
ActionCastSpellAtObject( SPELL_BURNING_HANDS, oPC );
ActionAttack( oPC );
```

When this script is run the actions will be added to the back of the queue in the order that they are called. The creature will continually pop the frontmost action and attempt to execute it. When it has completed (or failed) that action, it will pop the next one and keep going. The result is that the creature will walk from wherever he is to the PC, speak a cheesy line of dialog, cast a spell, and then attack the PC until either the PC dies, the creature dies, or a **ClearAllActions()** command is called on him.

The AssignCommand function is used to add actions to the queues of creatures other than the one executing the script. [*rkc: You can use ActionDoCommand() to add to your own action queue*] This is useful for synchronizing the action, or just simplifying it so that one script can coordinate several creatures.

Example:

```
AssignCommand( oHenchman, ActionAttack( oVillain ) );
```

The second parameter is an 'action' type parameter- which means any void returning function.

Noel Borstad
BioWare

```
int abs(int nValue)
float acos(float fValue)
void ActionAttack(object oAttacker, int bPassive=FALSE)
void ActionCastFakeSpellAtLocation(int nSpellID, location lTarget, int nProjPathType=PROJECTILE_PATH_TYPE_DEFAULT)
void ActionCastFakeSpellAtObject(int nSpellID, object oTarget, int nProjPathType=PROJECTILE_PATH_TYPE_DEFAULT)
void ActionCastSpellAtLocation(int nSpellID, location lTarget, int nMetaMagic=METAMAGIC_ANY, int bCheat=FALSE, int nProjectilePathType=PROJECTILE_PATH_TYPE_DEFAULT, int bInstantSpell=FALSE)
void ActionCastSpellAtObject(int nSpellID, object oTarget, int nMetaMagic=METAMAGIC_ANY, int bCheat=FALSE, int nDomainLevel=0, int nProjectilePathType=PROJECTILE_PATH_TYPE_DEFAULT, int bInstantSpell=FALSE)
void ActionCloseDoor(object oDoor)
void ActionDoCommand(action aActionToDo)
void ActionEquipItem(object oItem, int nInventorySlot)
void ActionEquipMostDamagingMelee(object oVersus=OBJECT_INVALID, int bOffHand=FALSE)
void ActionEquipMostDamagingRanged(object oVersus=OBJECT_INVALID)
void ActionEquipMostEffectiveArmor()
void ActionForceFollowObject(object oFollow, float fFollowDistance=0.0f)
void ActionForceMoveToLocation(location lDestination, int bRun=FALSE, float fTimeout=30.0f)
void ActionForceMoveToObject(object oMoveTo, int bRun=FALSE, float fRange=1.0f, float fTimeout=30.0f)
void ActionGiveItem(object oItem, object oGiveTo)
void ActionInteractObject(object oPlaceable)
void ActionJumpToLocation(location lLoc)
void ActionJumpToObject(object oToJumpTo, int nWalkStraightLineToPoint=1)
void ActionLockObject(object oTarget)
void ActionMoveAwayFromLocation(location lMoveAwayFrom, int bRun=FALSE, float fMoveAwayRange=40.0f)
void ActionMoveAwayFromObject(object oFleeFrom, int bRun=FALSE, float fMoveAwayRange=40.0f)
void ActionMoveToLocation(location lDestination, int bRun=FALSE)
void ActionMoveToObject(object oMoveTo, int bRun=FALSE, float fRange=1.0f)
void ActionOpenDoor(object oDoor)
void ActionPauseConversation()
void ActionPickUpItem(object oItem)
void ActionPlayAnimation(int nAnimation, float fSpeed=1.0, float fSeconds=0.0)
void ActionPutDownItem(object oItem)
void ActionRandomWalk()
void ActionRest()
void ActionResumeConversation()
void ActionSit(object oChair)
void ActionSpeakString(string sStringToSpeak, int nTalkVolume=TALKVOLUME_TALK)
void ActionSpeakStringByStringRef(int nStringRef, int nTalkVolume=TALKVOLUME_TALK)
void ActionStartConversation(object oObjectToConverse, string sDialogResRef, int bPrivateConversation=FALSE)
void ActionTakeItem(object oItem, object oTakeFrom)
void ActionUnequipItem(object oItem)
void ActionUnlockObject(object oTarget)
void ActionUseFeat(int nFeat, object oTarget)
void ActionUseSkill(int nSkill, object oTarget)
void ActionUseTalentAtLocation(talent lChosenTalent, location lTarget)
void ActionUseTalentOnObject(talent lChosenTalent, object oTarget)
void ActionWait(float fSeconds)
void ActivatePortal(object oTarget, string sIPAddress, string sPassword, string sWaypointTag, int bSeemless=FALSE)
void AddHenchman(object oMaster, object oHenchman=OBJECT_SELF)
void AddJournalQuestEntry(string szPlotID, int nState, object oObject, int bAllPartyMembers=TRUE, int bAllPlayers=FALSE, int bAllowOverrideHigher=FALSE)
void AddJournalWorldEntry(int nIndex, string szEntry, string szTitle=WorldEntry)
void AddJournalWorldEntryStringRef(int nStrRef, int nStrRefTitle)
void AdjustAlignment(object oSubject, int nAlignment, int nShift)
void AdjustReputation(object oTargetCreature, object oMemberOfSourceFaction, int nAdjustment)
void AmbientSoundChangeDay(object oArea, int nTrack)
```

