

```

1  Scriptname ActiveMagicEffect Hidden
2
3  ; Add an inventory event filter to this effect. Item added/removed events matching the
4  ; specified form (or in the specified form list) will now be let through.
5  Function AddInventoryEventFilter(Form akFilter) native
6
7  ; Dispel this effect
8  Function Dispel() native
9
10 ; Get the base MagicEffect this active effect is using
11 MagicEffect Function GetBaseObject() native
12
13 ; Get the actor that cast this spell
14 Actor Function GetCasterActor() native
15
16 ; Get the actor this spell is targeting (is attached to)
17 Actor Function GetTargetActor() native
18
19 ; Register for the specified animation event from the specified object - returns true if
20 ; it successfully registered
21 bool Function RegisterForAnimationEvent(ObjectReference akSender, string asEventName)
22 ; native
23
24 ; Register for LOS gain and lost events between the viewer and the target
25 ; A loss or gain event will be sent immediately, depending on whether or not the viewer
26 ; is already looking at the target or not
27 ; If the viewer is not the player, the target must be another actor
28 Function RegisterForLOS(Actor akViewer, ObjectReference akTarget) native
29
30 ; Register for only the first LOS gain event between the viewer and the target
31 ; If the viewer is already looking at the target, an event will be received almost
32 ; immediately
33 ; If the viewer is not the player, the target must be another actor
34 Function RegisterForSingleLOSGain(Actor akViewer, ObjectReference akTarget) native
35
36 ; Register for only the first LOS lost event between the viewer and the target
37 ; If the viewer is already not looking at the target, an event will be received almost
38 ; immediately
39 ; If the viewer is not the player, the target must be another actor
40 Function RegisterForSingleLOSLost(Actor akViewer, ObjectReference akTarget) native
41
42 ; Register for a single OnUpdate event, in afInterval seconds. All scripts attached to
43 ; this magic effect will get the update events
44 ; Of course, this means you don't need to call UnregisterForUpdate()
45 ; If you find yourself doing this:
46 ; Event OnUpdate()
47 ;     UnregisterForUpdate()
48 ;     {Do some stuff}
49 ; endEvent
50 ; Then you should use RegisterForSingleUpdate instead
51 Function RegisterForSingleUpdate(float afInterval) native
52
53 ; Registers this magic effect to receive events when the player sleeps and wakes up
54 Function RegisterForSleep() native
55
56 ; Registers this alias to receive events when tracked stats are updated
57 Function RegisterForTrackedStatsEvent() native
58
59 ; Register for OnUpdate events, every X seconds, where X is the interval. All scripts
60 ; attached to this magic effect will get the update events
61 Function RegisterForUpdate(float afInterval) native
62
63 ; Register for OnUpdateGameTime events, every X hours of game time, where X is the
64 ; interval. All scripts attached to this magic effect will get the update events
65 Function RegisterForUpdateGameTime(float afInterval) native
66
67 ; Register for a single OnUpdateGameTime event, in afInterval hours of game time. All
68 ; scripts attached to this magic effect will get the update events
69 Function RegisterForSingleUpdateGameTime(float afInterval) native

```

```

61
62 ; Remove all inventory event filters from this effect - all item added/removed events
will now be received
63 Function RemoveAllInventoryEventFilters() native
64
65 ; Remove an inventory event filter from this effect. Item added/removed events matching
the
66 ; specified form (or in the specified form list) will no longer be let through.
67 Function RemoveInventoryEventFilter(Form akFilter) native
68
69 ; Turns on profiling for this specific object and all scripts attached to it - setting
doesn't persist across saves
70 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
setting is off
71 Function StartObjectProfiling() native
72
73 ; Turns off profiling for this specific object and all scripts attached to it - setting
doesn't persist across saves
74 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
setting is off
75 Function StopObjectProfiling() native
76
77 ; Unregister for any LOS events between the viewer and target
78 Function UnregisterForLOS(Actor akViewer, ObjectReference akTarget) native
79
80 ; Unregister for the specified animation event from the specified object
81 Function UnregisterForAnimationEvent(ObjectReference akSender, string asEventName) native
82
83 ; Unregisters this magic effect to receive events when the player sleeps and wakes up
84 Function UnregisterForSleep() native
85
86 ; Unregisters this magic effect from receiving events when tracked stats are updated
87 Function UnregisterForTrackedStatsEvent() native
88
89 ; Unregister for OnUpdate events, all attached scripts will stop getting update events
90 Function UnregisterForUpdate() native
91
92 ; Unregister for OnUpdateGameTime events, all attached scripts will stop getting update
game time events
93 Function UnregisterForUpdateGameTime() native
94
95 ; Animation event, sent when an object we are listening to hits one of the events we are
listening for
96 Event OnAnimationEvent(ObjectReference akSource, string asEventName)
97 EndEvent
98
99 ; Event sent when you have been unregistered from receiving an animation event because
the target
100 ; object's animation graph has been unloaded
101 Event OnAnimationEventUnregistered(ObjectReference akSource, string asEventName)
102 EndEvent
103
104 ; Event received when this effect is first started (OnInit may not have been run yet!)
105 Event OnEffectStart(Actor akTarget, Actor akCaster)
106 EndEvent
107
108 ; Event received when this effect is finished (effect may already be deleted, calling
; functions on this effect will fail)
109 Event OnEffectFinish(Actor akTarget, Actor akCaster)
110 EndEvent
111
112
113 ; LOS event, sent whenever the viewer first sees the target (after registering)
114 Event OnGainLOS(Actor akViewer, ObjectReference akTarget)
115 EndEvent
116
117 ; Lost LOS event, sent whenever the viewer first loses sight of the target (after
registering)
118 Event OnLostLOS(Actor akViewer, ObjectReference akTarget)
119 EndEvent

```

```
120
121 ; Received when the player sleeps. Start and desired end time are in game time days
    (after registering)
122 Event OnSleepStart(float afSleepStartTime, float afDesiredSleepEndTime)
123 EndEvent
124
125 ; Received when the player stops sleeping - whether naturally or interrupted (after
    registering)
126 Event OnSleepStop(bool abInterrupted)
127 EndEvent
128
129 ; Event received when a tracked stat is updated for the player
130 Event OnTrackedStatsEvent(string arStatName, int aiStatValue)
131 EndEvent
132
133 ; Update event, sent every X seconds while this magic effect is registered for them
134 Event OnUpdate()
135 EndEvent
136
137 ; Update event, sent every X hours of game time while this magic effect is registered
    for them
138 Event OnUpdateGameTime()
139 EndEvent
140
141 ; The following events are received from the actor this effect is attached to:
142
143 ; Event received when this reference is activated
144 Event OnActivate(ObjectReference akActionRef)
145 EndEvent
146
147 ; Event received when this object has moved to an attached cell from a detached one
148 Event OnAttachedToCell()
149 EndEvent
150
151 ; Event received when this object's parent cell is attached
152 Event OnCellAttach()
153 EndEvent
154
155 ; Event received when this object's parent cell is detached
156 Event OnCellDetach()
157 EndEvent
158
159 ; Event received when every object in this object's parent cell is loaded (TODO: Find
    restrictions)
160 Event OnCellLoad()
161 EndEvent
162
163 ; Event received when this object is closed
164 Event OnClose(ObjectReference akActionRef)
165 EndEvent
166
167 ; Event received when this object enters, exits, or changes containers
168 Event OnContainerChanged(ObjectReference akNewContainer, ObjectReference akOldContainer)
169 EndEvent
170
171 ; Event received when this reference's destruction stage has changed
172 Event OnDestructionStageChanged(int aiOldStage, int aiCurrentStage)
173 EndEvent
174
175 ; Event recieved when this object moves to a detached cell from an attached one
176 Event OnDetachedFromCell()
177 EndEvent
178
179 ; Event received when this object is equipped by an actor
180 Event OnEquipped(Actor akActor)
181 EndEvent
182
183 ; Event received when this object is grabbed by the player
184 Event OnGrab()
```

```
185 EndEvent
186
187 ; Event received when this object is hit by a source (weapon, spell, explosion) or
projectile attack
188 Event OnHit(ObjectReference akAggressor, Form akSource, Projectile akProjectile, bool
abPowerAttack, bool abSneakAttack, bool abBashAttack, bool abHitBlocked)
189 EndEvent
190
191 ; Event received when an item is added to this object's inventory. If the item is a
persistant reference, akItemReference will
192 ; point at it - otherwise the parameter will be None
193 Event OnItemAdded(Form akBaseItem, int aiItemCount, ObjectReference akItemReference,
ObjectReference akSourceContainer)
194 EndEvent
195
196 ; Event received when an item is removed from this object's inventory. If the item is a
persistant reference, akItemReference
197 ; will point at it - otherwise the parameter will be None
198 Event OnItemRemoved(Form akBaseItem, int aiItemCount, ObjectReference akItemReference,
ObjectReference akDestContainer)
199 EndEvent
200
201 ; Event recieved when this object is completely loaded - will be fired every time this
object is loaded
202 Event OnLoad()
203 EndEvent
204
205 ; Event received when the lock on this object changes
206 Event OnLockStateChanged()
207 EndEvent
208
209 ; Event received when a magic affect is being applied to this object
210 Event OnMagicEffectApply(ObjectReference akCaster, MagicEffect akEffect)
211 EndEvent
212
213 ; Event received when this object is opened
214 Event OnOpen(ObjectReference akActionRef)
215 EndEvent
216
217 ; Event received when this actor finishes changing its race
218 Event OnRaceSwitchComplete()
219 EndEvent
220
221 ; Event received when this object, if a book, is read
222 Event OnRead()
223 EndEvent
224
225 ; Event received when this object is released by the player
226 Event OnRelease()
227 EndEvent
228
229 ; Event received when this reference is reset
230 Event OnReset()
231 EndEvent
232
233 ; Event received when this reference is sold by an actor
234 Event OnSell(Actor akSeller)
235 EndEvent
236
237 ; Event received when a spell is cast by this object
238 Event OnSpellCast(Form akSpell)
239 EndEvent
240
241 ; Event received when translation is complete (from a call to TranslateTo)
242 Event OnTranslationComplete()
243 EndEvent
244
245 ; Event received when translation is aborted (from a call to StopTranslateTo)
246 Event OnTranslationFailed()
```

```
247 EndEvent
248
249 ; Event recieved when this reference hits a target
250 Event OnTrapHit(ObjectReference akTarget, float afXVel, float afYVel, float afZVel,
float afXPos, float afYPos, float afZPos, \
251     int aeMaterial, bool abInitialHit, int aeMotionType)
252 EndEvent
253
254 ; Event recieved when this starts hitting a target
255 Event OnTrapHitStart(ObjectReference akTarget, float afXVel, float afYVel, float afZVel,
float afXPos, float afYPos, float afZPos, \
256     int aeMaterial, bool abInitialHit, int aeMotionType)
257 EndEvent
258
259 ; Event recieved when this stops hitting a target
260 Event OnTrapHitStop(ObjectReference akTarget)
261 EndEvent
262
263 ; Event received when a this trigger is tripped
264 Event OnTrigger(ObjectReference akActionRef)
265 EndEvent
266
267 ; Event received when this trigger volume is entered
268 Event OnTriggerEnter(ObjectReference akActionRef)
269 EndEvent
270
271 ; Event received when this trigger volume is left
272 Event OnTriggerLeave(ObjectReference akActionRef)
273 EndEvent
274
275 ; Event received when this object is unequipped by an actor
276 Event OnUnequipped(Actor akActor)
277 EndEvent
278
279 ; Event recieved when this object is being unloaded - will be fired every time this
object is unloaded
280 Event OnUnload()
281 EndEvent
282
283 ; Event that is triggered when this actor's combat state against the target changes
284 ; State is as follows:
285 ; 0 - not in combat
286 ; 1 - in combat
287 ; 2 - searching
288 Event OnCombatStateChanged(Actor akTarget, int aeCombatState)
289 EndEvent
290
291 ; Event that is triggered when this actor sits in the furniture
292 Event OnSit(ObjectReference akFurniture)
293 EndEvent
294
295 ; Event that is triggered when this actor leaves the furniture
296 Event OnGetUp(ObjectReference akFurniture)
297 EndEvent
298
299 ; Event that is triggered when this actor finishes dying
300 Event OnDeath(Actor akKiller)
301 EndEvent
302
303 ; Event that is triggered when this actor begins dying
304 Event OnDying(Actor akKiller)
305 EndEvent
306
307 ; Event that is triggered when this actor changes from one location to another
308 Event OnLocationChange(Location akOldLoc, Location akNewLoc)
309 EndEvent
310
311 ; Received when the lycanthropy state of this actor changes (when
SendLycanthropyStateChanged is called)
```

```
312 Event OnLycanthropyStateChanged(bool abIsWerewolf)
313 EndEvent
314
315 ; Event received when this actor equips something - akReference may be None if object is
not persistent
316 Event OnObjectEquipped(Form akBaseObject, ObjectReference akReference)
317 EndEvent
318
319 ; Event received when this actor unequips something - akReference may be None if object
is not persistent
320 Event OnObjectUnequipped(Form akBaseObject, ObjectReference akReference)
321 EndEvent
322
323 ; Event received when this actor starts a new package
324 Event OnPackageStart(Package akNewPackage)
325 EndEvent
326
327 ; Event received when this actor's package changes
328 Event OnPackageChange(Package akOldPackage)
329 EndEvent
330
331 ; Event received when this actor's package ends
332 Event OnPackageEnd(Package akOldPackage)
333 EndEvent
334
335 ; Event received when this object's Ward is hit by a spell
336 Event OnWardHit(ObjectReference akCaster, Spell akSpell, int aiStatus)
337 EndEvent
338
339 ; Received when the player fires a bow. akWeapon will be a bow, akAmmo is the ammo or
None,
340 ; afPower will be 1.0 for a full-power shot, less for a dud, and abSunGazing will be
true if the player is looking at the sun.
341 Event OnPlayerBowShot(Weapon akWeapon, Ammo akAmmo, float afPower, bool abSunGazing)
342 EndEvent
343
344 ; Received when the player finishes fast travel, gives the duration of game time the
travel took
345 Event OnPlayerFastTravelEnd(float afTravelGameTimeHours)
346 EndEvent
347
348 ; Received immediately after the player has loaded a save game. A good time to check for
additional content.
349 Event OnPlayerLoadGame()
350 EndEvent
351
352 ; Received when StartVampireFeed is called on an actor
353 Event OnVampireFeed(Actor akTarget)
354 EndEvent
355
356 ; Received when the vampirism state of this actor changes (when
SendVampirismStateChanged is called)
357 Event OnVampirismStateChanged(bool abIsVampire)
358 EndEvent
359
360
361 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
362 ; Additional useful effect information
363 float Function GetDuration() native
364 float Function GetTimeElapsed() native
365
366 ; Registers for OnKeyDown and OnKeyUp events for the given keycode.
367 Function RegisterForKey(int keyCode) native
368 Function UnregisterForKey(int keyCode) native
369 Function UnregisterForAllKeys() native
370
371 Event OnKeyDown(int keyCode)
372 EndEvent
373
```

```

374 Event OnKeyUp(int keyCode, float holdTime)
375 EndEvent
376
377 ; Registers for OnControlDown and OnControlUp events for the given control.
378 ; For a list of valid controls, see Input.psc.
379 Function RegisterForControl(string control) native
380 Function UnregisterForControl(string control) native
381 Function UnregisterForAllControls() native
382
383 Event OnControlDown(string control)
384 EndEvent
385
386 Event OnControlUp(string control, float holdTime)
387 EndEvent
388
389 ; Registers for OnMenuOpen and OnMenuClose events for the given menu.
390 ; Registrations have to be refreshed after each game load.
391 ; For a list of valid menu names, see UI.psc.
392 Function RegisterForMenu(string menuName) native
393 Function UnregisterForMenu(string menuName) native
394 Function UnregisterForAllMenus() native
395
396 Event OnMenuOpen(string menuName)
397 endEvent
398
399 Event OnMenuClose(string menuName)
400 endEvent
401
402 ; Registers a custom event callback for given event name.
403 ; Registrations have to be refreshed after each game load.
404 ;
405 ; Examples:
406 ;     RegisterForModEvent("myCustomEvent", "MyModEventCallback")
407 ;
408 ; Event signature of custom event callbacks:
409 ;     Event MyModEventCallback(string eventName, string strArg, float numArg, Form
sender)
410 ;     endEvent
411 ;
412 Function RegisterForModEvent(string eventName, string callbackName) native
413 Function UnregisterForModEvent(string eventName) native
414 Function UnregisterForAllModEvents() native
415
416 ; Sends custom event with given generic parameters.
417 Function SendModEvent(string eventName, string strArg = "", float numArg = 0.0) native
418
419 ; See Form.psc
420 Function RegisterForCameraState() native
421 Function UnregisterForCameraState() native
422
423 Event OnPlayerCameraState(int oldState, int newState)
424 EndEvent
425
426 ; See Form.psc
427 Function RegisterForCrosshairRef() native
428 Function UnregisterForCrosshairRef() native
429
430 Event OnCrosshairRefChange(ObjectReference ref)
431 EndEvent
432
433 ; See Form.psc
434 Function RegisterForActorAction(int actionType) native
435 Function UnregisterForActorAction(int actionType) native
436
437 Event OnActorAction(int actionType, Actor akActor, Form source, int slot)
438 EndEvent
439
440 ; Registers the script for when a QueueNiNodeUpdate is called
441 Function RegisterForNiNodeUpdate() native

```

```

442 Function UnregisterForNiNodeUpdate() native
443
444 Event OnNiNodeUpdate(ObjectReference akActor)
445 EndEvent
446
447 ; returns the magnitude of the active effect
448 float Function GetMagnitude() native    :: Add a newline between files
449 Scriptname Actor extends ObjectReference Hidden
450
451 ; Relationship functions use the following values:
452 ; 4 - Lover
453 ; 3 - Ally
454 ; 2 - Confidant
455 ; 1 - Friend
456 ; 0 - Acquaintance
457 ; -1 - Rival
458 ; -2 - Foe
459 ; -3 - Enemy
460 ; -4 - Archnemesis
461
462 ; DEPRECATED - use MakePlayerFriend() instead
463 ; replacement for ModFavorPoints
464 ; if iFavorPoints > 0, will setRelationshipRank to 1 if 0
465 ; otherwise, won't do anything
466 Function ModFavorPoints(int iFavorPoints = 1)
467     if iFavorPoints > 0
468         MakePlayerFriend()
469     else
470         debug.trace(self + " ModFavorPoints called with negative param. NO EFFECT.")
471     endif
472 endFunction
473
474 ; also DEPRECATED
475 Function ModFavorPointsWithGlobal(GlobalVariable FavorPointsGlobal)
476     ModFavorPoints(FavorPointsGlobal.GetValueInt())
477 endFunction
478
479 ;this function will make an actor a friend of the player if allowed
480 Function MakePlayerFriend()
481     ActorBase myBase = GetActorBase()
482     if myBase.IsUnique()
483         if GetRelationshipRank(Game.GetPlayer())== 0
484             debug.trace(self + " MakePlayerFriend called on neutral actor - changed to
FRIEND.")
485             SetRelationshipRank(Game.GetPlayer(), 1)
486         else
487             debug.trace(self + " MakePlayerFriend called on non-neutral actor - NO
EFFECT.")
488         endif
489     else
490         debug.trace(self + " MakePlayerFriend called on non-Unique actor. NO EFFECT.")
491     endif
492 endFunction
493
494 ; Adds the specified perk to this actor
495 Function AddPerk(Perk akPerk) native
496
497 ; Adds the specified shout to this actor - returns true on success
498 bool Function AddShout(Shout akShout) native
499
500 ; Adds the specified spell to this actor - returns true on success
501 bool Function AddSpell(Spell akSpell, bool abVerbose=true) native
502
503 ; Sets this a essential actors ability to talk when in a bleedout state
504 Function AllowBleedoutDialogue(bool abCanTalk ) native
505
506 ; overrides the race flag on an actor and determines if he can talk to the player in
dialogue menu
507 Function AllowPCDialogue(bool abTalk) native