```
void AmbientSoundChangeNight(object oArea, int nTrack)
void AmbientSoundPlay(object oArea)
void AmbientSoundStop(object oArea)
vector AngleToVector(float fAngle)
void ApplyEffectAtLocation(int nDurationType, effect e, location l, float fDuration=0.0f)
void ApplyEffectToObject(int nDurationType, effect e, object oTarget, float fDuration=0.0f)
float asin(float fValue)
void AssignCommand(object oTarget, action aActionToAssign)
float atan(float fValue)
int BeginConversation(string sResRef, object oObjectToDialog=OBJECT_INVALID)
void ChangeFaction(object oObjectToChangeFaction, object oMemberOfFactionToJoin)
void ChangeToStandardFaction(object oCreatureToChange, int nStandardFaction)
void ClearAllActions()
void ClearPersonalReputation(object oTarget, object oSource=OBJECT_SELF)
float cos(float fValue)
object CreateItemOnObject(string sItemTemplate, object oTarget=OBJECT_SELF, int nStackSize=1)
object CreateObject(int nObjectType, string sTemplate, location lLoc, int bUseAppearAnimation=FALSE)
int d2(int nDice=1)
int d3(int nDice=1)
int d4(int nDice=1)
int d6(int nDice=1)
int d8(int nDice=1)
int d10(int nDice=1)
int d12(int nDice=1)
int d20(int nDice=1)
int d100(int nDice=1)
void DelayCommand(float fSeconds, action aActionToDelay)
void DeleteJournalWorldAllEntries()
void DeleteJournalWorldEntry(int nIndex)
void DeleteJournalWorldEntryStringRef(int nStrRef)
void DeleteLocalFloat(object oObject, string sVarName)
void DeleteLocalInt(object oObject, string sVarName)
void DeleteLocalLocation(object oObject, string sVarName)
void DeleteLocalObject(object oObject, string sVarName)
void DeleteLocalString(object oObject, string sVarName)
void DestroyObject(object oDestroy, float fDelay=0.0f)
void DoDoorAction(object oTargetDoor, int nDoorAction)
void DoPlaceableObjectAction(object oTargetObject, int nPlaceableAction)
void DoSinglePlayerAutoSave()
void E3PlaySound(string sFileName, location lSoundLocation, float fGain, float fRefrDistance, float fRollOffFactor)
effect EffectAbilityDecrease(int nAttribute, int nModifyBy)
effect EffectAbilityIncrease(int nAttribute, int nModifyBy)
effect EffectACDecrease(int nValue, int nModifyType=AC_DODGE_BONUS)
effect EffectACIncrease(int nValue, int nModifyType=AC_DODGE_BONUS)
effect EffectAppear()
effect EffectAreaOfEffect(int nAreaEffectId, string sOnEnterScript, string sHeartbeatScript, string sOnExitScript)
effect EffectAttackDecrease(int nBonus, int nModifierType=ATTACK_BONUS_MISC)
effect EffectAttackIncrease(int nBonus, int nModifierType=ATTACK_BONUS_MISC)
effect EffectBeam(int nBeamVisualEffect, object oEffector, int nBodyPart, int nMissEffect=FALSE)
effect EffectBlindness()
effect EffectCharmed()
effect EffectConcealment(int nPercentage)
effect EffectConfused()
effect EffectCurse(int nStrMod=1, int nDexMod=1, int nConMod=1, int nIntMod=1, int nWisMod=1, int nChaMod=1)
effect EffectDamage(int nDamage, int nDamageType=DAMAGE_TYPE_MAGICAL, int nDamagePower=DAMAGE_POWER_NORMAL)
effect EffectDamageDecrease(int nBonus, int nDamageType=DAMAGE_TYPE_MAGICAL)
effect EffectDamageImmunityDecrease(int nDamageType, int nPercentImmunity)
effect EffectDamageImmunityIncrease(int nDamageType, int nPercentImmunity)
effect EffectDamageIncrease(int nValue, int nDamageType=DAMAGE_TYPE_MAGICAL)
effect EffectDamageReduction(int nAmount, int nDamagePower, int nLimit=0)
effect EffectDamageResistance(int nDamageType, int nAmount, int nLimit=0)
effect EffectDamageShield(int nDamageAmount, int nRandomAmount, int nDamageType)
effect EffectDarkness()
effect EffectDazed()
effect EffectDeaf()
effect EffectDeath(int nSpectacularDeath=FALSE, int nDisplayFeedback=TRUE)
effect EffectDisappear()
effect EffectDisappearAppear(location lLocation)
effect EffectDisease(int nDiseaseType)
effect EffectDispelMagicAll(int nCasterLevel)
effect EffectDispelMagicBest(int nCasterLevel)
effect EffectDominated()
effect EffectEntangle()
effect EffectFrightened()
effect EffectHaste()
effect EffectHeal(int nDamage)
effect EffectHitPointChangeWhenDying(float fHitPointChangePerRound)
effect EffectImmunity(int nImmunityType)
effect EffectInvisibility(int nInvisibilityType)
effect EffectKnockdown()
effect EffectLinkEffects(effect eLinkThis, effect eToThis)
effect EffectMissChance(int nPercentage)
effect EffectModifyAttacks(int nAttacks)
effect EffectMovementSpeedDecrease(int nPercentChange)
effect EffectMovementSpeedIncrease(int nPercentChange)
effect EffectNegativeLevel(int nNumLevels)
effect EffectParalyze()
effect EffectPoison(int nPoisonType)
effect EffectPolymorph(int nPolymorphSelection)
```

effect	EffectRegenerate(int nAmount, float fInterval)	float	GetFacing(object oTarget)
effect	EffectResurrection()	float	GetFacingFromLocation(location l)
effect	EffectSanctuary(int nDifficultyClass)	int	GetFactionAverageGoodEvilAlignment(object oFactionMember)
effect	EffectSavingThrowDecrease(int nSave, int nValue, int nSaveType=SAVING_THROW_TYPE_ALL)	int	GetFactionAverageLawChaosAlignment(object oFactionMember)
effect	EffectSavingThrowIncrease(int nSave, int nValue, int nSaveType=SAVING_THROW_TYPE_ALL)	int	GetFactionAverageLevel(object oFactionMember)
effect	EffectSeelInvisible()	int	GetFactionAverageReputation(object oSource, object oTarget)
effect	EffectSilence()	int	GetFactionAverageXP(object oFactionMember)
effect	EffectSkillDecrease(int nSkill, int nValue)	object	GetFactionBestAC(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectSkillIncrease(int nSkill, int nValue)	int	GetFactionEqual(object oFirstObject, object oSecondObject=OBJECT_SELF)
effect	EffectSleep()	int	GetFactionGold(object oFactionMember)
effect	EffectSlow()	object	GetFactionLeastDamagedMember(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectSpellImmunity(int nSpellLevel=-1, int nSpellSchool=SPELL_SCHOOL_GENERAL, int nSpell=-1)	object	GetFactionMostDamagedMember(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectSpellLevelAbsorption(int nMaxLevelAbsorbed, int nTotalLevelsAbsorbed=0, int nSchool=SPELL_SCHOOL_GENERAL)	int	GetFactionMostFrequentClass(object oFactionMember)
effect	EffectSpellResistanceDecrease(int nValue)	object	GetFactionStrongestMember(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectSpellResistanceIncrease(int nValue)	object	GetFactionWeakestMember(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectStunned()	object	GetFactionWorstAC(object oFactionMember=OBJECT_SELF, int bMustBeVisible=TRUE)