```



```

508
509 ; Attaches an "ash pile" to this actor, placing it at this actor's location and using
the specified
510 ; base object (or leveled item list) to represent the pile. If None is passed, it will
use the
511 ; default ash pile object
512 Function AttachAshPile(Form akAshPileBase = None) native
513
514 ; Can this actor fly here?
515 bool Function CanFlyHere() native
516
517 ; Clears this actor's arrested state
518 Function ClearArrested() native
519
520 ; Clears any expression override on the actor
521 Function ClearExpressionOverride() native
522
523 ; Clears this actor's extra arrows 3D
524 Function ClearExtraArrows() native
525
526 ; Remove the obligation to use a particular marker when this actor has to land.
527 Function ClearForcedLandingMarker()
528     SetForcedLandingMarker( None )
529 endFunction
530
531 ; Clear any keep offset from actor settings
532 Function ClearKeepOffsetFromActor() native
533
534 ; Clears this actor's look at target
535 Function ClearLookAt() native
536
537 ; Damages the specified actor value
538 Function DamageActorValue(string asValueName, float afDamage) native
539
540 ; Alias for DamageActorValue - damages the specified actor value
541 Function DamageAV(string asValueName, float afDamage)
542     DamageActorValue(asValueName, afDamage)
543 EndFunction
544
545 ; Initiates a dismount.
546 bool Function Dismount() native
547
548 ; Dispel all spells from this actor
549 Function DispelAllSpells() native
550
551 ; Dispel a spell from this actor
552 bool Function DispelSpell( Spell akSpell ) native
553
554 ; Apply a spell to a target in combat
555 Function DoCombatSpellApply( Spell akSpell, ObjectReference akTarget ) native
556
557 ; Enables or disable's this actor's AI
558 Function EnableAI(bool abEnable = true) native
559
560 ; End the Deferred Kill state. This must only be called if StartDeferredKill was called
first.
561 Function EndDeferredKill() native
562
563 ; Forces this actor to equip the specified item, preventing removal if requested
564 Function EquipItem(Form akItem, bool abPreventRemoval = false, bool abSilent = false)
native
565
566 ; Forces this actor to equip the specified shout
567 Function EquipShout(Shout akShout) native
568
569 ; Forces this actor to equip the specified spell. The casting source can be:
570 ; 0 - Left hand
571 ; 1 - Right hand
572 Function EquipSpell(Spell akSpell, int aiSource) native

```

```

573
574 ; Forces the AI to re-evaluate its package stack
575 Function EvaluatePackage() native
576
577 ; Force the specified actor value to a specified value
578 Function ForceActorValue(string asValueName, float afNewValue) native
579
580 ; Alias for ForceActorValue - force the specified actor value to a specified value
581 Function ForceAV(string asValueName, float afNewValue)
582     ForceActorValue(asValueName, afNewValue)
583 EndFunction
584
585 ;returns the ActorBase
586 ActorBase function GetActorBase()
587     return GetBaseObject() as ActorBase
588 endFunction
589
590 ; Gets the specified actor value - returns 0 and logs an error if the value is unknown
591 float Function GetActorValue(string asValueName) native
592
593 ; Gets the specified actor value's max, taking into account buffs/debuffs
594 float Function GetActorValueMax(string asValueName) native
595
596 ; Gets the specified actor value as a percentage of its max value - from 0 to 1
597 float Function GetActorValuePercentage(string asValueName) native
598
599 ; Alias for GetActorValue - retrieves the specified actor value
600 float Function GetAV(string asValueName)
601     return GetActorValue(asValueName)
602 EndFunction
603
604 ; Alias of GetActorValueMax - retrieves actor value's max, taking into account
605 ; buffs/debuffs
606 float Function GetAVMax(string asValueName)
607     return GetActorValueMax(asValueName)
608 EndFunction
609
610 ; Alias for GetActorValuePercentage - gets the actor value as a percent of max
611 float Function GetAVPercentage(string asValueName)
612     return GetActorValuePercentage(asValueName)
613 EndFunction
614
615 ; Gets the base value of the specified actor value - returns 0 and logs an error if the
616 ; value is unknown
617 float Function GetBaseActorValue(string asValueName) native
618
619 ; Alias for GetBaseActorValue - retrieves the specified actor value's base value
620 float Function GetBaseAV(string asValueName)
621     return GetBaseActorValue(asValueName)
622 EndFunction
623
624 ; Obtains how much it would cost to bribe this actor
625 int Function GetBribeAmount() native
626
627 ; Get the faction this actor reports crimes to
628 Faction Function GetCrimeFaction() native
629
630 ; Gets this actor's current combat state
631 int Function GetCombatState() native
632
633 ; Gets this actor's current combat target
634 Actor Function GetCombatTarget() native
635
636 ; Gets this actor's current AI package
637 Package Function GetCurrentPackage() native
638
639 ; Gets this actor's current dialogue target
640 Actor Function GetDialogueTarget() native
641

```

```

640 ; Obtain the armor currently equipped in the specified slot
641 Armor Function GetEquippedArmorInSlot(int aiSlot) native
642
643 ; Obtains the item quipped in the specified hand (0 - Left hand, 1 - Right hand)
644 ; Return values are:
645 ; -1 - Error
646 ; 0 - Nothing
647 ; 1 - One-handed sword
648 ; 2 - One-handed dagger
649 ; 3 - One-handed axe
650 ; 4 - One-handed mace
651 ; 5 - Two-handed sword
652 ; 6 - Two-handed axe
653 ; 7 - Bow
654 ; 8 - Staff
655 ; 9 - Magic spell
656 ; 10 - Shield
657 ; 11 - Torch
658 int Function GetEquippedItemType(int aiHand) native
659
660 ; Gets this actor's currently equipped shout
661 Shout Function GetEquippedShout() native
662
663 ; Gets this actor's currently equipped weapon
664 ; false - Default - Right Hand
665 ; true - Left Hand
666 Weapon Function GetEquippedWeapon(bool abLeftHand = false) native
667
668 ; Gets this actor's currently equipped shield
669 Armor Function GetEquippedShield() native
670
671 ; Gets the spell currently equipped in the specified source
672 ; 0 - Left Hand
673 ; 1 - Right Hand
674 ; 2 - Other
675 ; 3 - Instant
676 Spell Function GetEquippedSpell(int aiSource) native
677
678 ; Obtains this actor's rank with the specified faction - returns -1 if the actor is not
a member
679 int Function GetFactionRank(Faction akFaction) native
680
681 ; Obtains this actor's faction-based reaction to the other actor
682 ; 0 - Neutral
683 ; 1 - Enemy
684 ; 2 - Ally
685 ; 3 - Friend
686 int Function GetFactionReaction(Actor akOther) native
687
688 ; Obtains this actor's current flight state
689 ; 0 - Not flying
690 ; 1 - Taking off
691 ; 2 - Cruising
692 ; 3 - Hovering
693 ; 4 - Landing
694 int Function GetFlyingState() native
695
696 ; Get the ref at which this actor is obliged to land, if one is set (or none, if not).
697 ObjectReference Function GetForcedLandingMarker() native
698
699 ; Retrieves the amount of gold this actor has
700 int Function GetGoldAmount() native
701
702 ; Gets this actor's highest relationship rank - returns 0 if they have no relationships
703 int Function GetHighestRelationshipRank() native
704
705 ; Returns this actor's killer - or None if this actor is still alive
706 Actor Function GetKiller() native
707

```

```

708 ; Returns this actor's current level.
709 int Function GetLevel() native
710
711 ; Returns this actor's current light level.
712 float Function GetLightLevel() native
713
714 ; Gets this actor's highest relationship rank - returns 0 if they have no relationships
715 int Function GetLowestRelationshipRank() native
716
717 ; Obtains a leveled actor's "fake" base (the one generated by the game when the
718 ; actor is leveled. This differs from GetActorBase which will return the editor base
719 ; object)
720 ActorBase Function GetLeveledActorBase() native
721
722 ; Queries whether this actor has no bleedout recovery flag set.
723 bool Function GetNoBleedoutRecovery() native
724
725 ; Queries whether this actor receives player input
726 bool Function GetPlayerControls() native
727
728 ; Returns this actor's race
729 Race Function GetRace() native
730
731 ; Obtains the relationship rank between this actor and another
732 int Function GetRelationshipRank(Actor akOther) native
733
734 ; Obtains this actor's sit state, which is one of the following:
735 ; 0 - Not sitting
736 ; 2 - Not sitting, wants to sit
737 ; 3 - Sitting
738 ; 4 - Sitting, wants to stand
739 int Function GetSitState() native
740
741 ; Obtains this actor's sleep state, which is one of the following:
742 ; 0 - Not sleeping
743 ; 2 - Not sleeping, wants to sleep
744 ; 3 - Sleeping
745 ; 4 - Sleeping, wants to wake
746 int Function GetSleepState() native
747
748 ; Gets the voice recovery timer from the actor
749 float Function GetVoiceRecoveryTime() native
750
751 ; Gets the total "warmth rating" for this actor
752 float Function GetWarmthRating() native
753
754 ; Checks to see if this actor has the specified association with the other actor - or
755 ; anyone (if no actor is passed)
756 bool Function HasAssociation(AssociationType akAssociation, Actor akOther = None) native
757
758 ; Checks to see if this actor has a family relationship with the other actor - or anyone
759 ; (if no actor is passed)
760 bool Function HasFamilyRelationship(Actor akOther = None) native
761
762 ; Sees if this actor has line-of-sight to another object. Only the player can check LOS
763 ; to a non-actor
764 bool Function HasLOS(ObjectReference akOther) native
765
766 ; Checks to see if this actor is currently being affected by the given Magic Effect
767 bool Function HasMagicEffect(MagicEffect akEffect) native
768
769 ; Checks to see if this actor is currently being affected by a Magic Effect with the
770 ; given Keyword
771 bool Function HasMagicEffectWithKeyword(Keyword akKeyword) native
772
773 ; Checks to see if this actor has a parent relationship with the other actor
774 bool Function HasParentRelationship(Actor akOther) native
775
776 ; Checks to see if this actor has the given Perk

```

```
773 bool Function HasPerk(Perk akPerk) native
774
775 ; Checks to see if this actor has the given Spell or Shout
776 bool Function HasSpell(Form akForm) native
777
778 ; Returns if this actor is alarmed or not
779 bool Function IsAlarmed() native
780
781 ; Returns if this actor is alerted or not
782 bool Function IsAlerted() native
783
784 ; Is this actor allowed to fly?
785 bool Function IsAllowedToFly() native
786
787 ; Is this actor currently arrested?
788 bool Function IsArrested() native
789
790 ; Is this actor currently arresting his target? (Must be a guard and alarmed)
791 bool Function IsArrestingTarget() native
792
793 ; Is the actor being ridden?
794 bool Function IsBeingRidden() native
795
796 ; Is this actor currently bleeding out?
797 bool Function IsBleedingOut() native
798
799 ; Queries whether this actor has player bribe flag set.
800 bool Function IsBribed() native
801
802 ; Is this actor a child?
803 bool Function IsChild() native
804
805 ; Is this actor a commanded by another?
806 bool Function IsCommandedActor() native
807
808 ; Returns if this actor is dead or not
809 bool Function IsDead() native
810
811 ; Returns if this actor is detected by the other one
812 bool Function IsDetectedBy(Actor akOther) native
813
814 ; Is this actor doing a favor for the player?
815 bool Function IsDoingFavor() native
816
817 ; Returns if the specified object is equipped on this actor
818 bool Function IsEquipped(Form akItem) native
819
820 ; Is this actor essential?
821 bool Function IsEssential() native
822
823 ; Returns if this actor is flying or not
824 bool Function IsFlying() native
825
826 ; Returns if this actor is a guard or not
827 bool Function IsGuard() native
828
829 ; Is this actor flagged as a ghost?
830 bool Function IsGhost() native
831
832 ; Is this actor hostile to another actor?
833 bool Function IsHostileToActor(Actor akActor) native
834
835 ; Returns if this actor is currently in combat
836 bool Function IsInCombat() native
837
838 ; Checks to see if this actor is a member of the specified faction
839 bool Function IsInFaction(Faction akFaction) native
840
841 ; Returns if this actor is in a kill move or not
```

```

842 bool Function IsInKillMove() native
843
844 ; Queries whether this actor has player intimidated flag set.
845 bool Function IsIntimidated() native
846
847 ; Is the actor on a mount?
848 bool Function IsOnMount() native
849
850 ; Is the actor over-encumbered?
851 bool Function IsOverEncumbered() native
852
853 ; Checks to see if this actor the last ridden horse of the player
854 bool Function IsPlayersLastRiddenHorse() native
855
856 ; Is this actor currently a teammate of the player?
857 bool Function IsPlayerTeammate() native
858
859 ; Is this actor currently running?
860 bool Function IsRunning() native
861
862 ; Is this actor currently sneaking?
863 bool Function IsSneaking() native
864
865 ; Is this actor currently sprinting?
866 bool Function IsSprinting() native
867
868 ; Is this actor trespassing?
869 bool Function IsTrespassing() native
870
871 ; Is this actor unconscious?
872 bool Function IsUnconscious() native
873
874 ; Does this actor have his weapon and/or magic drawn?
875 bool Function IsWeaponDrawn() native
876
877 ; Sets the actor to a mode where it will keep a given offset from another actor
878 Function KeepOffsetFromActor(Actor arTarget, float afOffsetX, float afOffsetY, float
afOffsetZ, float afOffsetAngleX = 0.0, float afOffsetAngleY = 0.0, float
afOffsetAngleZ = 0.0, float afCatchUpRadius = 20.0, float afFollowRadius = 5.0) native
879
880 ; Kills this actor with the killer being the guilty party
881 Function Kill(Actor akKiller = None) native
882
883 ; Kills this actor even if essential
884 Function KillEssential(Actor akKiller = None)
885     ActorBase akActorBase = GetBaseObject() as ActorBase
886     if akActorBase.IsUnique()
887         akActorBase.SetEssential(0)
888     endif
889     Kill(akKiller)
890 endFunction
891
892 ; Kills this actor without a kill event with the killer being the guilty party
893 Function KillSilent(Actor akKiller = None) native
894
895 ; Modifies the specified actor value
896 Function ModActorValue(string asValueName, float afAmount) native
897
898 ; Alias for ModActorValue - modifies the specified actor value
899 Function ModAV(string asValueName, float afAmount)
900     ModActorValue(asValueName, afAmount)
901 EndFunction
902
903 ; Modifies this actor's rank in the faction
904 Function ModFactionRank(Faction akFaction, int aiMod) native
905
906 ; Pop this actor to the initial location for a package. Mainly for use on
907 ; disabled actors, since they would normally start at their editor locations.
908 Function MoveToPackageLocation( ) native

```

```

909
910 ; Opens this actor's inventory, as if you were pick-pocketing them. Only works on
    teammates, or anyone if forced.
911 Function OpenInventory(bool abForceOpen = false) native
912
913 ; Make the actor path to a reference, latent version
914 ; Note: this method doesn't return until the goal is reached or pathing
915 ; failed or was interrupted (by another request for instance)
916 bool Function PathToReference(ObjectReference aTarget, float afWalkRunPercent) native
917
918 ; Send an idle to the actor to load in and play.
919 bool Function PlayIdle(Idle akIdle) native
920
921 ; Send an idle to the actor to play, overriding its target with the specified reference
922 bool Function PlayIdleWithTarget(Idle akIdle, ObjectReference akTarget) native
923
924 ; Send an event to the subgraphs of an actor.
925 Function PlaySubGraphAnimation(string asEventName) native
926
927 ; Removes this actor from the specified faction
928 Function RemoveFromFaction(Faction akFaction) native
929
930 ; Removes this actor from all factions
931 Function RemoveFromAllFactions() native
932
933 ; Removes the specified perk from this actor
934 Function RemovePerk(Perk akPerk) native
935
936 ; Removes the specified shout from this actor - returns true on success
937 bool Function RemoveShout(Shout akShout) native
938
939 ; Removes the specified spell from this actor - returns true on success
940 bool Function RemoveSpell(Spell akSpell) native
941
942 ; Resets this actor's health and limb state
943 Function ResetHealthAndLimbs() native
944
945 ; Restores damage done to the actor value (up to 0 damage)
946 Function RestoreActorValue(string asValueName, float afAmount) native
947
948 ; Resurrects this actor
949 Function Resurrect() native
950
951 ; Alias for RestoreActorValue - restores damage done to the actor value
952 Function RestoreAV(string asValueName, float afAmount)
953     RestoreActorValue(asValueName, afAmount)
954 EndFunction
955
956 ; Has this actor behave as if assaulted
957 Function SendAssaultAlarm() native
958
959 ; Tell anyone who cares that the lycanthropy state of this actor has changed
960 Function SendLycanthropyStateChanged(bool abIsWerewolf) native
961
962 ; Has this actor behave as if they caught the target trespassing
963 Function SendTrespassAlarm(Actor akCriminal) native
964
965 ; Tell anyone who cares that the vampirism state of this actor has changed
966 Function SendVampirismStateChanged(bool abIsVampire) native
967
968 ; Sets the specified actor value
969 Function SetActorValue(string asValueName, float afValue) native
970
971 ; Sets the actor in an alerted state
972 Function SetAlert(bool abAlerted = true) native
973
974 ; Sets whether this actor is allowed to fly or not - if not, will land the actor
975 Function SetAllowFlying(bool abAllowed = true) native
976

```

```

977 ; Sets whether this actor is allowed to fly or not - if not, will land the actor
978 Function SetAllowFlyingEx(bool abAllowed = true, bool abAllowCrash = true, bool
abAllowSearch = false) native
979
980 ; Sets this actor's alpha - with an optional fade to that alpha
981 ; The alpha will be clamped between 0 and 1
982 Function SetAlpha(float afTargetAlpha, bool abFade = false) native
983
984 ; Sets this actor to be attacked by all other actors on sight
985 Function SetAttackActorOnSight(bool abAttackOnSight = true) native
986
987 ; Alias for SetActorValue - sets the specified actor value
988 Function SetAV(string asValueName, float afValue)
989     SetActorValue(asValueName, afValue)
990 EndFunction
991
992 ; Flags/unflags this actor as bribed by the player
993 Function SetBribed(bool abBribe = true) native
994
995 ; Sets the faction this actor reports crimes to
996 Function SetCrimeFaction(Faction akFaction) native
997
998 ; Sets this actor's critical stage, which is one of the following (properties below also
match this)
999 ; 0 - None
1000 ; 1 - Goo start
1001 ; 2 - Goo end
1002 ; 3 - Disintegrate start
1003 ; 4 - Disintegrate end
1004 Function SetCriticalStage(int aiStage) native
1005
1006 ; Flag this actor as currently doing a favor for the player
1007 Function SetDoingFavor(bool abDoingFavor = true) native
1008
1009 ; Sets this actor as "don't move" or not
1010 Function SetDontMove(bool abDontMove = true) native
1011
1012 ; Sets an expression to override any other expression other systems may give this actor.
1013 ;
1014 ; 0 - Dialogue Anger      8 - Mood Anger      15 - Combat Anger
1015 ; 1 - Dialogue Fear      9 - Mood Fear      16 - Combat Shout
1016 ; 2 - Dialogue Happy     10 - Mood Happy
1017 ; 3 - Dialogue Sad       11 - Mood Sad
1018 ; 4 - Dialogue Surprise  12 - Mood Surprise
1019 ; 5 - Dialogue Puzzled   13 - Mood Puzzled
1020 ; 6 - Dialogue Disgusted 14 - Mood Disgusted
1021 ; aiStrength is from 0 to 100 (percent)
1022 Function SetExpressionOverride(int aiMood, int aiStrength = 100) native
1023
1024 ;forces the eye texture for this actor to the give texture set
1025 Function SetEyeTexture(TextureSet akNewTexture) native
1026
1027 ; Sets this actor's rank with the specified faction
1028 Function SetFactionRank(Faction akFaction, int aiRank) native
1029
1030 ; Set a specific marker as the place at which this actor must land from flight.
1031 ; params:
1032 ; - aMarker: The ObjectReference to set as this actor's landing marker
1033 Function SetForcedLandingMarker( ObjectReference aMarker ) native
1034
1035 ; Flags/unflags this actor as a ghost
1036 Function SetGhost(bool abIsGhost = true) native
1037
1038 ; Adds this actor to a faction at rank 0 if they aren't already in it
1039 Function AddToFaction(Faction akFaction)
1040     if (!IsInFaction(akFaction))
1041         SetFactionRank(akFaction, 0)
1042     endif
1043 EndFunction