effect	EffectSummonCreature(string sResref, int nVisualEffectId=VFX_NONE, float fDelay=0.0f)	int	GetFamiliarCreatureType(object oTarget)
effect	EffectSwarm(int nLooping, string sTemplate1, string sTemplate2=, string sTemplate3=, string sTemplate4=)	string	GetFamiliarName(object oTarget)
effect	EffectTemporaryHitpoints(int nHitPoints)	effect	GetFirstEffect(object oCreature)
effect	EffectTimeStop()	object	GetFirstFactionMember(object oMemberOfFaction, int bPCOnly=TRUE)
effect	EffectTrueSeeing()	object	GetFirstInPersistentObject(object oPersistentObject=OBJECT_SELF, int nResidentObjectType=OBJECT_TYPE_CREATURE, int nPersZone=PERSISTENT_ZONE_ACTIVE)
effect	EffectTurned()	object	GetFirstItemInInventory(object oTarget=OBJECT_SELF)
effect	EffectTurnResistanceDecrease(int nHitDice)	object	GetFirstObjectInArea(object oArea=OBJECT_INVALID)
effect	EffectTurnResistanceIncrease(int nHitDice)	object	GetFirstObjectInShape(int nShape, float fSize, location lTarget, int bLineOfSight=FALSE, int nObjectFilter=OBJECT_TYPE_CREATURE, vector vOrigin=[0.0,0.0,0.0])
effect	EffectUltravision()	object	GetFirstPC()
effect	EffectVisualEffect(int nVisualEffectId, int nMissEffect=FALSE)	int	GetFortitudeSavingThrow(object oTarget)
event	EventActivateItem(object oItem, location lTarget, object oTarget=OBJECT_INVALID)	int	GetGameDifficulty()
event	EventConversation()	int	GetGender(object oCreature)
event	EventSpellCastAt(object oCaster, int nSpellId, int bHarmful=TRUE)	object	GetGoingToBeAttackedBy(object oTarget)
event	EventUserDefined(int nUserDefinedEventNumber)	int	GetGold(object oTarget=OBJECT_SELF)
void	ExecuteScript(string sScript, object oTarget)	int	GetGoldPieceValue(object oItem)
void	ExploreAreaForPlayer(object oArea, object oPlayer)	int	GetGoodEvilValue(object oCreature)
void	ExportAllCharacters()	int	GetHasFeat(int nFeat, object oCreature=OBJECT_SELF)
effect	ExtraordinaryEffect(effect e)	int	GetHasFeatEffect(int FeatId, object oObject=OBJECT_SELF)
float	fabs(float fValue)	int	GetHasSkill(int nSkill, object oCreature=OBJECT_SELF)
float	FeetToMeters(float fFeet)	int	GetHasSpell(int nSpellId, object oCreature=OBJECT_SELF)
int	FindSubString(string sString, string sSubString)	int	GetHasSpellEffect(int nSpellId, object oObject=OBJECT_SELF)
void	FloatingTextStringOnCreature(string sStringToFloat, object oCreatureFloatAbove, int bBroadcastToFaction=TRUE)	object	GetHenchman(object oMaster=OBJECT_SELF)
void	FloatingTextStrRefOnCreature(int nStrRefToFloat, object oCreatureFloatAbove, int bBroadcastToFaction=TRUE)	int	GetHitDice(object oCreature)
int	FloorToInt(float fFloat)	int	GetIdentified(object oItem)
string	FloatToString(float fFloat, int nWidth=18, int nDecimals=9)	int	GetIdFromTalent(talent tTalent)
int	FortitudeSave(object oCreature, int nDC, int nSaveType=SAVING_THROW_TYPE_NONE, object oSaveVersus=OBJECT_SELF)	object	GetInventoryDisturbItem()
int	GetAbilityModifier(int nAbility, object oCreature=OBJECT_SELF)	int	GetInventoryDisturbType()
int	GetAbilityScore(object oCreature, int nAbility)	int	GetIsDown()
int	GetAC(object oObject, int nVersusType=RACIAL_TYPE_ALL)	int	GetIsDay()
int	GetAge(object oTarget)	int	GetIsDead(object oCreature)
int	GetAlignmentGoodEvil(object oCreature)	int	GetIsDM(object oCreature)
int	GetAlignmentLawChaos(object oCreature)	int	GetIsDoorActionPossible(object oTargetDoor, int nDoorAction)
int	GetAnimalCompanionCreatureType(object oTarget)	int	GetIsDusk()
string	GetAnimalCompanionName(object oTarget)	int	GetIsEffectValid(effect eEffect)
object	GetArea(object oTarget)	int	GetIsEncounterCreature(object oCreature=OBJECT_SELF)
object	GetAreaFromLocation(location l)	int	GetIsEnemy(object oTarget, object oSource=OBJECT_SELF)
object	GetAreaOfEffectCreator(object oAreaOfEffectObject=OBJECT_SELF)	int	GetIsFriend(object oTarget, object oSource=OBJECT_SELF)
object	GetAssociate(int nAssociateType, object oMaster=OBJECT_SELF)	int	GetIsImmune(object oCreature, int nImmunityType, object oVersus=OBJECT_INVALID)
object	GetAttackTarget(object oTarget=OBJECT_SELF)	int	GetIsInCombat(object oCreature=OBJECT_SELF)
object	GetAttemptedAttackTarget()	int	GetIsListening(object oObject)
object	GetAttemptedSpellTarget()	int	GetIsNeutral(object oTarget, object oSource=OBJECT_SELF)
int	GetBaseItemType(object oItem)	int	GetIsNight()
object	GetBlockingDoor()	int	GetIsObjectValid(object oObject)
int	GetCalendarDay()	int	GetIsOpen(object oObject)
int	GetCalendarMonth()	int	GetIsPC(object oCreature)
int	GetCalendarYear()	int	GetIsPlaceableObjectActionPossible(object oTargetObject, int nPlaceableAction)
int	GetCasterLevel(object oCreature)	int	GetIsPlayableRacialType(object oCreature)
float	GetChallengeRating(object oTarget)	int	GetIsReactionTypeFriendly(object oTarget, object oSource=OBJECT_SELF)
int	GetClassByPosition(int nClassPosition, object oCreature=OBJECT_SELF)	int	GetIsReactionTypeHostile(object oTarget, object oSource=OBJECT_SELF)
object	GetClickingObject()	int	GetIsReactionTypeNeutral(object oTarget, object oSource=OBJECT_SELF)
int	GetCommandable(object oTarget=OBJECT_SELF)	int	GetIsResting(object oCreature=OBJECT_SELF)
int	GetCreatureHasTalent(talent tTalent, object oCreature=OBJECT_SELF)	int	GetIsTalentValid(talent tTalent)
int	GetCreatureSize(object oCreature)	int	GetIsTrapped(object oTrap)
talent	GetCreatureTalentBest(int nCategory, int nCRMax, object oCreature=OBJECT_SELF)	void	GetIsWeaponEffective(object oVersus=OBJECT_INVALID, int bOffHand=FALSE)
talent	GetCreatureTalentRandom(int nCategory, object oCreature=OBJECT_SELF)	object	GetItemActivated()
int	GetCurrentAction(object=OBJECT_SELF)	object	GetItemActivatedTarget()
int	GetCurrentHitPoints(object oObject=OBJECT_SELF)	location	GetItemActivatedTargetLocation()
int	GetDamageDealtByType(int nDamageType)	object	GetItemActivator()
string	GetDeity(object oTarget)	int	GetItemACValue(object oItem)
float	GetDistanceBetween(object oObjectA, object oObjectB)	int	GetItemHasItemProperty(object oItem, int nProperty)
float	GetDistanceBetweenLocations(location lLocationA, location lLocationB)	object	GetItemInSlot(int nInventorySlot, object oCreature=OBJECT_SELF)
float	GetDistanceToObject(object oObject)	object	GetItemPossessedBy(object oCreature, string sItemTag)
object	GetEffectCreator(effect eEffect)	object	GetItemPossessor(object oItem)
int	GetEffectDurationType(effect eEffect)	int	GetJournalQuestExperience(string szPlotID)
int	GetEffectSpellId(effect eSpellEffect)	int	GetLastAssociateCommand(object oAssociate=OBJECT_SELF)