```



```

1044
1045 ; Turns on/off headtracking on this actor
1046 Function SetHeadTracking(bool abEnable = true) native
1047
1048 ; Flags/unflags this actor as intimidated by the player
1049 Function SetIntimidated(bool abIntimidate = true) native
1050
1051 ; Sets this actor's head tracking target, optionally forcing it as their pathing look-at
    target
1052 Function SetLookAt(ObjectReference akTarget, bool abPathingLookAt = false) native
1053
1054 ; Set the no bleedout recovery flag on this actor
1055 Function SetNoBleedoutRecovery(bool abAllowed) native
1056
1057 ; Sets this actor to not effect the detection level on the stealth meter if he is not
    hostile to the player
1058 Function SetNotShowOnStealthMeter(bool abNotShow) native
1059
1060 ; Sets the actors outfit and makes him wear it
1061 Function SetOutfit( Outfit akOutfit, bool abSleepOutfit = false ) native
1062
1063 ; Set/reset whether player input being sent to the actor
1064 Function SetPlayerControls(bool abControls) native
1065
1066 ; Sets the player as resisting arrest from this actor's faction
1067 Function SetPlayerResistingArrest() native
1068
1069 ; Sets or clears this actor as a teammate of the player
1070 ; abCanDoFavor - OPTIONAL default is true the teammate can do favors
1071 Function SetPlayerTeammate(bool abTeammate = true, bool abCanDoFavor=true) native
1072
1073 ; Sets the actors race
1074 ; akRace - OPTIONAL (Def=None) New race for this actor. Default, no race, to switch back
    to the original race.
1075 Function SetRace( Race akRace = None ) native
1076
1077 ; Sets the relationship rank between this actor and another (See GetRelationshipRank for
    the ranks)
1078 Function SetRelationshipRank(Actor akOther, int aiRank) native
1079
1080 ; Sets this actor as restrained or not
1081 Function SetRestrained(bool abRestrained = true) native
1082
1083 ; Set a variable on all of an actor's subgraphs
1084 Function SetSubGraphFloatVariable(string asVariableName, float afValue) native
1085
1086 ; Sets this actor as unconscious or not
1087 Function SetUnconscious(bool abUnconscious = true) native
1088
1089 ; Attach the actor to (or detach it from) a horse, cart, or other vehicle.
1090 ; akVehicle is the vehicle ref. To detach the actor from its current vehicle, set
    akVehicle to None (or to the Actor itself).
1091 Function SetVehicle( ObjectReference akVehicle ) native
1092
1093 ; Sets the voice recovery timer on the actor
1094 ; afTime is recovery time in seconds
1095 Function SetVoiceRecoveryTime( float afTime ) native
1096
1097 ; Opens the Barter menu
1098 Function ShowBarterMenu() native
1099
1100 ; Opens the Gift menu
1101 ; Params:
1102 ; - abGivingGift: True if we're giving a gift to this Actor, false if the player is
    taking a gift from this Actor
1103 ; - apFilterList: OPTIONAL (Def=None) -- If present, this form list is used to filter
    the item list. Only items
1104 ; that match keywords / items in the list will get shown
1105 ; - abShowStolenItems: OPTIONAL (Def=false) -- If true, stolen items are shown

```

```

1106 ; - abUseFavorPoints: OPTIONAL (Def=true) -- If true, favor points are added /
1107 ; subtracted with each transaction. If false, FPs aren't used at all.
1108 ; Returns: The number of favor points spent / gained while in the menu.
1109 int Function ShowGiftMenu( bool abGivingGift, FormList apFilterList = None, bool
1110 abShowStolenItems = false, bool abUseFavorPoints = true ) native
1111
1112 ; Starts Cannibal with the target
1113 Function StartCannibal(Actor akTarget) native
1114
1115 ; Starts combat with the target
1116 Function StartCombat(Actor akTarget) native
1117
1118 ; Start the Deferred Kill state. Be sure to call EndDeferredKill or the actor will be
1119 ; invulnerable.
1120 Function StartDeferredKill() native
1121
1122 ; Starts vampire feed with the target
1123 Function StartVampireFeed(Actor akTarget) native
1124
1125 ; Removes this actor from combat
1126 Function StopCombat() native
1127
1128 ; Stops all combat and alarms against this actor
1129 Function StopCombatAlarm() native
1130
1131 ; Returns whether the actor can trap the soul of the given actor.
1132 bool Function TrapSoul(Actor akTarget) native
1133
1134 ; Unequips the all items from this actor
1135 Function UnequipAll() native
1136
1137 ; Unequips the specified item from this actor
1138 Function UnequipItem(Form akItem, bool abPreventEquip = false, bool abSilent = false)
1139 ; native
1140
1141 ; Unequips the all items in this slot for the actor
1142 Function UnequipItemSlot(int aiSlot) native
1143
1144 ; Forces this actor to unequip the specified shout
1145 Function UnequipShout(Shout akShout) native
1146
1147 ; Forces this actor to unequip the specified spell. The casting source can be:
1148 ; 0 - Left hand
1149 ; 1 - Right hand
1150 Function UnequipSpell(Spell akSpell, int aiSource) native
1151
1152 ; This actor will unlock all the doors that he qualifies for ownership in his current
1153 ; parentcell
1154 Function UnLockOwnedDoorsInCell() native
1155
1156 ; Returns whether intimidate will succeed against this actor or not
1157 bool Function WillIntimidateSucceed() native
1158
1159 ; Returns whether anything the actor is wearing has the specified keyword
1160 bool Function WornHasKeyword(Keyword akKeyword) native
1161
1162 ; Makes this actor start sneaking
1163 Function StartSneaking() native
1164
1165 ; Makes this actor draw his weapon
1166 Function DrawWeapon() native
1167
1168 ; Event that is triggered when this actor's combat state against the target changes
1169 ; State is as follows:
1170 ; 0 - not in combat
1171 ; 1 - in combat
1172 ; 2 - searching
1173 Event OnCombatStateChanged(Actor akTarget, int aeCombatState)
1174 EndEvent

```

```
1170
1171 ; Event that is triggered when this actor sits in the furniture
1172 Event OnSit(ObjectReference akFurniture)
1173 EndEvent
1174
1175 ; Event that is triggered when this actor leaves the furniture
1176 Event OnGetUp(ObjectReference akFurniture)
1177 EndEvent
1178
1179 ; Event that is triggered when this actor finishes dying
1180 Event OnDeath(Actor akKiller)
1181 EndEvent
1182
1183 ; Event that is triggered when this actor begins to die
1184 Event OnDying(Actor akKiller)
1185 EndEvent
1186
1187 ; Event received when an actor enters bleedout.
1188 Event OnEnterBleedout()
1189 EndEvent
1190
1191 ; Event that is triggered when this actor changes from one location to another
1192 Event OnLocationChange(Location akOldLoc, Location akNewLoc)
1193 EndEvent
1194
1195 ; Received when the lycanthropy state of this actor changes (when
1196 SendLycanthropyStateChanged is called)
1197 Event OnLycanthropyStateChanged(bool abIsWerewolf)
1198 EndEvent
1199
1200 ; Event received when this actor equips something - akReference may be None if object is
1201 not persistent
1202 Event OnObjectEquipped(Form akBaseObject, ObjectReference akReference)
1203 EndEvent
1204
1205 ; Event received when this actor unequips something - akReference may be None if object
1206 is not persistent
1207 Event OnObjectUnequipped(Form akBaseObject, ObjectReference akReference)
1208 EndEvent
1209
1210 ; Event received when this actor starts a new package
1211 Event OnPackageStart(Package akNewPackage)
1212 EndEvent
1213
1214 ; Event received when this actor's package changes
1215 Event OnPackageChange(Package akOldPackage)
1216 EndEvent
1217
1218 ; Event received when this actor's package ends
1219 Event OnPackageEnd(Package akOldPackage)
1220 EndEvent
1221
1222 ; Event received when this actor finishes changing its race
1223 Event OnRaceSwitchComplete()
1224 EndEvent
1225
1226 ; Received when the player fires a bow. akWeapon will be a bow, akAmmo is the ammo or
1227 None,
1228 ; afPower will be 1.0 for a full-power shot, less for a dud, and abSunGazing will be
1229 true if the player is looking at the sun.
1230 Event OnPlayerBowShot(Weapon akWeapon, Ammo akAmmo, float afPower, bool abSunGazing)
1231 EndEvent
1232
1233 ; Received immediately after the player has loaded a save game. A good time to check for
1234 additional content.
1235 Event OnPlayerLoadGame()
1236 EndEvent
1237
```

```

1233 ; Received when the player finishes fast travel, gives the duration of game time the
      travel took
1234 Event OnPlayerFastTravelEnd(float afTravelGameTimeHours)
1235 EndEvent
1236
1237 ; Received when StartVampireFeed is called on an actor
1238 Event OnVampireFeed(Actor akTarget)
1239 EndEvent
1240
1241 ; Received when the vampirism state of this actor changes (when
      SendVampirismStateChanged is called)
1242 Event OnVampirismStateChanged(bool abIsVampire)
1243 EndEvent
1244
1245 ; Set of read-only properties to essentially make a fake enum for critical stages
1246 int Property CritStage_None = 0 AutoReadOnly
1247 int Property CritStage_GooStart = 1 AutoReadOnly
1248 int Property CritStage_GooEnd = 2 AutoReadOnly
1249 int Property CritStage_DisintegrateStart = 3 AutoReadOnly
1250 int Property CritStage_DisintegrateEnd = 4 AutoReadOnly
1251
1252 ; **** For Debugging Movement Animations (not in release builds) ****
1253 ; Forces the movement direction on the actor
1254 ; afXAngle, afYAngle and afZAngle are in degrees
1255 Function ForceMovementDirection(float afXAngle = 0.0, float afYAngle = 0.0, float
      afZAngle = 0.0) native
1256
1257 ; Forces the movement speed on the actor
1258 ; afSpeedMult is a speed multiplier based on the current max speeds
1259 ; - 0 -> 1 Scales between 0 and the Walk speed
1260 ; - 1 -> 2 Scales between Walk speed and Run Speed
1261 ; - 2 and above is a multiplier of the run speed (less 1.0 since Run is 2.0)
1262 Function ForceMovementSpeed(float afSpeedMult) native
1263
1264 ; Forces the movement rotation speed on the actor
1265 ; Each component of the rotation speed is a multiplier following these rules:
1266 ; - 0 -> 1 Scales between 0 and the Walk speed
1267 ; - 1 -> 2 Scales between Walk speed and Run Speed
1268 ; - 2 and above is a multiplier of the run speed (less 1.0 since Run is 2.0)
1269 Function ForceMovementRotationSpeed(float afXMult = 0.0, float afYMult = 0.0, float
      afZMult = 0.0) native
1270
1271 ; Ramps the movement direction on the actor to the passed in value over the passed in
      time
1272 ; afXAngle, afYAngle and afZAngle are in degrees
1273 ; afRampTime is in seconds
1274 Function ForceMovementDirectionRamp(float afXAngle = 0.0, float afYAngle = 0.0, float
      afZAngle = 0.0, float afRampTime = 0.1) native
1275
1276 ; Ramps the movement speed on the actor to the passed in value over the passed in time
1277 ; afSpeedMult is a speed multiplier based on the current max speeds
1278 ; - 0 -> 1 Scales between 0 and the Walk speed
1279 ; - 1 -> 2 Scales between Walk speed and Run Speed
1280 ; - 2 and above is a multiplier of the run speed (less 1.0 since Run is 2.0)
1281 ; afRampTime is in seconds
1282 Function ForceMovementSpeedRamp(float afSpeedMult, float afRampTime = 0.1) native
1283
1284 ; Ramps the movement rotation speed on the actor to the passed in value over the passed
      in time
1285 ; Each component of the rotation speed is a multiplier following these rules:
1286 ; - 0 -> 1 Scales between 0 and the Walk speed
1287 ; - 1 -> 2 Scales between Walk speed and Run Speed
1288 ; - 2 and above is a multiplier of the run speed (less 1.0 since Run is 2.0)
1289 ; afRampTime is in seconds
1290 Function ForceMovementRotationSpeedRamp(float afXMult = 0.0, float afYMult = 0.0, float
      afZMult = 0.0, float afRampTime = 0.1) native
1291
1292 ; Sets the target movement direction on the actor
1293 ; afXAngle, afYAngle and afZAngle are in degrees

```

```

1294 Function ForceTargetDirection(float afXAngle = 0.0, float afYAngle = 0.0, float afZAngle
    = 0.0) native
1295
1296 ; Sets the target movement speed on the actor
1297 ; afSpeedMult is a speed multiplier based on the current max speeds
1298 ; - 0 -> 1 Scales between 0 and the Walk speed
1299 ; - 1 -> 2 Scales between Walk speed and Run Speed
1300 ; - 2 and above is a multiplier of the run speed (less 1.0 since Run is 2.0)
1301 Function ForceTargetSpeed(float afSpeed) native
1302
1303 ; Sets the target facing angle on the actor
1304 ; afXAngle, afYAngle and afZAngle are in degrees
1305 Function ForceTargetAngle(float afXAngle = 0.0, float afYAngle = 0.0, float afZAngle =
    0.0) native
1306
1307 ; Clears any forced movement on the actor and return it to its standard state
1308 Function ClearForcedMovement() native
1309
1310
1311 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1312 ; returns the form for the item worn at the specified slotMask
1313 ; use Armor.GetMaskForSlot() to generate appropriate slotMask
1314 Form Function GetWornForm(int slotMask) native
1315
1316 ; returns the itemId for the item worn at the specified slotMask
1317 int Function GetWornItemId(int slotMask) native
1318
1319 ; returns the object currently equipped in the specified location
1320 ; 0 - left hand
1321 ; 1 - right hand
1322 ; 2 - shout
1323 Form Function GetEquippedObject(int location) native
1324
1325 ; returns the itemId of the object currently equipped in the specified hand
1326 ; 0 - left hand
1327 ; 1 - right hand
1328 int Function GetEquippedItemId(int location) native
1329
1330 ; returns the number of added spells for the actor
1331 Int Function GetSpellCount() native
1332
1333 ; returns the specified added spell for the actor
1334 Spell Function GetNthSpell(int n) native
1335
1336 ; Updates an Actors meshes (Used for Armor mesh/texture changes and face changes)
1337 ; DO NOT USE WHILE MOUNTED
1338 Function QueueNiNodeUpdate() native
1339
1340 ; Updates an Actors head mesh
1341 Function RegenerateHead() native
1342
1343 int Property EquipSlot_Default = 0 AutoReadOnly
1344 int Property EquipSlot_RightHand = 1 AutoReadOnly
1345 int Property EquipSlot_LeftHand = 2 AutoReadOnly
1346
1347 ; equips item at the given slot
1348 Function EquipItemEx(Form item, int equipSlot = 0, bool preventUnequip = false, bool
    equipSound = true) native
1349
1350 ; equips item with matching itemId at the given slot
1351 Function EquipItemById(Form item, int itemId, int equipSlot = 0, bool preventUnequip =
    false, bool equipSound = true) native
1352
1353 ; unequips item at the given slot
1354 Function UnequipItemEx(Form item, int equipSlot = 0, bool preventEquip = false) native
1355
1356 ; Adds a headpart, if the type exists it will replace, must not be misc type
1357 ; Beware: This function also affects the ActorBase
1358 Function ChangeHeadPart(HeadPart hPart) native

```

```

1359
1360 ; Replaces a headpart on the loaded mesh does not affect ActorBase
1361 ; Both old and new must exist, and be of the same type
1362 Function ReplaceHeadPart(HeadPart oPart, HeadPart newPart) native
1363
1364 ; Visually updates the actors weight
1365 ; neckDelta = (oldWeight / 100) - (newWeight / 100)
1366 ; Neck changes are player persistent, but actor per-session
1367 ; Weight itself is persistent either way so keep track of your
1368 ; original weight if you use this for Actors other than the player
1369 ; DO NOT USE WHILE MOUNTED
1370 Function UpdateWeight(float neckDelta) native
1371
1372 ; Returns whether the actors AI is enabled
1373 bool Function IsAIEnabled() native
1374
1375 ; Resets Actor AI
1376 Function ResetAI() native
1377
1378 ; Returns whether the actor is currently swimming
1379 bool Function IsSwimming() native
1380
1381 ; Sheathes the actors weapon
1382 Function SheatheWeapon() native
1383
1384 ; Returns the reference of the furniture the actor is currently using
1385 ObjectReference Function GetFurnitureReference() native
1386
1387 ; 0 - "Aah"
1388 ; 1 - "BigAah"
1389 ; 2 - "BMP"
1390 ; 3 - "ChJSh"
1391 ; 4 - "DST"
1392 ; 5 - "Eee"
1393 ; 6 - "Eh"
1394 ; 7 - "FV"
1395 ; 8 - "I"
1396 ; 9 - "K"
1397 ; 10 - "N"
1398 ; 11 - "Oh"
1399 ; 12 - "OohQ"
1400 ; 13 - "R"
1401 ; 14 - "Th"
1402 ; 15 - "W"
1403 Function SetExpressionPhoneme(int index, float value) native
1404
1405 ; 0 - "BlinkLeft"
1406 ; 1 - "BlinkRight"
1407 ; 2 - "BrowDownLeft"
1408 ; 3 - "BrowDownRight"
1409 ; 4 - "BrowInLeft"
1410 ; 5 - "BrowInRight"
1411 ; 6 - "BrowUpLeft"
1412 ; 7 - "BrowUpRight"
1413 ; 8 - "LookDown"
1414 ; 9 - "LookLeft"
1415 ; 10 - "LookRight"
1416 ; 11 - "LookUp"
1417 ; 12 - "SquintLeft"
1418 ; 13 - "SquintRight"
1419 ; 14 - "HeadPitch"
1420 ; 15 - "HeadRoll"
1421 ; 16 - "HeadYaw"
1422 Function SetExpressionModifier(int index, float value) native
1423
1424 ; Resets all expression, phoneme, and modifiers
1425 Function ResetExpressionOverrides() native
1426
1427 ; Returns all factions with the specified min and max ranks (-128 to 127)

```

```

1428 Faction[] Function GetFactions(int minRank, int maxRank) native    :: Add a newline
    between files
1429 Scriptname ActorBase extends Form Hidden
1430
1431 ; Returns this actor's class
1432 Class Function GetClass() native
1433
1434 ; Gets the number of actors of this type that have been killed
1435 int Function GetDeadCount() native
1436
1437 ; Returns this actor's gift filter formlist
1438 FormList Function GetGiftFilter() native
1439
1440 ; Returns this actor's race
1441 Race Function GetRace() native
1442
1443 ; Returns this actor's sex. Values for sex are:
1444 ; -1 - None
1445 ; 0 - Male
1446 ; 1 - Female
1447 int Function GetSex() native
1448
1449 ; Is this actor essential?
1450 bool Function IsEssential() native
1451
1452 ; Is this actor invulnerable?
1453 bool Function IsInvulnerable() native
1454
1455 ; Is this actor protected (can only be killed by player)?
1456 bool Function IsProtected() native
1457
1458 ; Is this actor base unique?
1459 bool Function IsUnique() native
1460
1461 ; Sets this actor as essential or not - if set as essential, will UNSET protected
1462 Function SetEssential(bool abEssential = true) native
1463
1464 ; Sets this actor as invulnerable or not
1465 Function SetInvulnerable(bool abInvulnerable = true) native
1466
1467 ; Sets this actor as protected or not - if set as protected, will UNSET essential
1468 Function SetProtected(bool abProtected = true) native
1469
1470 ; Sets the actors outfit
1471 Function SetOutfit( Outfit akOutfit, bool abSleepOutfit = false ) native
1472
1473
1474 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1475 ; get/set the CombatStyle of the actor
1476 CombatStyle Function GetCombatStyle() native
1477 Function SetCombatStyle(CombatStyle cs) native
1478
1479 ; Get the Outfit of the actor
1480 Outfit Function GetOutfit(bool bSleepOutfit = false) native
1481
1482 ; set the Class of the actor
1483 Function SetClass(Class c) native
1484
1485 ; Get/Set the actors body height
1486 float Function GetHeight() native
1487 Function SetHeight(float height) native
1488
1489 ; Get/Set the actors body weight
1490 float Function GetWeight() native
1491 Function SetWeight(float weight) native
1492
1493 ; Get/Set actors HeadPart by index
1494 int Function GetNumHeadParts() native
1495 HeadPart Function GetNthHeadPart(int slotPart) native

```

```

1496 Function SetNthHeadPart(HeadPart headPart,int slotPart) native
1497 int Function GetIndexOfHeadPartByType(int type) native
1498
1499 ; These functions are READ-ONLY they are for accessing the
1500 ; HeadPart list when the ActorBase's Race has been overlayed
1501 ; with another race (e.g. Vampires)
1502 int Function GetNumOverlayHeadParts() native
1503 HeadPart Function GetNthOverlayHeadPart(int slotPart) native
1504 int Function GetIndexOfOverlayHeadPartByType(int type) native
1505
1506 ; Get/Set actors face morph value by index
1507 float Function GetFaceMorph(int index) native
1508 Function SetFaceMorph(float value, int index) native
1509
1510 ; Get/Set actors facemorph preset by index
1511 ; 0 - Nose
1512 ; 1 - ??
1513 ; 2 - Mouth
1514 ; 3 - Eyes
1515 int Function GetFacePreset(int index) native
1516 Function SetFacePreset(int value, int index) native
1517
1518 ColorForm Function GetHairColor() native
1519 Function SetHairColor(ColorForm color) native
1520
1521 ; returns the number of spells defined in the base actor form
1522 int Function GetSpellCount() native
1523
1524 ; returns the specified spell defined in the base actor form
1525 Spell Function GetNthSpell(int n) native
1526
1527 ; returns the face textureset of the actor (Player Only?)
1528 TextureSet Function GetFaceTextureSet() native
1529 Function SetFaceTextureSet(TextureSet textures) native
1530
1531 ; Gets/sets the Actor's voicetype
1532 VoiceType Function GetVoiceType() native
1533 Function SetVoiceType(VoiceType nVoice) native
1534
1535 ; Gets/sets the skin of the actorbase
1536 Armor Function GetSkin() native
1537 Function SetSkin(Armor skin) native
1538
1539 ; Gets/sets the far away skin of the actorbase
1540 Armor Function GetSkinFar() native
1541 Function SetSkinFar(Armor skin) native
1542
1543 ; Gets the root template of the ActorBase
1544 ActorBase Function GetTemplate() native    :: Add a newline between files
1545 Scriptname ActorValueInfo extends Form Hidden
1546
1547 ; Returns the AVI by name
1548 ActorValueInfo Function GetActorValueInfoByName(string avName) global native
1549 ActorValueInfo Function GetAVIByName(string avName) global
1550     return GetActorValueInfoByName(avName)
1551 EndFunction
1552
1553 ; Returns the AVI by id (0-164)
1554 ActorValueInfo Function GetActorValueInfoByID(int id) global native
1555 ActorValueInfo Function GetAVIByID(int id) global
1556     return GetActorValueInfoByID(id)
1557 EndFunction
1558
1559 ; Returns whether this AVI is a skill
1560 bool Function IsSkill() native
1561
1562 ; Skill Multiplier manipulation
1563 float Function GetSkillUseMult() native
1564 Function SetSkillUseMult(float value) native

```



```

1565
1566 float Function GetSkillOffsetMult() native
1567 Function SetSkillOffsetMult(float value) native
1568
1569 float Function GetSkillImproveMult() native
1570 Function SetSkillImproveMult(float value) native
1571
1572 float Function GetSkillImproveOffset() native
1573 Function SetSkillImproveOffset(float value) native
1574
1575 ; Returns the amount of experienced gained in this skill
1576 float Function GetSkillExperience() native
1577
1578 ; Does not trigger skill-up
1579 Function SetSkillExperience(float exp) native
1580
1581 ; Adds experience to this skill (Same as console AdvanceSkill, triggers skill-up)
1582 Function AddSkillExperience(float exp) native
1583
1584 ; Returns the experience required for skill-up
1585 ; (ImproveMult * currentLevel ^ fSkillUseCurve + ImproveOffset)
1586 float Function GetExperienceForLevel(int currentLevel) native
1587
1588 ; Returns the legendary level of this skill
1589 int Function GetSkillLegendaryLevel() native
1590
1591 ; Sets the legendary level of this skill
1592 Function SetSkillLegendaryLevel(int level) native
1593
1594 ; Returns perks from the skill into the FormList
1595 ; Actor filter applies to unowned and allRanks
1596 ; unowned will add perks that the actor does not own, or only perks the actor owns
1597 ; allRanks will add all ranks of each perk to the list, unowned/owned filter also applies
1598 Function GetPerkTree(FormList list, Actor akActor = None, bool unowned = true, bool
1599 allRanks = false) native
1600
1601 ; Same as GetPerkTree except returns into a new array
1602 Perk[] Function GetPerks(Actor akActor = None, bool unowned = true, bool allRanks =
1603 false) native
1604
1605 ; Same as Actor.GetActorValue (convenience function)
1606 float Function GetCurrentValue(Actor akActor) native
1607
1608 ; Same as Actor.GetBaseActorValue (convenience function)
1609 float Function GetBaseValue(Actor akActor) native
1610
1611 ; Acquires the Maximum value for the current ActorValue
1612 float Function GetMaximumValue(Actor akActor) native :: Add a newline between files
1613 Scriptname Alias Hidden
1614
1615 ; Returns the quest that owns this alias
1616 Quest Function GetOwningQuest() native
1617
1618 ; Register for the specified animation event from the specified object - returns true if
1619 it successfully registered
1620 bool Function RegisterForAnimationEvent(ObjectReference akSender, string asEventName)
1621 native
1622
1623 ; Register for LOS gain and lost events between the viewer and the target
1624 ; A loss or gain event will be sent immediately, depending on whether or not the viewer
1625 is already looking at the target or not
1626 ; If the viewer is not the player, the target must be another actor
1627 Function RegisterForLOS(Actor akViewer, ObjectReference akTarget) native
1628
1629 ; Register for only the first LOS gain event between the viewer and the target
1630 ; If the viewer is already looking at the target, an event will be received almost
1631 immediately
1632 ; If the viewer is not the player, the target must be another actor
1633 Function RegisterForSingleLOSGain(Actor akViewer, ObjectReference akTarget) native

```

```

1628
1629 ; Register for only the first LOS lost event between the viewer and the target
1630 ; If the viewer is already not looking at the target, an event will be received almost
    immediately
1631 ; If the viewer is not the player, the target must be another actor
1632 Function RegisterForSingleLOSLost(Actor akViewer, ObjectReference akTarget) native
1633
1634 ; Register for a single OnUpdate event, in afInterval seconds. All scripts attached to
    this alias will get the update events
1635 ; Of course, this means you don't need to call UnregisterForUpdate()
1636 ; If you find yourself doing this:
1637 ; Event OnUpdate()
1638 ;     UnregisterForUpdate()
1639 ;     {Do some stuff}
1640 ; endEvent
1641 ; Then you should use RegisterForSingleUpdate instead
1642 Function RegisterForSingleUpdate(float afInterval) native
1643
1644 ; Register for OnUpdate events, every X seconds, where X is the interval. All scripts
    attached to this alias will get the update events
1645 Function RegisterForUpdate(float afInterval) native
1646
1647 ; Register for OnUpdateGameTime events, every X hours of game time, where X is the
    interval. All scripts attached to this alias will get the update events
1648 Function RegisterForUpdateGameTime(float afInterval) native
1649
1650 ; Register for a single OnUpdateGameTime event, in afInterval hours of game time. All
    scripts attached to this alias will get the update events
1651 Function RegisterForSingleUpdateGameTime(float afInterval) native
1652
1653 ; Registers this alias to receive events when the player sleeps and wakes up
1654 Function RegisterForSleep() native
1655
1656 ; Registers this alias to receive events when tracked stats are updated
1657 Function RegisterForTrackedStatsEvent() native
1658
1659 ; Turns on profiling for this specific object and all scripts attached to it - setting
    doesn't persist across saves
1660 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
    setting is off
1661 Function StartObjectProfiling() native
1662
1663 ; Turns off profiling for this specific object and all scripts attached to it - setting
    doesn't persist across saves
1664 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
    setting is off
1665 Function StopObjectProfiling() native
1666
1667 ; Unregister for any LOS events between the viewer and target
1668 Function UnregisterForLOS(Actor akViewer, ObjectReference akTarget) native
1669
1670 ; Unregister for the specified animation event from the specified object
1671 Function UnregisterForAnimationEvent(ObjectReference akSender, string asEventName) native
1672
1673 ; Unregisters this alias to receive events when the player sleeps and wakes up
1674 Function UnregisterForSleep() native
1675
1676 ; Unregisters this alias from receiving events when tracked stats are updated
1677 Function UnregisterForTrackedStatsEvent() native
1678
1679 ; Unregister for OnUpdate events, all attached scripts will stop getting update events
1680 Function UnregisterForUpdate() native
1681
1682 ; Unregister for OnUpdateGameTime events, all attached scripts will stop getting update
    game time events
1683 Function UnregisterForUpdateGameTime() native
1684
1685 ; Animation event, sent when an object we are listening to hits one of the events we are
    listening for

```

```
1686 Event OnAnimationEvent(ObjectReference akSource, string asEventName)
1687 EndEvent
1688
1689 ; Event sent when you have been unregistered from receiving an animation event because
the target
1690 ; object's animation graph has been unloaded
1691 Event OnAnimationEventUnregistered(ObjectReference akSource, string asEventName)
1692 EndEvent
1693
1694 ; LOS event, sent whenever the viewer first sees the target (after registering)
1695 Event OnGainLOS(Actor akViewer, ObjectReference akTarget)
1696 EndEvent
1697
1698 ; Lost LOS event, sent whenever the viewer first loses sight of the target (after
registering)
1699 Event OnLostLOS(Actor akViewer, ObjectReference akTarget)
1700 EndEvent
1701
1702 ; Received when the player sleeps. Start and desired end time are in game time days
(after registering)
1703 Event OnSleepStart(float afSleepStartTime, float afDesiredSleepEndTime)
1704 EndEvent
1705
1706 ; Received when the player stops sleeping - whether naturally or interrupted (after
registering)
1707 Event OnSleepStop(bool abInterrupted)
1708 EndEvent
1709
1710 ; Event received when a tracked stat is updated for the player
1711 Event OnTrackedStatsEvent(string arStatName, int aiStatValue)
1712 EndEvent
1713
1714 ; Update event, sent every X seconds while this alias is registered for them
1715 Event OnUpdate()
1716 EndEvent
1717
1718 ; Update event, sent every X hours of game time while this alias is registered for them
1719 Event OnUpdateGameTime()
1720 EndEvent
1721
1722 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1723 ; return the name of the alias
1724 string Function GetName() native
1725
1726 ; return the id of the alias
1727 int Function GetID() native
1728
1729 ; Registers for OnKeyDown and OnKeyUp events for the given keycode.
1730 Function RegisterForKey(int keyCode) native
1731 Function UnregisterForKey(int keyCode) native
1732 Function UnregisterForAllKeys() native
1733
1734 Event OnKeyDown(int keyCode)
1735 EndEvent
1736
1737 Event OnKeyUp(int keyCode, float holdTime)
1738 EndEvent
1739
1740 ; Registers for OnControlDown and OnControlUp events for the given control.
1741 ; For a list of valid controls, see Input.psc.
1742 Function RegisterForControl(string control) native
1743 Function UnregisterForControl(string control) native
1744 Function UnregisterForAllControls() native
1745
1746 Event OnControlDown(string control)
1747 EndEvent
1748
1749 Event OnControlUp(string control, float holdTime)
1750 EndEvent
```

```

1751
1752 ; Registers for OnMenuOpen and OnMenuClose events for the given menu.
1753 ; Registrations have to be refreshed after each game load.
1754 ; For a list of valid menu names, see UI.psc.
1755 Function RegisterForMenu(string menuName) native
1756 Function UnregisterForMenu(string menuName) native
1757 Function UnregisterForAllMenus() native
1758
1759 Event OnMenuOpen(string menuName)
1760 endEvent
1761
1762 Event OnMenuClose(string menuName)
1763 endEvent
1764
1765 ; Registers a custom event callback for given event name.
1766 ; Registrations have to be refreshed after each game load.
1767 ;
1768 ; Examples:
1769 ;     RegisterForModEvent("myCustomEvent", "MyModEventCallback")
1770 ;
1771 ; Event signature of custom event callbacks:
1772 ;     Event MyModEventCallback(string eventName, string strArg, float numArg, Form
1773 sender)
1774 ;     endEvent
1775
1775 Function RegisterForModEvent(string eventName, string callbackName) native
1776 Function UnregisterForModEvent(string eventName) native
1777 Function UnregisterForAllModEvents() native
1778
1779 ; Sends custom event with given generic parameters.
1780 Function SendModEvent(string eventName, string strArg = "", float numArg = 0.0) native
1781
1782 ; See Form.psc
1783 Function RegisterForCameraState() native
1784 Function UnregisterForCameraState() native
1785
1786 Event OnPlayerCameraState(int oldState, int newState)
1787 EndEvent
1788
1789 ; See Form.psc
1790 Function RegisterForCrosshairRef() native
1791 Function UnregisterForCrosshairRef() native
1792
1793 Event OnCrosshairRefChange(ObjectReference ref)
1794 EndEvent
1795
1796 ; See Form.psc
1797 Function RegisterForActorAction(int actionType) native
1798 Function UnregisterForActorAction(int actionType) native
1799
1800 Event OnActorAction(int actionType, Actor akActor, Form source, int slot)
1801 EndEvent
1802
1803 ; Registers the script for when a QueueNiNodeUpdate is called
1804 Function RegisterForNiNodeUpdate() native
1805 Function UnregisterForNiNodeUpdate() native
1806
1807 Event OnNiNodeUpdate(ObjectReference akActor)
1808 EndEvent    :: Add a newline between files
1809 Scriptname Ammo extends Form Hidden
1810
1811 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1812
1813 ; Returns whether this ammo is a bolt
1814 bool Function IsBolt() native
1815
1816 ; Returns the projectile associated with this ammo
1817 Projectile Function GetProjectile() native
1818

```

```
1819 ; Returns the base damage of this ammo
1820 float Function GetDamage() native
1821 :: Add a newline between files
1822 Scriptname Apparatus extends MiscObject Hidden
1823
1824
1825 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1826
1827 int Function GetQuality() native
1828 Function SetQuality(int quality) native :: Add a newline between files
1829 Scriptname Armor extends Form Hidden
1830
1831 ; Returns the "warmth rating" for this armor
1832 float Function GetWarmthRating() native
1833
1834 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
1835 int Function GetArmorRating() native
1836 int Function GetAR()
1837     return GetArmorRating()
1838 endFunction
1839
1840 Function SetArmorRating(int armorRating) native
1841 Function SetAR(int armorRating)
1842     return SetArmorRating(armorRating)
1843 endFunction
1844
1845 Function ModArmorRating(int modBy) native
1846 Function ModAR(int modBy)
1847     return ModArmorRating(modBy)
1848 endFunction
1849
1850 ; works on the path to the nif file representing the in-game model of the weapon
1851 string Function GetModelPath(bool bFemalePath) native
1852 Function SetModelPath(string path, bool bFemalePath) native
1853
1854 ; works on the path to the nif file representing the icon for the weapon in the inventory
1855 string Function GetIconPath(bool bFemalePath) native
1856 Function SetIconPath(string path, bool bFemalePath) native
1857
1858 ; works on the path to the file representing the message icon for the weapon
1859 string Function GetMessageIconPath(bool bFemalePath) native
1860 Function SetMessageIconPath(string path, bool bFemalePath) native
1861
1862 ; Weight Class
1863 ; 0 = Light Armor
1864 ; 1 = Heavy Armor
1865 ; 2 = None
1866 int Function GetWeightClass() native
1867 Function SetWeightClass(int weightClass) native
1868
1869 ; works on the enchantment associated with the armor
1870 Enchantment Function GetEnchantment() native
1871 Function SetEnchantment(Enchantment e) native
1872
1873 ; Armor info by keyword
1874 bool Function IsLightArmor()
1875     return HasKeywordString("ArmorLight")
1876 endFunction
1877
1878 bool Function IsHeavyArmor()
1879     return HasKeywordString("ArmorHeavy")
1880 endFunction
1881
1882 bool Function IsClothing()
1883     return HasKeywordString("ArmorClothing")
1884 endFunction
1885
1886 bool Function IsBoots()
1887     return HasKeywordString("ArmorBoots")
```

```

1888     endFunction
1889
1890     bool Function IsCuirass()
1891         return HasKeywordString("ArmorCuirass")
1892     endFunction
1893
1894     bool Function IsGauntlets()
1895         return HasKeywordString("ArmorGauntlets")
1896     endFunction
1897
1898     bool Function IsHelmet()
1899         return HasKeywordString("ArmorHelmet")
1900     endFunction
1901
1902     bool Function IsShield()
1903         return HasKeywordString("ArmorShield")
1904     endFunction
1905
1906     bool Function IsJewelry()
1907         return HasKeywordString("ArmorJewelry")
1908     endFunction
1909
1910     bool Function IsClothingHead()
1911         return HasKeywordString("ClothingHead")
1912     endFunction
1913
1914     bool Function IsClothingBody()
1915         return HasKeywordString("ClothingBody")
1916     endFunction
1917
1918     bool Function IsClothingFeet()
1919         return HasKeywordString("ClothingFeet")
1920     endFunction
1921
1922     bool Function IsClothingHands()
1923         return HasKeywordString("ClothingHands")
1924     endFunction
1925
1926     bool Function IsClothingRing()
1927         return HasKeywordString("ClothingRing")
1928     endFunction
1929
1930     bool Function IsClothingRich()
1931         return HasKeywordString("ClothingRich")
1932     endFunction
1933
1934     bool Function IsClothingPoor()
1935         return HasKeywordString("ClothingPoor")
1936     endFunction
1937
1938
1939     ; Functions and Flags dealing the BipedObject slot values from the CK
1940     ; These are the equivalent of 1 << (SlotMask-30). Basically
1941     ; these are a flags where 30 is the first bit, and 61 is the 31st bit.
1942
1943     ; returns the slot mask for the armor.
1944     int Function GetSlotMask() native
1945     ; sets the slot mask for the armor
1946     Function SetSlotMask(int slotMask) native
1947     ; adds the specified slotMask to the armor
1948     int Function AddSlotToMask(int slotMask) native
1949     ; removes the specified slot masks from the armor
1950     int Function RemoveSlotFromMask(int slotMask) native
1951
1952     ; calculates the equivalent value for the properties below
1953     int Function GetMaskForSlot(int slot) global native
1954
1955     ; returns the number of armor addons for this armor
1956     int Function GetNumArmorAddons() native