int	GetEffectSubType(effect eEffect)	object	GetLastAttacker(object oAttacker=OBJECT_SELF)
int	GetEffectType(effect e)	int	GetLastAttackMode(object oTarget=OBJECT_SELF)
int	GetEncounterActive(object oEncounter=OBJECT_SELF)	int	GetLastAttackType(object oTarget=OBJECT_SELF)
int	GetEncounterDifficulty(object oEncounter=OBJECT_SELF)	object	GetLastDamager()
int	GetEncounterSpawnsCurrent(object oEncounter=OBJECT_SELF)	object	GetLastDisarmed()
int	GetEncounterSpawnsMax(object oEncounter=OBJECT_SELF)	object	GetLastDisturbed()
object	GetEnteringObject()		
object	GetExitingObject()		

object	GetLastHostileActor(object oVictim=OBJECT_SELF)	int	GetStandardFactionReputation(int nStandardFactionId, object oCreatures=OBJECT_SELF)
object	GetLastKiller()	location	GetStartingLocation()
object	GetLastLocked()	string	GetStringByStrRef(int nStrRef)
object	GetLastOpenedBy()	string	GetStringLeft(string sString, int nCount)
object	GetLastPCRested()	int	GetStringLength(string sString)
object	GetLastPerceived()	string	GetStringLowerCase(string sString)
int	GetLastPerceptionHeard()	string	GetStringRight(string sString, int nCount)
int	GetLastPerceptionInaudible()	string	GetStringUpperCase(string sString)
int	GetLastPerceptionSeen()	string	GetSubRace(object oTarget)
int	GetLastPerceptionVanished()	string	GetSubString(string sString, int nStart, int nCount)
object	GetLastPlayerDied()	string	GetTag(object oObject)
object	GetLastPlayerDying()	int	GetTileMainLight1Color(location ITile)
object	GetLastRespawnButtonPresser()	int	GetTileMainLight2Color(location ITile)
int	GetLastRestEventType()	int	GetTileSourceLight1Color(location ITile)
object	GetLastSpeaker()	int	GetTileSourceLight2Color(location ITile)
int	GetLastSpell()	int	GetTimeHour()
object	GetLastSpellCaster()	int	GetTimeMillisecond()
int	GetLastSpellHarmful()	int	GetTimeMinute()
object	GetLastTrapDetected(object oTarget=OBJECT_SELF)	int	GetTimeSecond()
object	GetLastUnlocked()	int	GetTotalDamageDealt()
object	GetLastUsedBy()	object	GetTransitionTarget(object oTransition)
object	GetLastWeaponUsed(object oCreature)	int	GetTrapBaseType(object oTrapObject)
int	GetLawChaosValue(object oCreature)	object	GetTrapCreator(object oTrapObject)
int	GetLevelByClass(int nClassType, object oCreature=OBJECT_SELF)	int	GetTrapDetectable(object oTrapObject)
int	GetLevelByPosition(int nClassPosition, object oCreature=OBJECT_SELF)	int	GetTrapDetectDC(object oTrapObject)
int	GetListenPatternNumber()	int	GetTrapDetectedBy(object oTrapObject, object oCreature)
float	GetLocalFloat(object oObject, string sVarName)	int	GetTrapDisarmable(object oTrapObject)
int	GetLocalInt(object oObject, string sVarName)	int	GetTrapDisarmDC(object oTrapObject)
location	GetLocalLocation(object oObject, string sVarName)	int	GetTrapFlagged(object oTrapObject)
object	GetLocalObject(object oObject, string sVarName)	string	GetTrapKeyTag(object oTrapObject)
string	GetLocalString(object oObject, string sVarName)	int	GetTrapOneShot(object oTrapObject)
location	GetLocation(object oObject)	int	GetTurnResistanceHD(object oUndead=OBJECT_SELF)
int	GetLocked(object oTarget)	int	GetTypeFromTalent(talent ITalent)
int	GetLockKeyRequired(object oObject)	int	GetUserDefinedEventNumber()
int	GetLockKeyTag(object oObject)	object	GetWaypointByTag(string sWaypointTag)
int	GetLockLockable(object oObject)	int	GetWeaponRanged(object oItem)
int	GetLockLockDC(object oObject)	int	GetWillSavingThrow(object oTarget)
int	GetLockUnlockDC(object oObject)	int	GetXP(object oTarget)
object	GetMaster(object oAssociate=OBJECT_SELF)	void	GiveGoldToCreature(object oCreature, int nGP)
string	GetMatchedSubstring(int nString)	void	GiveXPToCreature(object oCreature, int nXPAmount)
int	GetMatchedSubstringsCount()	float	HoursToSeconds(int nHours)
int	GetMaxHitPoints(object oObject=OBJECT_SELF)	string	InsertString(string sDestination, string sString, int nPosition)
int	GetMetaMagicFeat()	float	IntToFloat(int nInteger)
object	GetModule()	string	IntToHexString(int nInteger)
object	GetModuleItemAcquired()	string	IntToString(int nInteger)
object	GetModuleItemAcquiredFrom()	int	IsInConversation(object oObject)
object	GetModuleItemLost()	void	JumpToLocation(location IDestination)
object	GetModuleItemLostBy()	void	JumpToObject(object oToJumpTo, int nWalkStraightLineToPoint=1)
int	GetMovementRate(object oTarget)	location	Location(object oArea, vector vPosition, float fOrientation)
string	GetName(object oObject)	float	log(float fValue)
object	GetNearestCreature(int nFirstCreatureType, int nFirstParameter, object oTarget=OBJECT_SELF, int nNth=1, int nSecondCreatureType=-1, int nSecondParameter=-1, int nThirdCreatureType=-1, int nThirdParameter=-1)	effect	MagicalEffect(effect e)
object	GetNearestCreatureToLocation(int nFirstCreatureType, int nFirstParameter, location l, int nNth=1, int nSecondCreatureType=-1, int nSecondParameter=-1, int nThirdCreatureType=-1, int nThirdParameter=-1)	void	MusicBackgroundChangeDay(object oArea, int nTrack)
object	GetNearestObject(int nObjectType=OBJECT_TYPE_ALL, object oTarget=OBJECT_SELF, int nNth=1)	void	MusicBackgroundChangeNight(object oArea, int nTrack)
object	GetNearestObjectByTag(string sTag, object oTarget=OBJECT_SELF, int nNth=1)	void	MusicBackgroundPlay(object oArea)
object	GetNearestObjectToLocation(int nObjectType, location l, int nNth=1)	void	MusicBackgroundSetDelay(object oArea, int nDelay)
object	GetNearestTrapToObject(object oTarget=OBJECT_SELF, int nTrapDetected=TRUE)	void	MusicBackgroundStop(object oArea)
effect	GetNextEffect(object oCreature)	void	MusicBattleChange(object oArea, int nTrack)
object	GetNextFactionMember(object oMemberOfFaction, int bPCOnly=TRUE)	void	MusicBattlePlay(object oArea)
object	GetNextInPersistentObject(object oPersistentObject=OBJECT_SELF, int nResidentObjectType=OBJECT_TYPE_CREATURE, int nPersistentZone=PERSISTENT_ZONE_ACTIVE)	void	MusicBattleStop(object oArea)
object	GetNextItemInInventory(object oTarget=OBJECT_SELF)	string	ObjectToString(object oObject)
object	GetNextObjectInArea(object oArea=OBJECT_INVALID)	void	OpenStore(object oStore, object oPC, int nBonusMarkUp=0, int nBonusMarkDown=0)
object	GetNextObjectInShape(int nShape, float fSize, location ITarget, int bLineOfSight=FALSE, int nObjectFilter=OBJECT_TYPE_CREATURE, vector vOrigin=[0.0,0.0,0.0])	void	PlayAnimation(int nAnimation, float fSpeed=1.0, float fSeconds=0.0)
object	GetNextPC()	void	PlaySound(string sSoundName)