```

```

1957
1958 ; returns the nth armor addon for this armor
1959 ArmorAddon Function GetNthArmorAddon(int n) native
1960
1961 ; returns the SlotMask for a single slot from the CK
1962 ; can be used with the non-global SlotMask functions above
1963 ; and with the Math bit shifting functions
1964 int Property kSlotMask30 = 0x00000001 AutoReadOnly
1965 int Property kSlotMask31 = 0x00000002 AutoReadOnly
1966 int Property kSlotMask32 = 0x00000004 AutoReadOnly
1967 int Property kSlotMask33 = 0x00000008 AutoReadOnly
1968 int Property kSlotMask34 = 0x00000010 AutoReadOnly
1969 int Property kSlotMask35 = 0x00000020 AutoReadOnly
1970 int Property kSlotMask36 = 0x00000040 AutoReadOnly
1971 int Property kSlotMask37 = 0x00000080 AutoReadOnly
1972 int Property kSlotMask38 = 0x00000100 AutoReadOnly
1973 int Property kSlotMask39 = 0x00000200 AutoReadOnly
1974 int Property kSlotMask40 = 0x00000400 AutoReadOnly
1975 int Property kSlotMask41 = 0x00000800 AutoReadOnly
1976 int Property kSlotMask42 = 0x00001000 AutoReadOnly
1977 int Property kSlotMask43 = 0x00002000 AutoReadOnly
1978 int Property kSlotMask44 = 0x00004000 AutoReadOnly
1979 int Property kSlotMask45 = 0x00008000 AutoReadOnly
1980 int Property kSlotMask46 = 0x00010000 AutoReadOnly
1981 int Property kSlotMask47 = 0x00020000 AutoReadOnly
1982 int Property kSlotMask48 = 0x00040000 AutoReadOnly
1983 int Property kSlotMask49 = 0x00080000 AutoReadOnly
1984 int Property kSlotMask50 = 0x00100000 AutoReadOnly
1985 int Property kSlotMask51 = 0x00200000 AutoReadOnly
1986 int Property kSlotMask52 = 0x00400000 AutoReadOnly
1987 int Property kSlotMask53 = 0x00800000 AutoReadOnly
1988 int Property kSlotMask54 = 0x01000000 AutoReadOnly
1989 int Property kSlotMask55 = 0x02000000 AutoReadOnly
1990 int Property kSlotMask56 = 0x04000000 AutoReadOnly
1991 int Property kSlotMask57 = 0x08000000 AutoReadOnly
1992 int Property kSlotMask58 = 0x10000000 AutoReadOnly
1993 int Property kSlotMask59 = 0x20000000 AutoReadOnly
1994 int Property kSlotMask60 = 0x40000000 AutoReadOnly
1995 int Property kSlotMask61 = 0x80000000 AutoReadOnly :: Add a newline between files
1996 Scriptname ArmorAddon extends Form Hidden
1997
1998 ; returns the model path of the particular model
1999 string Function GetModelPath(bool firstPerson, bool female) native
2000
2001 ; sets the model path of the particular model
2002 Function SetModelPath(string path, bool firstPerson, bool female) native
2003
2004 ; returns the number of texturesets for the particular model
2005 int Function GetModelNumTextureSets(bool first, bool female) native
2006
2007 ; returns the nth textureset for the particular model
2008 TextureSet Function GetModelNthTextureSet(int n, bool first, bool female) native
2009
2010 ; sets the nth textureset for the particular model
2011 Function SetModelNthTextureSet(TextureSet texture, int n, bool first, bool female) native
2012
2013 ; returns the number of races this armor addon applies to
2014 int Function GetNumAdditionalRaces() native
2015
2016 ; returns the nth race this armor addon applies to
2017 Race Function GetNthAdditionalRace(int n) native
2018
2019 ; Functions and Flags dealing the BipedObject slot values from the CK
2020 ; These are the equivalent of 1 << (SlotMask-30). Basically
2021 ; these are a flags where 30 is the first bit, and 61 is the 31st bit.
2022
2023 ; returns the slot mask for the armor addon.
2024 int Function GetSlotMask() native
2025 ; sets the slot mask for the armor addon

```

```
2026 Function SetSlotMask(int slotMask) native
2027 ; adds the specified slotMask to the armor addon
2028 int Function AddSlotToMask(int slotMask) native
2029 ; removes the specified slot masks from the armor addon
2030 int Function RemoveSlotFromMask(int slotMask) native
2031
2032 ; calculates the equivalent mask value for the slot
2033 ; This is a global function, use it directly from Armor as it is faster
2034 int Function GetMaskForSlot(int slot) global
2035     return Armor.GetMaskForSlot(slot)
2036 EndFunction    :: Add a newline between files
2037 Scriptname Art extends Form Hidden
2038
2039 string Function GetModelPath() native
2040 Function SetModelPath(string path) native    :: Add a newline between files
2041 Scriptname Book Extends Form Hidden
2042
2043 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2044 ; Returns the spell that this book teaches
2045 Spell Function GetSpell() native
2046 Int Function GetSkill() native
2047 bool Function IsRead() native
2048 bool Function IsTakeable() native    :: Add a newline between files
2049 Scriptname Camera Hidden
2050
2051
2052 ; Returns the character's current camera state
2053 ; 0 - first person
2054 ; 1 - auto vanity
2055 ; 2 - VATS
2056 ; 3 - free
2057 ; 4 - iron sights
2058 ; 5 - furniture
2059 ; 6 - transition
2060 ; 7 - tweenmenu
2061 ; 8 - third person 1
2062 ; 9 - third person 2
2063 ; 10 - horse
2064 ; 11 - bleedout
2065 ; 12 - dragon
2066 int Function GetCameraState() global native
2067
2068 ; Updates the camera when changing Shoulder positions
2069 Function UpdateThirdPerson() global native
2070
2071 ; Returns the player's camera FOV
2072 float Function GetWorldFieldOfView() global native
2073 float Function GetWorldFOV() global
2074     return GetWorldFieldOfView()
2075 EndFunction
2076
2077 ; Sets the player's camera FOV
2078 Function SetWorldFieldOfView(float fov) global native
2079 Function SetWorldFOV(float fov) global
2080     SetWorldFieldOfView(fov)
2081 EndFunction
2082
2083 ; Returns the player's camera FOV
2084 float Function GetFirstPersonFieldOfView() global native
2085 float Function GetFirstPersonFOV() global
2086     return GetFirstPersonFieldOfView()
2087 EndFunction
2088
2089 ; Sets the player's camera FOV
2090 Function SetFirstPersonFieldOfView(float fov) global native
2091 Function SetFirstPersonFOV(float fov) global
2092     SetFirstPersonFieldOfView(fov)
2093 EndFunction    :: Add a newline between files
2094 Scriptname Cell extends Form Hidden
```



```
2095
2096 ; Gets the actor that owns this cell (or none if not owned by an actor)
2097 ActorBase Function GetActorOwner() native
2098
2099 ; Gets the faction that owns this cell (or none if not owned by a faction)
2100 Faction Function GetFactionOwner() native
2101
2102 ; Is this cell "attached"? (In the loaded area)
2103 bool Function IsAttached() native
2104
2105 ; Is this cell an interior cell?
2106 bool Function IsInterior() native
2107
2108 ; Flags the cell for reset on next load
2109 Function Reset() native
2110
2111 ; Sets this cell's owner as the specified actor
2112 Function SetActorOwner(ActorBase akActor) native
2113
2114 ; Sets this cell's owner as the specified faction
2115 Function SetFactionOwner(Faction akFaction) native
2116
2117 ; Sets the fog color for this cell (interior, non-sky-lit cells only)
2118 Function SetFogColor(int aiNearRed, int aiNearGreen, int aiNearBlue, \
2119     int aiFarRed, int aiFarGreen, int aiFarBlue) native
2120
2121 ; Adjusts this cell's fog near and far planes (interior, non-sky-lit cells only)
2122 Function SetFogPlanes(float afNear, float afFar) native
2123
2124 ; Sets the fog power for this cell (interior, non-sky-lit cells only)
2125 Function SetFogPower(float afPower) native
2126
2127 ; Sets this cell as public or private
2128 Function SetPublic(bool abPublic = true) native
2129
2130 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2131 ; Returns the number of refs in the cell
2132 int Function GetNumRefs(int formTypeFilter = 0) native
2133
2134 ; returns the ref at the specified index
2135 ObjectReference Function GetNthRef(int n, int formTypeFilter = 0) native
2136
2137 ; Returns the water level of the cell (-2147483648 if no water)
2138 float Function GetWaterLevel() native
2139
2140 ; Returns water level of the cell, if default returns water level from worldspace
2141 float Function GetActualWaterLevel() native    :: Add a newline between files
2142 Scriptname ColorComponent Hidden
2143
2144 int Function GetAlpha(int argb) global native
2145 int Function GetRed(int argb) global native
2146 int Function GetGreen(int argb) global native
2147 int Function GetBlue(int argb) global native
2148
2149 float Function GetHue(int argb) global native
2150 float Function GetSaturation(int argb) global native
2151 float Function GetValue(int argb) global native
2152
2153 int Function SetAlpha(int argb, int a) global native
2154 int Function SetRed(int argb, int r) global native
2155 int Function SetGreen(int argb, int g) global native
2156 int Function SetBlue(int argb, int b) global native
2157
2158 int Function SetHue(int argb, float h) global native
2159 int Function SetSaturation(int argb, float s) global native
2160 int Function SetValue(int argb, float v) global native    :: Add a newline between files
2161 Scriptname ColorForm extends Form Hidden
2162
2163 int Function GetColor() native
```

```
2164 Function SetColor(int color) native
2165
2166 int Function GetRed()
2167     return ColorComponent.GetRed(Self.GetColor())
2168 EndFunction
2169
2170 int Function GetGreen()
2171     return ColorComponent.GetGreen(Self.GetColor())
2172 EndFunction
2173
2174 int Function GetBlue()
2175     return ColorComponent.GetBlue(Self.GetColor())
2176 EndFunction
2177
2178 float Function GetHue()
2179     return ColorComponent.GetHue(Self.GetColor())
2180 EndFunction
2181
2182 float Function GetSaturation()
2183     return ColorComponent.GetSaturation(Self.GetColor())
2184 EndFunction
2185
2186 float Function GetValue()
2187     return ColorComponent.GetValue(Self.GetColor())
2188 EndFunction    :: Add a newline between files
2189 Scriptname CombatStyle extends Form Hidden
2190
2191 ; functions related to the General Tab values
2192 float Function GetOffensiveMult() native
2193 float Function GetDefensiveMult() native
2194 float Function GetGroupOffensiveMult() native
2195 float Function GetAvoidThreatChance() native
2196 float Function GetMeleeMult() native
2197 float Function GetRangedMult() native
2198 float Function GetMagicMult() native
2199 float Function GetShoutMult() native
2200 float Function GetStaffMult() native
2201 float Function GetUnarmedMult() native
2202
2203 Function SetOffensiveMult(float mult) native
2204 Function SetDefensiveMult(float mult) native
2205 Function SetGroupOffensiveMult(float mult) native
2206 Function SetAvoidThreatChance(float chance) native
2207 Function SetMeleeMult(float mult) native
2208 Function SetRangedMult(float mult) native
2209 Function SetMagicMult(float mult) native
2210 Function SetShoutMult(float mult) native
2211 Function SetStaffMult(float mult) native
2212 Function SetUnarmedMult(float mult) native
2213
2214 ; functions related to the Melee tab values
2215 float Function GetMeleeAttackStaggeredMult() native
2216 float Function GetMeleePowerAttackStaggeredMult() native
2217 float Function GetMeleePowerAttackBlockingMult() native
2218 float Function GetMeleeBashMult() native
2219 float Function GetMeleeBashRecoiledMult() native
2220 float Function GetMeleeBashAttackMult() native
2221 float Function GetMeleeBashPowerAttackMult() native
2222 float Function GetMeleeSpecialAttackMult() native
2223 bool Function GetAllowDualWielding() native
2224
2225 Function SetMeleeAttackStaggeredMult(float mult) native
2226 Function SetMeleePowerAttackStaggeredMult(float mult) native
2227 Function SetMeleePowerAttackBlockingMult(float mult) native
2228 Function SetMeleeBashMult(float mult) native
2229 Function SetMeleeBashRecoiledMult(float mult) native
2230 Function SetMeleeBashAttackMult(float mult) native
2231 Function SetMeleeBashPowerAttackMult(float mult) native
2232 Function SetMeleeSpecialAttackMult(float mult) native
```

```

2233 Function SetAllowDualWielding(bool allow) native
2234
2235 ; functions related to the Close Range tab values
2236 float Function GetCloseRangeDuelingCircleMult() native
2237 float Function GetCloseRangeDuelingFallbackMult() native
2238 float Function GetCloseRangeFlankingFlankDistance() native
2239 float Function GetCloseRangeFlankingStalkTime() native
2240
2241 Function SetCloseRangeDuelingCircleMult(float mult) native
2242 Function SetCloseRangeDuelingFallbackMult(float mult) native
2243 Function SetCloseRangeFlankingFlankDistance(float mult) native
2244 Function SetCloseRangeFlankingStalkTime(float mult) native
2245
2246 ; functions related to the LongRange tab values
2247 float Function GetLongRangeStrafeMult() native
2248 Function SetLongRangeStrafeMult(float mult) native
2249
2250 ; functions related to the Flight tab values
2251 float Function GetFlightHoverChance() native
2252 float Function GetFlightDiveBombChance() native
2253 float Function GetFlightFlyingAttackChance() native
2254
2255 Function SetFlightHoverChance(float chance) native
2256 Function SetFlightDiveBombChance(float chance) native
2257 Function SetFlightFlyingAttackChance(float mult) native
2258
2259
2260 :: Add a newline between files
2261 Scriptname ConstructibleObject extends MiscObject Hidden
2262
2263 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2264
2265 ; Gets/Sets the result of this recipe
2266 Form Function GetResult() native
2267 Function SetResult(Form result) native
2268
2269 ; Gets/Sets the amount of results of this recipe
2270 int Function GetResultQuantity() native
2271 Function SetResultQuantity(int quantity) native
2272
2273 ; Gets the number of ingredients
2274 int Function GetNumIngredients() native
2275
2276 ; Gets/Sets the Nth ingredient required
2277 Form Function GetNthIngredient(int n) native
2278 Function SetNthIngredient(Form required, int n) native
2279
2280 ; Gets/Sets the quantity of Nth ingredient required
2281 int Function GetNthIngredientQuantity(int n) native
2282 Function SetNthIngredientQuantity(int value, int n) native
2283
2284 ; Gets/Sets the Workbench keyword (Which apparatus creates this)
2285 Keyword Function GetWorkbenchKeyword() native
2286 Function SetWorkbenchKeyword(Keyword aKeyword) native :: Add a newline between files
2287 Scriptname DefaultObjectManager extends Form Hidden
2288
2289 ; Returns the default form for this key e.g. 'GOLD'
2290 Form Function GetForm(string key) native
2291
2292 ; Sets the default form for the particular key
2293 Function SetForm(string key, Form newForm) native
2294
2295 ; Valid Keys
2296 ; WWSP - Werewolf Spell
2297 ; SALT - Sitting Angle Limit
2298 ; APSH - Allow Player Shout
2299 ; GOLD - Gold
2300 ; LKPK - Lockpick
2301 ; SKLK - SkeletonKey

```

2302 ; PFAC - Player Faction
2303 ; GFAC - Guard Faction
2304 ; DFMS - Default Music
2305 ; BTMS - Battle Music
2306 ; DTMS - Death Music
2307 ; SCMS - Success Music
2308 ; LUMS - Level Up Music
2309 ; DCMS - Dungeon Cleared Music
2310 ; PVMA - Player Voice (Male)
2311 ; PVMC - Player Voice (Male Child)
2312 ; PVFA - Player Voice (Female)
2313 ; PVFC - Player Voice (Female Child)
2314 ; EPDF - Eat Package Default Food
2315 ; LHEQ - LeftHand Equip
2316 ; RHEQ - RightHand Equip
2317 ; EHEQ - EitherHand Equip
2318 ; VOEQ - Voice Equip
2319 ; POEQ - Potion Equip
2320 ; EACA - Every Actor Ability
2321 ; CACA - Commanded Actor Ability
2322 ; DEIS - Drug Wears Off Image Space
2323 ; DFTS - Footstep Set
2324 ; DLMT - Landscape Material
2325 ; DLZM - Dragon Land Zone Marker
2326 ; DCZM - Dragon Crash Zone Marker
2327 ; CSTY - Combat Style
2328 ; PLST - Default Pack List
2329 ; PWFD - Wait-For-Dialogue Package
2330 ; LRTB - LocRefType Boss
2331 ; VLOC - Virtual Location
2332 ; PLOC - PersistAll Location
2333 ; INVP - Inventory Player
2334 ; PTNP - Pathing Test NPC
2335 ; FPCS - Favor Cost Small
2336 ; FPCM - Favor Cost Medium
2337 ; FPCL - Favor Cost Large
2338 ; FGPD - Favor Gifts Per Day
2339 ; AASW - Action Swim State Change
2340 ; AALK - Action Look
2341 ; AALA - Action LeftAttack
2342 ; AALD - Action LeftReady
2343 ; AALR - Action LeftRelease
2344 ; AALI - Action LeftInterrupt
2345 ; AARA - Action RightAttack
2346 ; AARD - Action RightReady
2347 ; AARR - Action RightRelease
2348 ; AARI - Action RightInterrupt
2349 ; AADA - Action DualAttack
2350 ; AADL - Action DualRelease
2351 ; AAAC - Action Activate
2352 ; AAJP - Action Jump
2353 ; AAFA - Action Fall
2354 ; AALN - Action Land
2355 ; AASN - Action Sneak
2356 ; AAVC - Action Voice
2357 ; AAVD - Action VoiceReady
2358 ; AAVR - Action VoiceRelease
2359 ; AAVI - Action VoiceInterrupt
2360 ; AAID - Action Idle
2361 ; AAST - Action Sprint Start
2362 ; AASP - Action Sprint Stop
2363 ; AADR - Action Draw
2364 ; AASH - Action Sheath
2365 ; ALPA - Action Left Power Attack
2366 ; AAPA - Action Right Power Attack
2367 ; ADPA - Action Dual Power Attack
2368 ; AAS1 - Action Stagger Start
2369 ; AABH - Action Block Hit
2370 ; AABA - Action Block Anticipate

2371 ; AARC - Action Recoil
2372 ; AAR2 - Action Large Recoil
2373 ; AAB1 - Action Bleedout Start
2374 ; AAB2 - Action Bleedout Stop
2375 ; AAIS - Action Idle Stop
2376 ; AAWH - Action Ward Hit
2377 ; AAFQ - Action Force Equip
2378 ; AASC - Action Shield Change
2379 ; AAPS - Action Path Start
2380 ; AAPE - Action Path End
2381 ; AALM - Action Large Movement Delta
2382 ; AAF1 - Action Fly Start
2383 ; AAF2 - Action Fly Stop
2384 ; AAH1 - Action Hover Start
2385 ; AAH2 - Action Hover Stop
2386 ; AABI - Action Bumped Into
2387 ; AASS - Action Summoned Start
2388 ; ATKI - Action Talking Idle
2389 ; ALTI - Action Listen Idle
2390 ; AADE - Action Death
2391 ; AADW - Action Death Wait
2392 ; AIDW - Action Idle Warn
2393 ; AMST - Action Move Start
2394 ; AMSP - Action Move Stop
2395 ; ATRI - Action Turn Right
2396 ; ATLE - Action Turn Left
2397 ; ATSP - Action Turn Stop
2398 ; AMFD - Action Move Forward
2399 ; AMBK - Action Move Backward
2400 ; AMLT - Action Move Left
2401 ; AMRT - Action Move Right
2402 ; ARAG - Action Reset Animation Graph
2403 ; AKDN - Action Knockdown
2404 ; AAGU - Action Get Up
2405 ; ASID - Action Idle Stop Instant
2406 ; ARG1 - Action Ragdoll Instant
2407 ; AWWs - Action Waterwalk Start
2408 ; AREL - Action Reload
2409 ; PUSG - Pickup Sound Generic
2410 ; PDSG - Putdown Sound Generic
2411 ; PUSW - Pickup Sound Weapon
2412 ; PDSW - Putdown Sound Weapon
2413 ; PUSA - Pickup Sound Armor
2414 ; PDsA - Putdown Sound Armor
2415 ; PUSB - Pickup Sound Book
2416 ; PDSB - Putdown Sound Book
2417 ; PUSI - Pickup Sound Ingredient
2418 ; PDSI - Putdown Sound Ingredient
2419 ; HVSS - Harvest Sound
2420 ; HVFS - Harvest Failed Sound
2421 ; WBSN - Ward Break Sound
2422 ; WASN - Ward Absorb Sound
2423 ; WDSN - Ward Deflect Sound
2424 ; MFSN - Magic Fail Sound
2425 ; SFSN - Shout Fail Sound
2426 ; HFSD - Heartbeat Sound Fast
2427 ; HSSD - Heartbeat Sound Slow
2428 ; IMLH - Imagespace: Low Health
2429 ; SCSD - Soul Captured Sound
2430 ; NASD - No-Activation Sound
2431 ; MMSD - Map Menu Looping Sound
2432 ; DDSC - Dialogue Voice Category
2433 ; NDSC - Non-Dialogue Voice Category
2434 ; SFDC - SFX To Fade In Dialogue Category
2435 ; PDMC - Pause During Menu Category (Fade)
2436 ; PIMC - Pause During Menu Category (Immediate)
2437 ; PDLC - Pause During Loading Menu Category
2438 ; MDSC - Music Sound Category
2439 ; SMSC - Stats Mute Category