int	GetNumStackedItems(object oItem)	void	PlayVoiceChat(int nVoiceChatID, object oTarget=OBJECT_SELF)
object	GetObjectByTag(string sTag, int nNth=0)	void	PopUpDeathGUIPanel(object oPC, int bRespawnButtonEnabled=TRUE, int bWaitForHelpButtonEnabled=TRUE, int nHelpStringReference=0, string sHelpString="")
int	GetObjectHeard(object oTarget, object oSource=OBJECT_SELF)	void	PopUpGUIPanel(object oPC, int nGUIPanel)
int	GetObjectSeen(object oTarget, object oSource=OBJECT_SELF)	float	pow(float fValue, float fExponent)
int	GetObjectType(object oTarget)	void	PrintFloat(float fFloat, int nWidth=18, int nDecimals=9)
object	GetPCLevelingUp()	void	PrintInteger(int nInteger)
object	GetPCSpeaker()	void	PrintObject(object oObject)
int	GetPlaceableIllumination(object oPlaceable=OBJECT_SELF)	void	PrintString(string sString)
int	GetPlotFlag(object oTarget=OBJECT_SELF)	void	PrintVector(vector vVector)
vector	GetPosition(object oTarget)	int	Random(int nMaxInteger)
vector	GetPositionFromLocation(location l)	string	RandomName()
int	GetRacialType(object oCreature)	void	RecomputeStaticLighting(object oArea)
int	GetReflexAdjustedDamage(int nDamage, object oCreature, int nDC, int nSaveType=SAVING_THROW_TYPE_NONE, object oSaveVersus=OBJECT_SELF)	int	ReflexSave(object oCreature, int nDC, int nSaveType=SAVING_THROW_TYPE_NONE, object oSaveVersus=OBJECT_SELF)
int	GetReflexSavingThrow(object oTarget)	void	RemoveEffect(object oCreature, effect eEffect)
int	GetReputation(object oSource, object oTarget)	void	RemoveHenchman(object oMaster, object oHenchman=OBJECT_SELF)
object	GetSittingCreature(object oObject)	void	RemoveJournalQuestEntry(string szPlotID, object oObject, int bAllPartyMembers=TRUE, int bAllPlayers=FALSE)
int	GetSkillRank(int nSkill, object oTarget=OBJECT_SELF)	void	RemoveSummonedAssociate(object oMaster, object oAssociate=OBJECT_SELF)
object	GetSpellCastItem()	int	ResistSpell(object oCaster, object oTarget)
int	GetSpellId()	float	RoundsToSeconds(int nRounds)
int	GetSpellSaveDC()	void	SetAreaTransitionBMP(int nAreaTransition, string sCustomBMP=)
location	GetSpellTargetLocation()	void	SetAssociateListenPatterns(object oTarget=OBJECT_SELF)
object	GetSpellTargetObject()	void	SetCalendar(int nYear, int nMonth, int nDay)
		void	SetCameraFacing(float fDirection)
		void	SetCameraMode(object oPlayer, int nCameraMode)
		void	SetCommandable(int bCommandable, object oTarget=OBJECT_SELF)

void	SetCustomToken(int nCustomTokenNumber, string sTokenValue)	float	Sin(float fValue)
void	SetEncounterActive(int nNewValue, object oEncounter=OBJECT_SELF)	void	SoundObjectPlay(object oSound)
void	SetEncounterDifficulty(int nEncounterDifficulty, object oEncounter=OBJECT_SELF)	void	SoundObjectSetPosition(object oSound, vector vPosition)
void	SetEncounterSpawnsCurrent(int nNewValue, object oEncounter=OBJECT_SELF)	void	SoundObjectSetVolume(object oSound, int nVolume)
void	SetEncounterSpawnsMax(int nNewValue, object oEncounter=OBJECT_SELF)	void	SoundObjectStop(object oSound)
void	SetFacing(float fDirection)	void	SpeakOneLinerConversation(string sDialogResRef, object oTokenTarget=OBJECT_TYPE_INVALID)
void	SetFacingPoint(vector vTarget)	void	SpeakString(string sStringToSpeak, int nTalkVolume=TALKVOLUME_TALK)
void	SetIdentified(object oItem, int bIdentified)	float	sqrt(float fValue)
void	SetIsDestroyable(int bDestroyable, int bRaiseable=TRUE, int bSelectableWhenDead=FALSE)	void	StartNewModule(string sModuleName)
void	SetIsTemporaryEnemy(object oTarget, object oSource=OBJECT_SELF, int bDecays=FALSE, float fDurationInSecs=180.0f)	float	StringToFloat(string sNumber)
void	SetIsTemporaryFriend(object oTarget, object oSource=OBJECT_SELF, int bDecays=FALSE, float fDurationInSecs=180.0f)	int	StringToInt(string sNumber)
void	SetIsTemporaryNeutral(object oTarget, object oSource=OBJECT_SELF, int bDecays=FALSE, float fDurationInSecs=180.0f)	void	SummonAnimalCompanion(object oMaster=OBJECT_SELF)
void	SetJournalQuestEntryPicture(string szPlotID, object oObject, int nPicthIndex, int bAllPartyMembers=TRUE, int bAllPlayers=FALSE)	void	SummonFamiliar(object oMaster=OBJECT_SELF)
void	SetListening(object oObject, int bValue)	effect	SupernaturalEffect(effect e)
void	SetListenPattern(object oObject, string sPattern, int nNumber=0)	void	SurrenderToEnemies()
void	SetLocalFloat(object oObject, string sVarName, float fValue)	void	TakeGoldFromCreature(int nAmount, object oCreatureToTakeFrom, int bDestroy=FALSE)
void	SetLocalInt(object oObject, string sVarName, int nValue)	talent	TalentFeat(int nFeatID)
void	SetLocalLocation(object oObject, string sVarName, location lValue)	talent	TalentSkill(int nSkillID)
void	SetLocalObject(object oObject, string sVarName, object oValue)	talent	TalentSpell(int nSpellID)
void	SetLocalString(object oObject, string sVarName, string sValue)	float	tan(float fValue)
void	SetLocked(object oTarget, int bLocked)	int	TestStringAgainstPattern(string sPattern, string sStringToTest)
void	SetMapPinEnabled(object oMapPin, int nEnabled)	int	TouchAttackMelee(object oTarget, int bDisplayFeedback=TRUE)
void	SetPanelButtonFlash(object oPlayer, int nButton, int nEnableFlash)	int	TouchAttackRanged(object oTarget, int bDisplayFeedback=TRUE)
void	SetPlaceableIllumination(object oPlaceable=OBJECT_SELF, int bIlluminate=TRUE)	float	TurnsToSeconds(int nTurns)
void	SetPlotFlag(object oTarget, int nPlotFlag)	vector	Vector(float x=0.0f, float y=0.0f, float z=0.0f)
void	SetStandardFactionReputation(int nStandardFactionId, int nNewReputation, object oCreature=OBJECT_SELF)	float	VectorMagnitude(vector vVector)
void	SetTitleMainLightColor(location lTile, int nMainLight1Color, int nMainLight2Color)	vector	VectorNormalize(vector vVector)
void	SetTitleSourceLightColor(location lTile, int nSourceLight1Color, int nSourceLight2Color)	float	VectorToAngle(vector vVector)
void	SetTime(int nHour, int nMinute, int nSecond, int nMillisecond)	effect	VersusAlignmentEffect(effect eEffect, int nLawChaos=ALIGNMENT_ALL, int nGoodEvil=ALIGNMENT_ALL)
int	SetTrapDetectedBy(object oTrap, object oDetector)	effect	VersusRacialTypeEffect(effect eEffect, int nRacialType)
void	SetTrapDisabled(object oTrap)	effect	VersusTrapEffect(effect eEffect)
void	SetWeather(object oTarget, int nWeather)	int	WillSave(object oCreature, int nDC, int nSaveType=SAVING_THROW_TYPE_NONE, object oSaveVersus=OBJECT_SELF)
void	SetXP(object oTarget, int nXPAmount)	float	YardsToMeters(float fYards)
void	SignalEvent(object oObject, event evToRun)		