2440 ; SSSC - Stats Music
2441 ; MTSC - Master Sound Category
2442 ; TSSC - Time Sensitive Sound Category
2443 ; DOP2 - Dialogue Output Model (3D)
2444 ; DOP3 - Dialogue Output Model (2D)
2445 ; POPM - Player's Output Model (1st Person)
2446 ; P3OM - Player's Output Model (3rd Person)
2447 ; IOPM - Interface Output Model
2448 ; RVBT - Reverb Type
2449 ; UWLS - Underwater Loop Sound
2450 ; URVT - Underwater Reverb Type
2451 ; HRSK - Keyword - Horse
2452 ; UNDK - Keyword - Undead
2453 ; NPCK - Keyword - NPC
2454 ; KWBR - Keyword - BeastRace
2455 ; KWDM - Keyword - DummyObject
2456 ; KWGE - Keyword - UseGeometryEmitter
2457 ; KWMS - Keyword - MustStop
2458 ; KWUA - Keyword - UpdateDuringArchery
2459 ; KWOT - Keyword - Skip Outfit Items
2460 ; FTHD - Male Face Texture Set: Head
2461 ; FTMO - Male Face Texture Set: Mouth
2462 ; FTEL - Male Face Texture Set: Eyes
2463 ; FTHF - Female Face Texture Set: Head
2464 ; FTMF - Female Face Texture Set: Mouth
2465 ; FTRF - Female Face Texture Set: Eyes
2466 ; IMID - ImageSpaceModifier for inventory menu.
2467 ; PTEM - Package template
2468 ; MMCL - Main Menu Cell
2469 ; DMWL - Default MovementType: Walk
2470 ; DMRN - Default MovementType: Run
2471 ; DMSW - Default MovementType: Swim
2472 ; DMFL - Default MovementType: Fly
2473 ; DMSN - Default MovementType: Sneak
2474 ; DMSP - Default MovementType: Sprint
2475 ; SPFK - Keyword - Special Furniture
2476 ; FFFP - Keyword - Furniture Forces 1st Person
2477 ; FFTP - Keyword - Furniture Forces 3rd Person
2478 ; AFNP - Keyword - Activator Furniture No Player
2479 ; TKGS - Telekinesis Grab Sound
2480 ; TKTS - Telekinesis Throw Sound
2481 ; WMWE - World Map Weather
2482 ; HMPC - Help Manual PC
2483 ; HMXB - Help Manual XBox
2484 ; TKAM - Keyword - Type Ammo
2485 ; TKAR - Keyword - Type Armor
2486 ; TKBK - Keyword - Type Book
2487 ; TKIG - Keyword - Type Ingredient
2488 ; TKKY - Keyword - Type Key
2489 ; TKMS - Keyword - Type Misc
2490 ; TKSG - Keyword - Type SoulGem
2491 ; TKWP - Keyword - Type Weapon
2492 ; TKPT - Keyword - Type Potion
2493 ; BENW - Base Weapon Enchantment
2494 ; BENA - Base Armor Enchantment
2495 ; BAPO - Base Potion
2496 ; BAPS - Base Poison
2497 ; DRAK - Keyword - Dragon
2498 ; MVBL - Keyword - Movable
2499 ; ABSE - Art Object - Absorb Effect
2500 ; WEML - Weapon Material List
2501 ; ARTL - Armor Material List
2502 ; DIEN - Keyword - Disallow Enchanting
2503 ; FTML - Favor travel marker location
2504 ; LKHO - Keyword - Hold Location
2505 ; CWOK - Keyword - Civil War Owner
2506 ; CWNE - Keyword - Civil War Neutral
2507 ; LRSO - LocRefType - Civil War Soldier
2508 ; KWDO - Keyword - ClearableLocation

2509 ; LRRD - LocRefType - Resource Destructible
2510 ; HCLL - FormList - Hair Color List
2511 ; CMPX - Complex Scene Object
2512 ; RUSG - Keyword - Reusable SoulGem
2513 ; ANML - Keyword - Animal
2514 ; DAED - Keyword - Daedra
2515 ; BEEP - Keyword - Robot
2516 ; NRNT - Keyword - Nirnroot
2517 ; FTGF - Fighters' Guild Faction
2518 ; MGGF - Mages' Guild Faction
2519 ; TVGF - Thieves' Guild Faction
2520 ; DBHF - Dark Brotherhood Faction
2521 ; JRLF - Jarl Faction
2522 ; AWWW - Bunny Faction
2523 ; PIVV - Player Is Vampire Variable
2524 ; PIWV - Player Is Werewolf Variable
2525 ; NMRD - Road Marker
2526 ; SAT1 - Keyword: Scale Actor To 1.0
2527 ; VAMP - Keyword: Vampire
2528 ; FORG - Keyword: Forge
2529 ; COOK - Keyword: Cooking Pot
2530 ; SMLT - Keyword: Smelter
2531 ; TANN - Keyword: Tanning Rack
2532 ; HBLK - Help - Basic Lockpicking (PC)
2533 ; HBLX - Help - Basic Lockpicking (Console)
2534 ; HBFG - Help - Basic Forging
2535 ; HBCO - Help - Basic Cooking
2536 ; HBML - Help - Basic Smelting
2537 ; HBTA - Help - Basic Tanning
2538 ; HBOC - Help - Basic Object Creation
2539 ; HBEC - Help - Basic Enchanting
2540 ; HBSM - Help - Basic Smithing Weapon
2541 ; HBSA - Help - Basic Smithing Armor
2542 ; HBAL - Help - Basic Alchemy
2543 ; HBBR - Help - Barter
2544 ; HBLU - Help - Leveling up
2545 ; HBSK - Help - Skills Menu
2546 ; HBMM - Help - Map Menu
2547 ; HBJL - Help - Journal
2548 ; HBLH - Help - Low Health
2549 ; HBLM - Help - Low Magicka
2550 ; HBLS - Help - Low Stamina
2551 ; HBHJ - Help - Jail
2552 ; HBFT - Help - Teamate Favor
2553 ; HBWC - Help - Weapon Charge
2554 ; HBFS - Help - Favorites
2555 ; KHFL - Kinect Help FormList
2556 ; HBFM - Help - Flying Mount
2557 ; HBTL - Help - Target Lock
2558 ; HBAT - Help - Attack Target
2559 ; LSIS - Imagespace: Load screen
2560 ; WMDA - Keyword - Weapon Material Daedric
2561 ; WMDR - Keyword - Weapon Material Draugr
2562 ; WMDH - Keyword - Weapon Material DraugrHoned
2563 ; WMDW - Keyword - Weapon Material Dwarven
2564 ; WMEB - Keyword - Weapon Material Ebony
2565 ; WMEL - Keyword - Weapon Material Elven
2566 ; WMFA - Keyword - Weapon Material Falmer
2567 ; WMFH - Keyword - Weapon Material FalmerHoned
2568 ; WMGL - Keyword - Weapon Material Glass
2569 ; WMIM - Keyword - Weapon Material Imperial
2570 ; WMIR - Keyword - Weapon Material Iron
2571 ; WMOR - Keyword - Weapon Material Orcish
2572 ; WMST - Keyword - Weapon Material Steel
2573 ; WMWO - Keyword - Weapon Material Wood
2574 ; WTBA - Keyword - WeaponTypeBoundArrow
2575 ; AODA - Keyword - Armor Material Daedric
2576 ; AODP - Keyword - Armor Material Dragonplate
2577 ; AODS - Keyword - Armor Material Dragonscale

```

2578 ; AODB - Keyword - Armor Material Dragonbone
2579 ; AODW - Keyword - Armor Material Dwarven
2580 ; AOEB - Keyword - Armor Material Ebony
2581 ; AOEL - Keyword - Armor Material Elven
2582 ; AOES - Keyword - Armor Material ElvenSplinted
2583 ; AOFL - Keyword - Armor Material FullLeather
2584 ; AOGL - Keyword - Armor Material Glass
2585 ; AOHI - Keyword - Armor Material Hide
2586 ; AOIM - Keyword - Armor Material Imperial
2587 ; AOIH - Keyword - Armor Material ImperialHeavy
2588 ; AOIR - Keyword - Armor Material ImperialReinforced
2589 ; AOFE - Keyword - Armor Material Iron
2590 ; AOIB - Keyword - Armor Material IronBanded
2591 ; AOOR - Keyword - Armor Material Orcish
2592 ; AOSC - Keyword - Armor Material Scaled
2593 ; AOST - Keyword - Armor Material Steel
2594 ; AOSP - Keyword - Armor Material SteelPlate
2595 ; AOSK - Keyword - Armor Material Stormcloak
2596 ; AOSD - Keyword - Armor Material Studded
2597 ; GCK1 - Keyword - Generic Craftable Keyword 01
2598 ; GCK2 - Keyword - Generic Craftable Keyword 02
2599 ; GCK3 - Keyword - Generic Craftable Keyword 03
2600 ; GCK4 - Keyword - Generic Craftable Keyword 04
2601 ; GCK5 - Keyword - Generic Craftable Keyword 05
2602 ; GCK6 - Keyword - Generic Craftable Keyword 06
2603 ; GCK7 - Keyword - Generic Craftable Keyword 07
2604 ; GCK8 - Keyword - Generic Craftable Keyword 08
2605 ; GCK9 - Keyword - Generic Craftable Keyword 09
2606 ; GCKX - Keyword - Generic Craftable Keyword 10
2607 ; JWLR - Keyword - Jewelry
2608 ; KWCU - Keyword - Cuirass
2609 ; LMHP - Local Map Hide Plane
2610 ; SLDM - Snow LOD Material
2611 ; SLHD - Snow LOD Material (HD)
2612 ; ALDM - Ash LOD Material
2613 ; ALHD - Ash LOD Material (HD)
2614 ; DGFL - DialogueFollower Quest
2615 ; PTFR - PotentialFollower Faction
2616 ; AVWP - Werewolf Available Perks
2617 ; AVVP - Vampire Available Perks
2618 ; RIWR - Werewolf Race
2619 ; RIVR - Vampire Race
2620 ; RIVS - Vampire Spells
2621 ; DMXL - Dragon Mount No Land List
2622 ; PCMD - Player Can Mount Dragon Here List
2623 ; FMYS - Flying Mount - Allowed Spells
2624 ; FMNS - Flying Mount - Disallowed Spells
2625 ; MNT2 - Keyword - Mount
2626 ; AIVC - Verlet Cape
2627 ; FTNP - Furniture Test NPC
2628 ; COEX - Keyword - Conditional Explosion
2629 ; VFNC - Vampire Feed No Crime Faction
2630 ; KWSP - Skyrim - Worldspace
2631 ; ALBM - Keyword - Armor Material Light Bonemold
2632 ; ALCH - Keyword - Armor Material Light Chitin
2633 ; ALNC - Keyword - Armor Material Light Nordic
2634 ; ALSM - Keyword - Armor Material Light Stalhrim
2635 ; FMFF - Flying Mount - Fly Fast Worldspaces
2636 ; AHBM - Keyword - Armor Material Heavy Bonemold
2637 ; AHCH - Keyword - Armor Material Heavy Chitin
2638 ; AHNC - Keyword - Armor Material Heavy Nordic
2639 ; AHSM - Keyword - Armor Material Heavy Stalhrim
2640 ; WPNC - Keyword - Weapon Material Nordic
2641 ; WPSM - Keyword - Weapon Material Stalhrim :: Add a newline between files
2642 Scriptname Enchantment extends Form Hidden
2643
2644 ; Is this enchantment classified as hostile?
2645 bool Function IsHostile() native
2646

```



```
2647 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2648 ; return the number of the effects
2649 int Function GetNumEffects() native
2650
2651 ; return the magnitude of the specified effect
2652 float Function GetNthEffectMagnitude(int index) native
2653
2654 ; return the area of the specified effect
2655 int Function GetNthEffectArea(int index) native
2656
2657 ; return the duration of the specified effect
2658 int Function GetNthEffectDuration(int index) native
2659
2660 ; return the magic effect of the specified effect
2661 MagicEffect Function GetNthEffectMagicEffect(int index) native
2662
2663 ; return the index of the costliest effect
2664 int Function GetCostliestEffectIndex() native
2665
2666 ; sets the magnitude of the specified effect
2667 Function SetNthEffectMagnitude(int index, float value) native
2668
2669 ; sets the area of the specified effect
2670 Function SetNthEffectArea(int index, int value) native
2671
2672 ; sets the duration of the specified effect
2673 Function SetNthEffectDuration(int index, int value) native
2674
2675 ; returns the base enchantment of this enchantment
2676 Enchantment Function GetBaseEnchantment() native
2677
2678 ; Returns a Formlist of Keywords
2679 FormList Function GetKeywordRestrictions() native
2680
2681 ; Sets the FormList of keywords
2682 Function SetKeywordRestrictions(FormList newKeywordList) native    :: Add a newline
between files
2683 Scriptname EquipSlot extends Form Hidden
2684
2685 ; Returns the number of parent slots
2686 int Function GetNumParents() native
2687
2688 ; Returns the Nth parent slot
2689 EquipSlot Function GetNthParent(int n) native    :: Add a newline between files
2690 Scriptname Faction extends Form Hidden
2691
2692 ; Checks to see if the player can pay the crime gold for this faction
2693 bool Function CanPayCrimeGold() native
2694
2695 ; Gets the amount of gold the player is to pay to this faction for crimes
2696 int Function GetCrimeGold() native
2697
2698 ; Gets the amount of gold the player is to pay to this faction for non-violent crimes
2699 int Function GetCrimeGoldNonViolent() native
2700
2701 ; Gets the amount of gold the player is to pay to this faction for violent crimes
2702 int Function GetCrimeGoldViolent() native
2703
2704 ; Get the player's "infamy" with this faction (accumulated crime gold)
2705 int Function GetInfamy() native
2706
2707 ; Get the player's "non-violent infamy" with this faction (accumulated non-violent crime
gold)
2708 int Function GetInfamyNonViolent() native
2709
2710 ; Get the player's "violent infamy" with this faction (accumulated violent crime gold)
2711 int Function GetInfamyViolent() native
2712
2713 ; Gets this faction's reaction towards the other
```

```

2714 int Function GetReaction(Faction akOther) native
2715
2716 ; Obtains the value of all items stolen by the player from this faction that was
witnessed
2717 int Function GetStolenItemValueCrime() native
2718
2719 ; Obtains the value of all items stolen by the player from this faction that was NOT
witnessed
2720 int Function GetStolenItemValueNoCrime() native
2721
2722 ; Is the passed in faction in this faction's crime group
2723 bool Function IsFactionInCrimeGroup(Faction akOther) native
2724
2725 ; Is the player expelled from this faction?
2726 bool Function IsPlayerExpelled() native
2727
2728 ; Modifies the amount of crime gold for this faction - violent or non-violent
2729 Function ModCrimeGold(int aiAmount, bool abViolent = false) native
2730
2731 ; Modifies this faction's reaction towards the other faction
2732 Function ModReaction(Faction akOther, int aiAmount) native
2733
2734 ; Has the player pay the crime gold for this faction
2735 Function PlayerPayCrimeGold(bool abRemoveStolenItems = true, bool abGoToJail = true)
native
2736
2737 ; Finds a nearby NPC in this faction and has them behave as if assaulted
2738 Function SendAssaultAlarm() native
2739
2740 ; Sends the player to this faction's jail - removing inventory if requested, and to a
"real" jail or not
2741 Function SendPlayerToJail(bool abRemoveInventory = true, bool abRealJail = true) native
2742
2743 ; Sets this faction and the other as allies or friends - if the friend booleans are true
- the specified one-way relationship
2744 ; is a friend instead of an ally
2745 Function SetAlly(Faction akOther, bool abSelfIsFriendToOther = false, bool
abOtherIsFriendToSelf = false) native
2746
2747 ; Sets the non-violent crime gold on this faction
2748 Function SetCrimeGold(int aiGold) native
2749
2750 ; Sets the violent crime gold on this faction
2751 Function SetCrimeGoldViolent(int aiGold) native
2752
2753 ; Sets this faction and the other as enemies or neutral - if the friend booleans are
true - the specified one-way relationship
2754 ; is a neutral instead of an enemy
2755 Function SetEnemy(Faction akOther, bool abSelfIsNeutralToOther = false, bool
abOtherIsNeutralToSelf = false) native
2756
2757 ; Sets or clears the player as an enemy of this faction
2758 Function SetPlayerEnemy(bool abIsEnemy = true) native
2759
2760 ; Sets or clears the expelled flag for this faction on the player
2761 Function SetPlayerExpelled(bool abIsExpelled = true) native
2762
2763 ; Sets this faction's reaction towards the other
2764 Function SetReaction(Faction akOther, int aiNewValue) native
2765
2766 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2767
2768 int property kFaction_HiddenFromNPC = 0x00000001 AutoReadOnly
2769 int property kFaction_SpecialCombat = 0x00000002 AutoReadOnly
2770 int property kFaction_TrackCrime = 0x00000010 AutoReadOnly
2771 int property kFaction_IgnoreMurder = 0x00000020 AutoReadOnly
2772 int property kFaction_IgnoreAssault = 0x00000040 AutoReadOnly
2773 int property kFaction_IgnoreStealing = 0x00000080 AutoReadOnly
2774 int property kFaction_IgnoreTrespass = 0x00000100 AutoReadOnly