```

////////////////////////////////////
// The Game of Life                      Copyright (c) 2001 Bioware Corp.
//
// v1.0 06/24/01 Noel Borstad
// v1.1 06/26/01 Richard Conner  Optimized, removed unneeded LocalInt use
////////////////////////////////////
// If a cell has less than 2, or more than 3 neighbours then it 'dies'
// If a dead one has exactly 3 neighbours, it becomes 'alive'
//
// This script is meant to be placed on the Area as a User Defined Event
// script. The area is scripted to fire an event at itself when a user enters,
// which triggers this script, which will continue to trigger itself
// indefinitely. Every time the script is run it will do one iteration.
//
// At any time, players can walk onto the 'board' and rearrange the patterns
// just by picking up the items that we use as cells, and dropping them in
// approximately the right place.
////////////////////////////////////
void main()
{
    float  fOffset=0.3125f;           // space between cells (length a tile)
    float  fSelfLength=fOffset / 2.0f; // error tolerance for cell placement
    float  fLength=fOffset * 1.7f;    // area to count neighbours in

    float  fIterDelay=0.75f;         // Delay between iterations

    location  lSpawn, lOrigin;
    vector    vSpawn, vOrigin;
    object    oCell, oOtherCell, oArea;
    int       bCellAtPosition, nCount, i, j;

    lOrigin=GetLocation( GetWaypointByTag(Origin) );
    vOrigin=GetPositionFromLocation(lOrigin);
    oArea=    GetAreaFromLocation(lOrigin);

    for (i=0; i<19; i++)
    {
        for (j=0; j<19; j++)
        {
            // Calculate location that this cell should be at
            vSpawn=Vector(i * fOffset,j * fOffset) + vOrigin;
            lSpawn=Location(oArea, vSpawn, 0.0f);

            // Get the cell at this location
            oCell=GetFirstObjectInShape(SHAPE_CUBE, fSelfLength, lSpawn, FALSE, OBJECT_TYPE_ITEM);
            bCellAtPosition=GetIsObjectValid(oCell) && GetTag(oCell) == CELL;

            // Count this location's neighbours
            nCount=0;
            oOtherCell=GetFirstObjectInShape(SHAPE_CUBE, fLength, lSpawn, FALSE, OBJECT_TYPE_ITEM);
            while (nCount < 5 && GetIsObjectValid(oOtherCell) )
            {
                if (oOtherCell != oCell && GetTag(oOtherCell) == CELL)
                {
                    nCount++;
                }
                oOtherCell=GetNextObjectInShape(SHAPE_CUBE, fLength, lSpawn, FALSE, OBJECT_TYPE_ITEM);
            }

            // Do changes if necessary
            if (bCellAtPosition)
            {
                if (nCount < 2 || nCount > 3)
                {
                    DestroyObject(oCell);    // Note: This will occur after script completes
                }
            }
            else
            {
                if (nCount == 3)
                {
                    AssignCommand(OBJECT_SELF, CreateCell(lSpawn));
                }
            }
        }
    }

    // This command will rerun this script in fIterDelay seconds, by creating a
    // user-defined event which, in turn, runs this script!
    DelayCommand(fIterDelay, SignalEvent(OBJECT_SELF, EventUserDefined(0)));
}

// This is simply because CreateObject returns an Object and therefore can't be used directly in an AssignCommand() statement.
void CreateCell(location loc)
{
    CreateObject(OBJECT_TYPE_ITEM, CELL, loc);
}

```



Finally, here's a sample script that we wrote to implement what we call a full-edge area transition. In Baldur's Gate, when you entered a new area, all characters went to a specific point in the new area. The full-edge area transition allows you to move characters to the same relative location in the new area (i.e. if they travel through the north part of a trigger, they appear in the north part of the new area).

```
void main() // The Mighty Teleport Script, (c) BioWare Corp, 2000
{
    object oEnter=GetEnteringObject();
    object oDestArea=GetAreaByString(AREA1);

    location locEnter=GetLocation(oEnter);
    vector vEnter=locEnter.vPosition; // ...or... GetPosition(oEnter)
    vector vDest=Vector(vEnter.x, 2.5);

    float fFacing=locEnter.fFacing; // ...or... ?? GetFacing(oEnter) ??

    location locNew=Location(oDestArea, vDest, fFacing); // East is at 0.0 radians.

    AssignCommand(oEnter, ClearAllActions());
    AssignCommand(oEnter, JumpToLocation(locNew));
}
```

Note: This was modified to (hopefully) approximate use of new JumpToLocation(loc).

We have a trigger object that covers the edge of an area, and this script is run whenever an object enters the trigger. **GetEnteringObject()** retrieves the ID of the object that entered the trigger. Each **AssignCommand()** function then executes commands on the entering object.

The first **AssignCommand()** removes the object's list of AI actions (i.e. if it was walking somewhere, stop walking!).

The second **AssignCommand()** sets location and facing for the object.

```
void BattleCry(string sCry1, string sCry2)

void main()
{
    int nCount;
    object oTarget= GetNearestCreature(CREATURE_TYPE_PLAYER_CHAR, PLAYER_CHAR_IS_PC)

    if (GetIsObjectValid(oTarget))
    {
        if (GetDistanceToObject(oTarget) < 40.0f && GetIsDead(oTarget)==FALSE)
        {
            if (GetLocalInt(OBJECT_SELF, FIREBALL) == 0)
            {
                ActionCastSpellAtObject(SPELL_FIREBALL, oTarget);
                SetLocalInt(OBJECT_SELF, FIREBALL, 1);
            }
            else if (GetLocalInt(OBJECT_SELF, WALK) == 0 )
            {
                ActionMoveToLocation( GetLocalLocation(OBJECT_SELF, ShamanStart), TRUE);
                SetLocalInt(OBJECT_SELF, WALK, 1);
            }
            else if (GetLocalInt(OBJECT_SELF, MAGIC_MISSILE) < 3)
            {
                ActionCastSpellAtObject (SPELL_MAGIC_MISSILE, oTarget);
                nCount=GetLocalInt(OBJECT_SELF, MAGIC_MISSILE);
                nCount++;
                SetLocalInt(OBJECT_SELF, MAGIC_MISSILE, nCount);
            }
            else
            {
                ActionAttack(oTarget);
            }
            BattleCry(goblin_bc1_tmp,goblin_bc2_tmp);
        }
    }
}
```

Mark Brockington on user-defined classes, and array simulation:

Java-style classes (with member functions) within the scripting language are still not supported yet. [Hmm... something we might get? ☺] We have control of the major engine classes, and you can think of each script as the implementation of a member function for those classes.

... and on how to simulate array functionality:

```
int GetLocalArrayInt(object oidObject, string sVarName, int nVarNum)
{
    string sFullVarName=sVarName + IntToString(nVarNum);
    return GetLocalInt(oidObject, sFullVarName);
}
```