```

```

2775 int property kFaction_NoReportCrime           = 0x00000200 AutoReadOnly
2776 int property kFaction_CrimeGoldDefaults        = 0x00000400 AutoReadOnly
2777 int property kFaction_IgnorePickpocket         = 0x00000800 AutoReadOnly
2778 int property kFaction_Vendor                   = 0x00001000 AutoReadOnly
2779 int property kFaction_CanBeOwner                = 0x00002000 AutoReadOnly
2780 int property kFaction_IgnoreWerewolf           = 0x00004000 AutoReadOnly
2781
2782 ; Not recommended unless the faction was previously a vendor
2783 ; due to the faction not having a package location the vendor
2784 ; may not be able to set up shop anywhere at all
2785 Function MakeVendor()
2786     SetFactionFlag(self.kFaction_Vendor)
2787 EndFunction
2788
2789 bool Function IsVendor()
2790     return IsFactionFlagSet(self.kFaction_Vendor)
2791 EndFunction
2792
2793 Function ClearVendor()
2794     ClearFactionFlag(self.kFaction_Vendor)
2795 EndFunction
2796
2797 bool Function IsFactionFlagSet(int flag) native
2798 Function SetFactionFlag(int flag) native
2799 Function ClearFactionFlag(int flag) native
2800
2801 bool Function OnlyBuysStolenItems() native
2802 Function SetOnlyBuysStolenItems(bool onlyStolen) native
2803
2804 int Function GetVendorStartHour() native
2805 Function SetVendorStartHour(int hour) native
2806
2807 int Function GetVendorEndHour() native
2808 Function SetVendorEndHour(int hour) native
2809
2810 int Function GetVendorRadius() native
2811 Function SetVendorRadius(int radius) native
2812
2813 ObjectReference Function GetMerchantContainer() native
2814 Function SetMerchantContainer(ObjectReference akContainer) native
2815
2816 bool Function IsNotSellBuy() native
2817 Function SetNotSellBuy(bool notSellBuy) native
2818
2819 FormList Function GetBuySellList() native
2820 Function SetBuySellList(FormList akList) native    :: Add a newline between files
2821 Scriptname Flora extends Activator Hidden
2822
2823
2824 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2825 SoundDescriptor Function GetHarvestSound() native
2826 Function SetHarvestSound(SoundDescriptor akSoundDescriptor) native
2827
2828 Form Function GetIngredient() native
2829 Function SetIngredient(Form akIngredient) native    :: Add a newline between files
2830 Scriptname Form Hidden
2831
2832 ; Returns the formID for this object
2833 Int Function GetFormID() native
2834
2835 ; Obtains this form's value in gold. Will return -1 if the form doesn't have any value
2836 ; (like a quest)
2837 int Function GetGoldValue() native
2838
2839 ; Returns if this form has the specified keyword attached
2840 bool Function HasKeyword(Keyword akKeyword) native
2841
2842 ; Is the "Known" flag set for this form?
2843 bool Function PlayerKnows() native

```

```

2843
2844 ; Register for the specified animation event from the specified object - returns true if
    it successfully registered
2845 bool Function RegisterForAnimationEvent(ObjectReference akSender, string asEventName)
    native
2846
2847 ; Register for LOS gain and lost events between the viewer and the target
2848 ; A loss or gain event will be sent immediately, depending on whether or not the viewer
    is already looking at the target or not
2849 ; If the viewer is not the player, the target must be another actor
2850 Function RegisterForLOS(Actor akViewer, ObjectReference akTarget) native
2851
2852 ; Register for only the first LOS gain event between the viewer and the target
2853 ; If the viewer is already looking at the target, an event will be received almost
    immediately
2854 ; If the viewer is not the player, the target must be another actor
2855 Function RegisterForSingleLOSGain(Actor akViewer, ObjectReference akTarget) native
2856
2857 ; Register for only the first LOS lost event between the viewer and the target
2858 ; If the viewer is already not looking at the target, an event will be received almost
    immediately
2859 ; If the viewer is not the player, the target must be another actor
2860 Function RegisterForSingleLOSLost(Actor akViewer, ObjectReference akTarget) native
2861
2862 ; Register for a single OnUpdate event, in afInterval seconds. All scripts attached to
    this form will get the update events
2863 ; Of course, this means you don't need to call UnregisterForUpdate()
2864 ; If you find yourself doing this:
2865 ; Event OnUpdate()
2866 ;     UnregisterForUpdate()
2867 ;     {Do some stuff}
2868 ; endEvent
2869 ; Then you should use RegisterForSingleUpdate instead
2870 Function RegisterForSingleUpdate(float afInterval) native
2871
2872 ; Registers this form to receive events when the player sleeps and wakes up
2873 Function RegisterForSleep() native
2874
2875 ; Registers this form to receive events when tracked stats are updated
2876 Function RegisterForTrackedStatsEvent() native
2877
2878 ; Register for OnUpdate events, every X seconds, where X is the interval. All scripts
    attached to this form will get the update events
2879 Function RegisterForUpdate(float afInterval) native
2880
2881 ; Register for OnUpdateGameTime events, every X hours of game time, where X is the
    interval. All scripts attached to this form will get the update events
2882 Function RegisterForUpdateGameTime(float afInterval) native
2883
2884 ; Register for a single OnUpdateGameTime event, in afInterval hours of game time. All
    scripts attached to this form will get the update events
2885 Function RegisterForSingleUpdateGameTime(float afInterval) native
2886
2887 ; Turns on profiling for this specific object and all scripts attached to it - setting
    doesn't persist across saves
2888 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
    setting is off
2889 Function StartObjectProfiling() native
2890
2891 ; Turns off profiling for this specific object and all scripts attached to it - setting
    doesn't persist across saves
2892 ; Will do nothing on release console builds, and if the Papyrus:bEnableProfiling ini
    setting is off
2893 Function StopObjectProfiling() native
2894
2895 ; Unregister for the specified animation event from the specified object
2896 Function UnregisterForAnimationEvent(ObjectReference akSender, string asEventName) native
2897
2898 ; Unregister for any LOS events between the viewer and target

```

```
2899 Function UnregisterForLOS(Actor akViewer, ObjectReference akTarget) native
2900
2901 ; Unregisters this form to receive events when the player sleeps and wakes up
2902 Function UnregisterForSleep() native
2903
2904 ; Unregisters this form from receiving events when tracked stats are updated
2905 Function UnregisterForTrackedStatsEvent() native
2906
2907 ; Unregister for OnUpdate events, all attached scripts will stop getting update events
2908 Function UnregisterForUpdate() native
2909
2910 ; Unregister for OnUpdateGameTime events, all attached scripts will stop getting update
2911 game time events
2912 Function UnregisterForUpdateGameTime() native
2913
2914 ; Animation event, sent when an object we are listening to hits one of the events we are
2915 listening for
2916 Event OnAnimationEvent(ObjectReference akSource, string asEventName)
2917 EndEvent
2918
2919 ; Event sent when you have been unregistered from receiving an animation event because
2920 the target
2921 ; object's animation graph has been unloaded
2922 Event OnAnimationEventUnregistered(ObjectReference akSource, string asEventName)
2923 EndEvent
2924
2925 ; LOS event, sent whenever the viewer first sees the target (after registering)
2926 Event OnGainLOS(Actor akViewer, ObjectReference akTarget)
2927 EndEvent
2928
2929 ; Lost LOS event, sent whenever the viewer first loses sight of the target (after
2930 registering)
2931 Event OnLostLOS(Actor akViewer, ObjectReference akTarget)
2932 EndEvent
2933
2934 ; Received when the player sleeps. Start and desired end time are in game time days
2935 (after registering)
2936 Event OnSleepStart(float afSleepStartTime, float afDesiredSleepEndTime)
2937 EndEvent
2938
2939 ; Received when the player stops sleeping - whether naturally or interrupted (after
2940 registering)
2941 Event OnSleepStop(bool abInterrupted)
2942 EndEvent
2943
2944 ; Event received when a tracked stat is updated for the player
2945 Event OnTrackedStatsEvent(string arStatName, int aiStatValue)
2946 EndEvent
2947
2948 ; Update event, sent every X seconds while this form is registered for them
2949 Event OnUpdate()
2950 EndEvent
2951
2952 ; Update event, sent every X hours of game time while this form is registered for them
2953 Event OnUpdateGameTime()
2954 EndEvent
2955
2956 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
2957
2958 ; Returns the typecode for this form object
2959 Int Function GetType() native
2960
2961 ; returns the form's name, full name if possible
2962 string Function GetName() native
2963
2964 ; sets the name of the form
2965 Function SetName(string name) native
2966
2967 ; returns the weight of the form
```

```

2962 float Function GetWeight() native
2963
2964 ; sets the weight of the form
2965 Function SetWeight(float weight) native
2966
2967 ; sets the gold value of the form
2968 Function SetGoldValue(int value) native
2969
2970 ; returns the number of keywords on the form
2971 int Function GetNumKeywords() native
2972
2973 ; returns the keyword at the specified index
2974 Keyword Function GetNthKeyword(int index) native
2975
2976 ; returns all keywords of the form
2977 Keyword[] Function GetKeywords() native
2978
2979 bool Function HasKeywordString(string s)
2980     Keyword k = Keyword.GetKeyword(s)
2981     if k == None
2982         return false
2983     endif
2984     return HasKeyword(k)
2985 endFunction
2986
2987 ; Sets whether the player knows this form
2988 ; Should only be used for Magic Effects,
2989 ; Words of Power, and Enchantments
2990 Function SetPlayerKnows(bool knows) native
2991
2992 ; Registers for OnKeyDown and OnKeyUp events for the given keycode.
2993 Function RegisterForKey(int keyCode) native
2994 Function UnregisterForKey(int keyCode) native
2995 Function UnregisterForAllKeys() native
2996
2997 Event OnKeyDown(int keyCode)
2998 EndEvent
2999
3000 Event OnKeyUp(int keyCode, float holdTime)
3001 EndEvent
3002
3003 ; Registers for OnControlDown and OnControlUp events for the given control.
3004 ; For a list of valid controls, see Input.psc.
3005 Function RegisterForControl(string control) native
3006 Function UnregisterForControl(string control) native
3007 Function UnregisterForAllControls() native
3008
3009 Event OnControlDown(string control)
3010 EndEvent
3011
3012 Event OnControlUp(string control, float holdTime)
3013 EndEvent
3014
3015 ; Registers for OnMenuOpen and OnMenuClose events for the given menu.
3016 ; Registrations have to be refreshed after each game load.
3017 ; For a list of valid menu names, see UI.psc.
3018 Function RegisterForMenu(string menuName) native
3019 Function UnregisterForMenu(string menuName) native
3020 Function UnregisterForAllMenus() native
3021
3022 Event OnMenuOpen(string menuName)
3023 endEvent
3024
3025 Event OnMenuClose(string menuName)
3026 endEvent
3027
3028 ; Registers a custom event callback for given event name.
3029 ; Registrations have to be refreshed after each game load.
3030 ;

```

```

3031 ; Examples:
3032 ;     RegisterForModEvent("myCustomEvent", "MyModEventCallback")
3033 ;
3034 ; Event signature of custom event callbacks:
3035 ;     Event MyModEventCallback(string eventName, string strArg, float numArg, Form
sender)
3036 ;     endEvent
3037 ;
3038 Function RegisterForModEvent(string eventName, string callbackName) native
3039 Function UnregisterForModEvent(string eventName) native
3040 Function UnregisterForAllModEvents() native
3041
3042 ; Sends custom event with given generic parameters.
3043 Function SendModEvent(string eventName, string strArg = "", float numArg = 0.0) native
3044
3045 ; Registers for OnPlayerCameraState events
3046 Function RegisterForCameraState() native
3047 Function UnregisterForCameraState() native
3048
3049 Event OnPlayerCameraState(int oldState, int newState)
3050 EndEvent
3051
3052 ; Registers for OnCrosshairRefChange events
3053 Function RegisterForCrosshairRef() native
3054 Function UnregisterForCrosshairRef() native
3055
3056 ; Note: ref is none for no target
3057 Event OnCrosshairRefChange(ObjectReference ref)
3058 EndEvent
3059
3060 Function RegisterForActorAction(int actionType) native
3061 Function UnregisterForActorAction(int actionType) native
3062
3063 ; ActionTypes
3064 ; 0 - Weapon Swing (Melee weapons that are swung, also barehand)
3065 ; 1 - Spell Cast (Spells and staves)
3066 ; 2 - Spell Fire (Spells and staves)
3067 ; 3 - Voice Cast
3068 ; 4 - Voice Fire
3069 ; 5 - Bow Draw
3070 ; 6 - Bow Release
3071 ; 7 - Unsheathe Begin
3072 ; 8 - Unsheathe End
3073 ; 9 - Sheathe Begin
3074 ; 10 - Sheathe End
3075 ; Slots
3076 ; 0 - Left Hand
3077 ; 1 - Right Hand
3078 ; 2 - Voice
3079 Event OnActorAction(int actionType, Actor akActor, Form source, int slot)
3080 EndEvent
3081
3082 ; Registers the script for when a QueueNiNodeUpdate is called
3083 Function RegisterForNiNodeUpdate() native
3084 Function UnregisterForNiNodeUpdate() native
3085
3086 Event OnNiNodeUpdate(ObjectReference akActor)
3087 EndEvent
3088
3089 ; Returns a temporary clone of this form
3090 Form Function TempClone() native
3091
3092 ; Returns whether this Form has a World Model (fast)
3093 bool Function HasWorldModel() native
3094
3095 ; Returns the world model path of this Form, if it has a world model
3096 string Function GetWorldModelPath() native
3097 Function SetWorldModelPath(string path) native
3098

```

```

3099 ; Returns the number of texture sets the world model has, if its textures can be swapped
3100 int Function GetWorldModelNumTextureSets() native
3101
3102 ; Returns the Nth texture set of the world model, if the textures can be swapped
3103 TextureSet Function GetWorldModelNthTextureSet(int n) native
3104
3105 ; Sets the world models Nth texture set, if the textures can be set
3106 Function SetWorldModelNthTextureSet(TextureSet nSet, int n) native
3107
3108 ; Returns whether this Form is playable, only applied to Forms with the playable flag
3109 bool Function IsPlayable() native      :: Add a newline between files
3110 Scriptname FormList extends Form
3111
3112 ; Adds the given form to this form list
3113 Function AddForm(Form apForm) native
3114
3115 ; Finds the specified form in the form list and returns its index.
3116 ; If not found, returns a negative number
3117 int Function Find(Form apForm) native
3118
3119 ; Returns the number of forms in the list
3120 int Function GetSize() native
3121
3122 ; Returns the form at index 'aiIndex' in the list
3123 Form Function GetAt(int aiIndex) native
3124
3125 ; Queries the form list to see if it contains the passed in form
3126 bool Function HasForm(Form akForm) native
3127
3128 ; Removes the given added form from this form list
3129 Function RemoveAddedForm(Form apForm) native
3130
3131 ; Removes all script added forms from this form list
3132 Function Revert() native
3133
3134
3135 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
3136 ; Returns a Form array of this list (Invalid entries will be None)
3137 Form[] Function ToArray() native
3138
3139 ; Adds an Array of Forms to this list
3140 Function AddForms(Form[] forms) native      :: Add a newline between files
3141 Scriptname FormType Hidden
3142
3143 int Property kNone = 0 AutoReadOnly
3144 int Property kTES4 = 1 AutoReadOnly
3145 int Property kGroup = 2 AutoReadOnly
3146 int Property kGMST = 3 AutoReadOnly
3147 int Property kKeyword = 4 AutoReadOnly
3148 int Property kLocationRef = 5 AutoReadOnly
3149 int Property kAction = 6 AutoReadOnly
3150 int Property kTextureSet = 7 AutoReadOnly
3151 int Property kMenuItem = 8 AutoReadOnly
3152 int Property kGlobal = 9 AutoReadOnly
3153 int Property kClass = 10 AutoReadOnly
3154 int Property kFaction = 11 AutoReadOnly
3155 int Property kHeadPart = 12 AutoReadOnly
3156 int Property kEyes = 13 AutoReadOnly
3157 int Property kRace = 14 AutoReadOnly
3158 int Property kSound = 15 AutoReadOnly
3159 int Property kAcousticSpace = 16 AutoReadOnly
3160 int Property kSkill = 17 AutoReadOnly
3161 int Property kEffectSetting = 18 AutoReadOnly
3162 int Property kScript = 19 AutoReadOnly
3163 int Property kLandTexture = 20 AutoReadOnly
3164 int Property kEnchantment = 21 AutoReadOnly
3165 int Property kSpell = 22 AutoReadOnly
3166 int Property kScrollItem = 23 AutoReadOnly
3167 int Property kActivator = 24 AutoReadOnly

```



```
3168 int Property kTalkingActivator = 25 AutoReadOnly
3169 int Property kArmor = 26 AutoReadOnly
3170 int Property kBook = 27 AutoReadOnly
3171 int Property kContainer = 28 AutoReadOnly
3172 int Property kDoor = 29 AutoReadOnly
3173 int Property kIngredient = 30 AutoReadOnly
3174 int Property kLight = 31 AutoReadOnly
3175 int Property kMisc = 32 AutoReadOnly
3176 int Property kApparatus = 33 AutoReadOnly
3177 int Property kStatic = 34 AutoReadOnly
3178 int Property kStaticCollection = 35 AutoReadOnly
3179 int Property kMovableStatic = 36 AutoReadOnly
3180 int Property kGrass = 37 AutoReadOnly
3181 int Property kTree = 38 AutoReadOnly
3182 int Property kFlora = 39 AutoReadOnly
3183 int Property kFurniture = 40 AutoReadOnly
3184 int Property kWeapon = 41 AutoReadOnly
3185 int Property kAmmo = 42 AutoReadOnly
3186 int Property kNPC = 43 AutoReadOnly
3187 int Property kLeveledCharacter = 44 AutoReadOnly
3188 int Property kKey = 45 AutoReadOnly
3189 int Property kPotion = 46 AutoReadOnly
3190 int Property kIdleMarker = 47 AutoReadOnly
3191 int Property kNote = 48 AutoReadOnly
3192 int Property kConstructibleObject = 49 AutoReadOnly
3193 int Property kProjectile = 50 AutoReadOnly
3194 int Property kHazard = 51 AutoReadOnly
3195 int Property kSoulGem = 52 AutoReadOnly
3196 int Property kLeveledItem = 53 AutoReadOnly
3197 int Property kWeather = 54 AutoReadOnly
3198 int Property kClimate = 55 AutoReadOnly
3199 int Property kShaderParticleGeometryData = 56 AutoReadOnly
3200 int Property kReferenceEffect = 57 AutoReadOnly
3201 int Property kRegion = 58 AutoReadOnly
3202 int Property kNAVI = 59 AutoReadOnly
3203 int Property kCell = 60 AutoReadOnly
3204 int Property kReference = 61 AutoReadOnly
3205 int Property kCharacter = 62 AutoReadOnly
3206 int Property kMissile = 63 AutoReadOnly
3207 int Property kArrow = 64 AutoReadOnly
3208 int Property kGrenade = 65 AutoReadOnly
3209 int Property kBeamProjectile = 66 AutoReadOnly
3210 int Property kFlameProjectile = 67 AutoReadOnly
3211 int Property kConeProjectile = 68 AutoReadOnly
3212 int Property kBarrierProjectile = 69 AutoReadOnly
3213 int Property kPHZD = 70 AutoReadOnly
3214 int Property kWorldSpace = 71 AutoReadOnly
3215 int Property kLand = 72 AutoReadOnly
3216 int Property kNavMesh = 73 AutoReadOnly
3217 int Property kTLOD = 74 AutoReadOnly
3218 int Property kTopic = 75 AutoReadOnly
3219 int Property kTopicInfo = 76 AutoReadOnly
3220 int Property kQuest = 77 AutoReadOnly
3221 int Property kIdle = 78 AutoReadOnly
3222 int Property kPackage = 79 AutoReadOnly
3223 int Property kCombatStyle = 80 AutoReadOnly
3224 int Property kLoadScreen = 81 AutoReadOnly
3225 int Property kLeveledSpell = 82 AutoReadOnly
3226 int Property kANIO = 83 AutoReadOnly
3227 int Property kWater = 84 AutoReadOnly
3228 int Property kEffectShader = 85 AutoReadOnly
3229 int Property kTOFT = 86 AutoReadOnly
3230 int Property kExplosion = 87 AutoReadOnly
3231 int Property kDebris = 88 AutoReadOnly
3232 int Property kImageSpace = 89 AutoReadOnly
3233 int Property kImageSpaceModifier = 90 AutoReadOnly
3234 int Property kList = 91 AutoReadOnly
3235 int Property kPerk = 92 AutoReadOnly
3236 int Property kBodyPartData = 93 AutoReadOnly
```

```

3237 int Property kAddonNode = 94 AutoReadOnly
3238 int Property kActorValueInfo = 95 AutoReadOnly
3239 int Property kCameraShot = 96 AutoReadOnly
3240 int Property kCameraPath = 97 AutoReadOnly
3241 int Property kVoiceType = 98 AutoReadOnly
3242 int Property kMaterialType = 99 AutoReadOnly
3243 int Property kImpactData = 100 AutoReadOnly
3244 int Property kImpactDataSet = 101 AutoReadOnly
3245 int Property kARMA = 102 AutoReadOnly
3246 int Property kEncounterZone = 103 AutoReadOnly
3247 int Property kLocation = 104 AutoReadOnly
3248 int Property kMessage = 105 AutoReadOnly
3249 int Property kRagdoll = 106 AutoReadOnly
3250 int Property kDefaultObject = 107 AutoReadOnly
3251 int Property kLightingTemplate = 108 AutoReadOnly
3252 int Property kMusicType = 109 AutoReadOnly
3253 int Property kFootstep = 110 AutoReadOnly
3254 int Property kFootstepSet = 111 AutoReadOnly
3255 int Property kStoryBranchNode = 112 AutoReadOnly
3256 int Property kStoryQuestNode = 113 AutoReadOnly
3257 int Property kStoryEventNode = 114 AutoReadOnly
3258 int Property kDialogueBranch = 115 AutoReadOnly
3259 int Property kMusicTrack = 116 AutoReadOnly
3260 int Property kDLVW = 117 AutoReadOnly
3261 int Property kWordOfPower = 118 AutoReadOnly
3262 int Property kShout = 119 AutoReadOnly
3263 int Property kEquipSlot = 120 AutoReadOnly
3264 int Property kRelationship = 121 AutoReadOnly
3265 int Property kScene = 122 AutoReadOnly
3266 int Property kAssociationType = 123 AutoReadOnly
3267 int Property kOutfit = 124 AutoReadOnly
3268 int Property kArt = 125 AutoReadOnly
3269 int Property kMaterial = 126 AutoReadOnly
3270 int Property kMovementType = 127 AutoReadOnly
3271 int Property kSoundDescriptor = 128 AutoReadOnly
3272 int Property kDualCastData = 129 AutoReadOnly
3273 int Property kSoundCategory = 130 AutoReadOnly
3274 int Property kSoundOutput = 131 AutoReadOnly
3275 int Property kCollisionLayer = 132 AutoReadOnly
3276 int Property kColorForm = 133 AutoReadOnly
3277 int Property kReverbParam = 134 AutoReadOnly
3278 :: Add a newline between files
3279 Scriptname Game Hidden
3280
3281 ; Adds the specified achievement to the player's profile
3282 Function AddAchievement(int aiAchievementID) native global
3283
3284 ; Add the specified number of perk points to the player
3285 Function AddPerkPoints(int aiPerkPoints) native global
3286
3287 ; Advance the given skill on the player by the provided amount of skill usage
3288 Function AdvanceSkill(string asSkillName, float afMagnitude) native global
3289
3290 ; Adds a ball-and-socket constraint between two rigid bodies, identified by their ref
and node names
3291 bool Function AddHavokBallAndSocketConstraint( ObjectReference arRefA, string
arRefANode, ObjectReference arRefB, string arRefBNode, float afRefALocalOffsetX = 0.0,
float afRefALocalOffsetY = 0.0, float afRefALocalOffsetZ = 0.0, float afRefBLocalOffsetX
= 0.0, float afRefBLocalOffsetY = 0.0, float afRefBLocalOffsetZ = 0.0) native global
3292
3293 ; Removes any constraint between two rigid bodies
3294 bool Function RemoveHavokConstraints(ObjectReference arFirstRef, string
arFirstRefNodeName, ObjectReference arSecondRef, string arSecondRefNodeName) native
global
3295
3296 ; Calculates how much a x point favor would cost the player
3297 int Function CalculateFavorCost(int aiFavorPrice) native global
3298
3299 ; Clears the prison variables on the player