```
void SetLocalArrayInt(object oidObject, string sVarName, int nVarNum, int nValue)
{
    string sFullVarName=sVarName + IntToString(nVarNum);
    SetLocalInt(oidObject, sFullVarName, nValue);
}
```

The UserDefinedEvent is Your Friend *(...from the Mighty Dave Gaider... @)*

Seriously. No joke at all here. There will naturally be some users who will prefer to go deeper and customize the various scripts that make up the generic AI... some will even go further than that and come up with their own versions of the generic AI include file. That's great, and there's nothing wrong with doing that.

The great thing is: you don't have to. 98% of the creature scripting that was done in the NWN official campaign was done using UserDefinedEvents. I sat down with Preston (one of our scripters and one of the main guys behind the generic AI) and fired off a rapid series of things I would like to script... just about every one was do-able using the UserDefinedEvent. And it is easy to implement.

Let's back up for a second here and discuss what an event is. Basically there are numerous scripts (sets of instructions) that exist on a creature... and each one is tied to events. When an event occurs, it tells the creature to fire off the proper script. No event=the script doesn't run.

There are a number of events which get mentioned here a lot. The 'OnHeartbeat' event fires every 6 seconds... so the script attached to that even runs that often. The 'OnPerceived' event fires whenever the NPC perceives anything... the script attached gets run. These scripts can even fire off other events, prompting different scripts to be run... and therein lies the UserDefinedEvent.

Basically, the UserDefinedEvent allows you to access the normal events a creature uses (OnPerception, OnDeath, etc.) without disturbing the normal AI routines or modifying the already-existing scripts.

Using the UserDefinedEvent

There is one script on every creature that has been mentioned lots before, and it's one you will get to know well. This is the OnSpawn script. This is an event that runs only once when the creature first comes into existence... think of it as an 'initializing' script.

The generic OnSpawn script has a whole series of commands in it which are commented out... as a matter of fact, without the user changing anything in it, all that large script really does is three commands: it tells the creature to walk waypoints, set up its listening patterns and generate a small amount of generic treasure in inventory (some gold, etc.)

It is designed to allow the user to set states that access the generic AI, as they wish. All they do is go through the list and uncomment any state flags they wish to use. Then they simply save the script under a different name and use it in place of the regular OnSpawn script... voila, the generic behaviors have been altered.

The state flags we need to pay attention to are these:

```
SetSpawnInCondition(NW_FLAG_PERCEIVE_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1002
SetSpawnInCondition(NW_FLAG_ATTACK_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1005
SetSpawnInCondition(NW_FLAG_DAMAGED_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1006
SetSpawnInCondition(NW_FLAG_DISTURBED_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1008
SetSpawnInCondition(NW_FLAG_END_COMBAT_ROUND_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1003
SetSpawnInCondition(NW_FLAG_ON_DIALOGUE_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1004
SetSpawnInCondition(NW_FLAG_DEATH_EVENT) //OPTIONAL BEHAVIOR - Fire User Defined Event 1007
```

What Next?

Okay... you've uncommented the flags you want to script for and re-saved the OnSpawn script. That's taken you 2 minutes. Now how do you use it?

Every creature has an OnUserDefined section that you can put a script in. For any creature you make, it will be completely blank... this is where you do your actual scripting. Any script you put here will be 'checked' every time an event of the type you uncommented occurs. So if you uncomment the SetSpawnInCondition(NW_FLAG_DISTURBED_EVENT), then every time the creature's inventory was disturbed (say someone pickpockets an item), the OnUserDefined script would be run.

The basic OnUserDefined script would look like this:

```
void main()
{
    int nUser=GetUserDefinedEventNumber();
    if (nUser == the [constant] in the OnSpawn, [NW_FLAG_ATTACK_EVENT, etc])
    {
        // -- do something here --
    }
}
```

Let's say I want to make a creature wave to the first PC that he sees. I go into the OnSpawn and uncomment SetSpawnInCondition(NW_FLAG_PERCEIVE_EVENT) and then re-save it as a new script.

Then I create a script for the OnUserDefined perhaps as so:

```
void main()
{
    int nUser=GetUserDefinedEventNumber();
    if (nUser == NW_FLAG_PERCEIVE_EVENT)
    {
        object oCreature=GetLastPerceived();
        int iWave=GetLocalInt(OBJECT_SELF,Wave_Once)
        //check to see if I actually saw a PC
        if ((oCreature == GetLastPerceptionSeen()) && (GetIsPC(oCreature)) && (iWave == 0))
        {
            // wave only once
            SetLocalInt(OBJECT_SELF,Wave_Once,1);
            // turn to face the PC
            ActionDoCommand(SetFacingPoint(GetPosition(oCreature)));
            // wave howdy
            ActionPlayAnimation(ANIMATION_FIREFORGET_GREET);
            SpeakString("Hello!")
        }
    }
}
```

And that's it. I could also set other checks like a minimum distance, see if the PC is facing me... or get as complex a behavior as I wish.

Here's another, more complicated, example. This is a barmaid who will select customers in the bar... she will move between them and the bar, speaking dialogue (What would you like?, Here you go. and so forth) from a dialogue file that isn't the one normally attached to her (the one used when the player speaks to her).

```
void main()
{
    int nUser=GetUserDefinedEventNumber();
    int nRandom=d8();
    object oCustomer=GetLocalObject(OBJECT_SELF, CUSTOMER);
    object oBar=GetWaypointByTag(Bar);

    if (nUser == NW_FLAG_HEARTBEAT_EVENT) // This is a heartbeat event, every 6 sec
    {
        if (!GetIsObjectValid(oCustomer))
        {
            // Randomly seek out up to the 8th-nearest non-PC
            oCustomer=GetNearestCreature(CREATURE_TYPE_PLAYER_CHAR, PLAYER_CHAR_NOT_PC, OBJECT_SELF, nRandom);
            if (oCustomer != OBJECT_SELF && GetIsObjectValid(oCustomer) )
            {
                // Move to Customer
                SetLocalInt(OBJECT_SELF, BARMAID_STATE, 1);
                SetLocalObject(OBJECT_SELF, CUSTOMER, oCustomer);
                ActionMoveToObject(oCustomer);
                ActionDoCommand(SpeakOneLinerConversation(WaitTables);
                ActionWait(1.0);

                // Move to Bar
                ActionDoCommand(SetLocalInt(OBJECT_SELF, BARMAID_STATE, 2));
                ActionMoveToObject(oBar);
                ActionDoCommand(SpeakOneLinerConversation(WaitTables);
                ActionWait(1.0);

                // Move back to the customer
                ActionDoCommand(SetLocalInt(OBJECT_SELF, BARMAID_STATE, 3));
                ActionMoveToObject(oCustomer);
                ActionDoCommand(SpeakOneLinerConversation(WaitTables);
                ActionWait(3.0);
                ActionDoCommand(SetLocalObject(OBJECT_SELF, CUSTOMER, OBJECT_INVALID));
            }
        }
    }
    if (nUser == NW_FLAG_ON_DIALOGUE_EVENT) // this is the OnDialogue event
    {
        // Reset my actions
        SetLocalObject(OBJECT_SELF, CUSTOMER, OBJECT_INVALID);
        SetLocalInt(OBJECT_SELF, BARMAID_STATE, 0);
    }
}
```