```

```

3300 Function ClearPrison() native global
3301
3302 ; Clears temp effects from game
3303 Function ClearTempEffects() native global
3304
3305 ; Disables the user's controls
3306 Function DisablePlayerControls(bool abMovement = true, bool abFighting = true, bool
abCamSwitch = false, bool abLooking = false, \
3307     bool abSneaking = false, bool abMenu = true, bool abActivate = true, bool
abJournalTabs = false, int aiDisablePOVType = 0) native global
3308
3309 ; Enables the user's controls
3310 Function EnablePlayerControls(bool abMovement = true, bool abFighting = true, bool
abCamSwitch = true, bool abLooking = true, \
3311     bool abSneaking = true, bool abMenu = true, bool abActivate = true, bool abJournalTabs
= true, int aiDisablePOVType = 0) native global
3312
3313 ; Enables or disables the ability to fast travel
3314 Function EnableFastTravel(bool abEnable = true) native global
3315
3316 ; Fades out the game to black, or vice versa
3317 Function FadeOutGame(bool abFadingOut, bool abBlackFade, float afSecsBeforeFade, float
afFadeDuration) native global
3318
3319 ; Fast-travels the player to the specified object's location
3320 Function FastTravel(ObjectReference akDestination) native global
3321
3322 ; Finds the closest reference of a given base object within a given radius of a location
3323 ObjectReference Function FindClosestReferenceOfType(Form arBaseObject, float afX, float
afY, float afZ, float afRadius) native global
3324
3325 ; Finds a random reference of a given base object within a given radius of a location
3326 ObjectReference Function FindRandomReferenceOfType(Form arBaseObject, float afX, float
afY, float afZ, float afRadius) native global
3327
3328 ; Finds the closest reference of any base object in the list within a given radius of a
location
3329 ObjectReference Function FindClosestReferenceOfAnyTypeInList(FormList arBaseObjects,
float afX, float afY, float afZ, float afRadius) native global
3330
3331 ; Finds a random reference of a any base object in the list within a given radius of a
location
3332 ObjectReference Function FindRandomReferenceOfAnyTypeInList(FormList arBaseObjects,
float afX, float afY, float afZ, float afRadius) native global
3333
3334 ; Finds the closest reference of a given base object within a given radius of a reference
3335 ObjectReference Function FindClosestReferenceOfTypeFromRef(Form arBaseObject,
ObjectReference arCenter, float afRadius) global
3336     return FindClosestReferenceOfType(arBaseObject, arCenter.X, arCenter.Y, arCenter.Z,
afRadius)
3337 endFunction
3338
3339 ; Finds a random reference of a given base object within a given radius of a reference
3340 ObjectReference Function FindRandomReferenceOfTypeFromRef(Form arBaseObject,
ObjectReference arCenter, float afRadius) global
3341     return FindRandomReferenceOfType(arBaseObject, arCenter.X, arCenter.Y, arCenter.Z,
afRadius)
3342 endFunction
3343
3344 ; Finds the closest reference of a given base object within a given radius of a reference
3345 ObjectReference Function FindClosestReferenceOfAnyTypeInListFromRef(FormList
arBaseObjects, ObjectReference arCenter, float afRadius) global
3346     return FindClosestReferenceOfAnyTypeInList(arBaseObjects, arCenter.X, arCenter.Y,
arCenter.Z, afRadius)
3347 endFunction
3348
3349 ; Finds a random reference of a given base object within a given radius of a reference
3350 ObjectReference Function FindRandomReferenceOfAnyTypeInListFromRef(FormList
arBaseObjects, ObjectReference arCenter, float afRadius) global

```

```

3351         return FindRandomReferenceOfAnyTypeInList(arBaseObjects, arCenter.X, arCenter.Y,
3352             arCenter.Z, afRadius)
3353     endFunction
3354 ; Finds the closest actor within a given radius of a location
3355 Actor Function FindClosestActor(float afX, float afY, float afZ, float afRadius) native
3356 global
3357 ; Finds a random actor within a given radius of a location
3358 Actor Function FindRandomActor(float afX, float afY, float afZ, float afRadius) native
3359 global
3360 ; Finds the closest actor within a given radius of a reference
3361 Actor Function FindClosestActorFromRef(ObjectReference arCenter, float afRadius) global
3362     return FindClosestActor(arCenter.X, arCenter.Y, arCenter.Z, afRadius)
3363 endFunction
3364
3365 ; Finds a random actor within a given radius of a reference
3366 Actor Function FindRandomActorFromRef(ObjectReference arCenter, float afRadius) global
3367     return FindRandomActor(arCenter.X, arCenter.Y, arCenter.Z, afRadius)
3368 endFunction
3369
3370 ; Make the player got to 3rd person camera mode
3371 Function ForceThirdPerson() native global
3372
3373 ; Make the player got to 1st person camera mode
3374 Function ForceFirstPerson() native global
3375
3376 ; Show the players first person geometry.
3377 Function ShowFirstPersonGeometry( bool abShow = true ) native global
3378
3379 ; Returns the form specified by the ID
3380 Form Function GetForm(int aiFormID) native global
3381
3382 ; Returns the form specified by the ID originating in the given file
3383 Form Function GetFormFromFile(int aiFormID, string asFilename) native global
3384
3385 ; Obtains the value of a game setting - one for each type of game setting
3386 float Function GetGameSettingFloat(string asGameSetting) native global
3387 int Function GetGameSettingInt(string asGameSetting) native global
3388 string Function GetGameSettingString(string asGameSetting) native global
3389
3390 ; Returns the player actor
3391 Actor Function GetPlayer() native global
3392
3393 ; Returns the reference the player is currently grabbing
3394 ObjectReference Function GetPlayerGrabbedRef() native global
3395
3396 ; Returns the horse last ridden by the player
3397 Actor Function GetPlayersLastRiddenHorse() native global
3398
3399 ; Returns the X position of the Sun.
3400 float Function GetSunPositionX() native global
3401
3402 ; Returns the Y position of the Sun.
3403 float Function GetSunPositionY() native global
3404
3405 ; Returns the Z position of the Sun.
3406 float Function GetSunPositionZ() native global
3407
3408 ; Returns the number of days spent in play
3409 float Function GetRealHoursPassed() native global
3410
3411 ; Increment the given skill on the player by the one point
3412 Function IncrementSkill(string asSkillName) native global
3413
3414 ; Increment the given skill on the player by the given number of points
3415 Function IncrementSkillBy(string asSkillName, int aiCount) native global
3416

```

```

3417 ; Modifies the specified MiscStat by the given amount.
3418 Function IncrementStat(string asStatName, int aiModAmount = 1) native global
3419
3420 ; Are the activation controls enabled?
3421 bool Function IsActivateControlsEnabled() native global
3422
3423 ; Are the camera switch controls enabled?
3424 bool Function IsCamSwitchControlsEnabled() native global
3425
3426 ; Is fast travel controls enabled? Returns false if EnableFastTravel(false) has been
    called
3427 bool Function IsFastTravelControlsEnabled() native global
3428
3429 ; Is fast travel enabled?
3430 bool Function IsFastTravelEnabled() native global
3431
3432 ; Are the fighting controls enabled?
3433 bool Function IsFightingControlsEnabled() native global
3434
3435 ; Are the journal menu controls enabled?
3436 bool Function IsJournalControlsEnabled() native global
3437
3438 ; Are the looking controls enabled?
3439 bool Function IsLookingControlsEnabled() native global
3440
3441 ; Are the menu controls enabled?
3442 bool Function IsMenuControlsEnabled() native global
3443
3444 ; Are the movement controls enabled?
3445 bool Function IsMovementControlsEnabled() native global
3446
3447 ; Is the player looking at the sun?
3448 bool Function IsPlayerSungazing() native global
3449
3450 ; Are the sneaking controls enabled?
3451 bool Function IsSneakingControlsEnabled() native global
3452
3453 ; Is the specified Word of Power Unlocked?
3454 bool Function IsWordUnlocked(WordOfPower akWord) native global
3455
3456 ; Plays a blink video - does not return until blink has finished, use with care!
3457 Function PlayBink(string asFileName, bool abInterruptible = false, bool abMuteAudio =
    true, bool abMuteMusic = true, \
3458     bool abLetterbox = true ) native global
3459
3460 ; Precaches character gen data.
3461 Function PrecacheCharGen() native global
3462
3463 ; Clears Precached character gen data.
3464 Function PrecacheCharGenClear() native global
3465
3466 ; Queries the given stat and returns its value
3467 int Function QueryStat(string asStat) native global
3468
3469 ; Forces the game back to the main menu
3470 Function QuitToMainMenu() native global
3471
3472 ; Request that an auto-save be made
3473 Function RequestAutoSave() native global
3474
3475 ; Requests the specified model
3476 Function RequestModel(string asModelName) native global
3477
3478 ; Request that a normal save be made
3479 Function RequestSave() native global
3480
3481 ; Has the player serve their prison time
3482 Function ServeTime() native global
3483

```

```

3484 ; Finds an actor in high who can detect the player to call werewolf crime on the player
3485 Function SendWereWolfTransformation() native global
3486
3487 ; Called as we enter/exit beast form
3488 Function SetBeastForm(bool abEntering) native global
3489
3490 ; Sets the camera target actor
3491 Function SetCameraTarget(Actor arTarget) native global
3492
3493 ; Sets or clears "cart mode" for the HUD
3494 Function SetHudCartMode(bool abSetCartMode = true) native global
3495
3496 ; Informs the game whether we are in CharGen or not
3497 Function SetInChargen(bool abDisableSaving, bool abDisableWaiting, bool
abShowControlsDisabledMessage) native global
3498
3499 ; Enables or disables the AI driven flag on Player
3500 Function SetPlayerAIDriven(bool abAIDriven = true) native global
3501
3502 ; Enables or disables crime reporting on Player
3503 Function SetPlayerReportCrime(bool abReportCrime = true) native global
3504
3505 ; Set the players sitting camera rotation - in degrees, offset from the standard angle.
3506 Function SetSittingRotation(float afValue) native global
3507
3508 ; Shakes the object from the location of the passed-in object. If none, it will shake
the camera from the player's location.
3509 ; Strength is clamped from 0 to 1
3510 ; Duration in seconds. By default (0.0) use the game setting.
3511 Function ShakeCamera(ObjectReference akSource = None, float afStrength = 0.5, float
afDuration = 0.0) native global
3512
3513 ; Shakes the controller for the specified length of time (in seconds). The strength
values are clamped from 0 to 1
3514 Function ShakeController(float afSmallMotorStrength, float afBigMotorStreangth, float
afDuration) native global
3515
3516 ; Displays the race/sex menu
3517 Function ShowRaceMenu() native global
3518 Function ShowLimitedRaceMenu() native global
3519
3520 ; Title Sequence menu functions
3521 Function ShowTitleSequenceMenu() native global
3522 Function HideTitleSequenceMenu() native global
3523 Function StartTitleSequence(string asSequenceName) native global
3524
3525 ; Allow or disallow player requests to have a flying mount land.
3526 Function SetAllowFlyingMountLandingRequests(bool abAllow) native global
3527
3528 ; Sets the Image Space Modifier that is triggered when the player gazes at the sun.
3529 Function SetSunGazeImageSpaceModifier(ImageSpaceModifier apImod = NONE ) native global
3530
3531 ; Displays the training menu based on passed in trainer actor
3532 Function ShowTrainingMenu(Actor aTrainer) native global
3533
3534 ; Teaches the specified word of power to the player
3535 Function TeachWord(WordOfPower akWord) native global
3536
3537 ; Trigger screen blood with the given count
3538 Function TriggerScreenBlood(int aiValue) native global
3539
3540 ; Unlocks the specified word of power so the player can use it
3541 Function UnlockWord(WordOfPower akWord) native global
3542
3543 ; Returns true if we're using a gamepad
3544 bool Function UsingGamepad() native global
3545
3546
3547 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC

```

```

3548 ; Get/Set Perk Points
3549 int Function GetPerkPoints() global native
3550 Function SetPerkPoints(int perkPoints) global native
3551 Function ModPerkPoints(int perkPoints) global native
3552
3553 ; returns the number of active mods
3554 int Function GetModCount() native global
3555
3556 ; returns the index of the specified mod
3557 int Function GetModByName(string name) native global
3558
3559 ; returns the name of the mod at the specified modIndex
3560 string Function GetModName(int modIndex) native global
3561
3562 ; returns the author of the mod at the specified modIndex
3563 string Function GetModAuthor(int modIndex) native global
3564
3565 ; returns the description of the mod at the specified modIndex
3566 string Function GetModDescription(int modIndex) native global
3567
3568 ; gets the count of mods the specified mod depends upon
3569 int Function GetModDependencyCount(int modIndex) native global
3570
3571 ; gets the index of the nth mod dependency of the specified mod
3572 ; int Function GetNthModDependency(int modIndex, int n) native global
3573
3574 bool Function IsPluginInstalled(string name) native global
3575
3576 ; light mod functions
3577 int Function GetLightModCount() native global
3578 int Function GetLightModByName(string name) native global
3579 string Function GetLightModName(int idx) native global
3580 string Function GetLightModAuthor(int idx) native global
3581 string Function GetLightModDescription(int idx) native global
3582 int Function GetLightModDependencyCount(int idx) native global
3583 int Function GetNthLightModDependency(int modIdx, int idx) native global
3584
3585 ; GameSetting functions - SKSE 1.5.10
3586 Function SetGameSettingFloat(string setting, float value) global native
3587 Function SetGameSettingInt(string setting, int value) global native
3588 Function SetGameSettingBool(string setting, bool value) global native
3589 Function SetGameSettingString(string setting, string value) global native
3590
3591 ; save/load game
3592 Function SaveGame(string name) native global
3593 Function LoadGame(string name) native global
3594
3595 ; TintMasks (AARRGGBB)
3596
3597 ; Returns the total number of tints for the player
3598 int Function GetNumTintMasks() native global
3599
3600 ; Returns the color of the Nth tint mask
3601 int Function GetNthTintMaskColor(int n) native global
3602
3603 ; Returns the type of the Nth tint mask
3604 int Function GetNthTintMaskType(int n) native global
3605
3606 ; Sets the color of the Nth tint mask
3607 Function SetNthTintMaskColor(int n, int color) native global
3608
3609 ; Returns the texture path of the Nth tint mask
3610 string Function GetNthTintMaskTexturePath(int n) native global
3611
3612 ; Sets the texturepath of the Nth tint mask
3613 Function SetNthTintMaskTexturePath(string path, int n) native global
3614
3615 ; Types
3616 ; 0 - Frekles

```

```

3617 ; 1 - Lips
3618 ; 2 - Cheeks
3619 ; 3 - Eyeliner
3620 ; 4 - Upper Eyesocket
3621 ; 5 - Lower Eyesocket
3622 ; 6 - SkinTone
3623 ; 7 - Warpaint
3624 ; 8 - Frownlines
3625 ; 9 - Lower Cheeks
3626 ; 10 - Nose
3627 ; 11 - Chin
3628 ; 12 - Neck
3629 ; 13 - Forehead
3630 ; 14 - Dirt
3631
3632 ; Returns how many indexes there are for this type
3633 int Function GetNumTintsByType(int type) native global
3634
3635 ; Returns the color for the particular tintMask type and index
3636 int Function GetTintMaskColor(int type, int index) global native
3637
3638 ; Sets the tintMask color for the particular type and index
3639 Function SetTintMaskColor(int color, int type, int index) global native
3640
3641 ; Returns the texture path for the particular tintMask type and index
3642 string Function GetTintMaskTexturePath(int type, int index) global native
3643
3644 ; Sets the tintMask texture for the particular type and index
3645 Function SetTintMaskTexturePath(string path, int type, int index) global native
3646
3647 ; Updates tintMask colors without updating the entire model
3648 Function UpdateTintMaskColors() global native
3649
3650 ; Updates the players hair color immediately
3651 Function UpdateHairColor() global native
3652
3653 ; Returns the character's current camera state
3654 ; 0 - first person
3655 ; 1 - auto vanity
3656 ; 2 - VATS
3657 ; 3 - free
3658 ; 4 - iron sights
3659 ; 5 - furniture
3660 ; 6 - transition
3661 ; 7 - tweenmenu
3662 ; 8 - third person 1
3663 ; 9 - third person 2
3664 ; 10 - horse
3665 ; 11 - bleedout
3666 ; 12 - dragon
3667 int Function GetCameraState() global
3668     return Camera.GetCameraState()
3669 EndFunction
3670
3671 ; set a misc stat value
3672 ; use QueryStat to read the value
3673 Function SetMiscStat(string name, int value) global native
3674
3675 ; Sets the players last ridden horse, None will clear the lastRiddenHorse
3676 Function SetPlayersLastRiddenHorse(Actor horse) global native
3677
3678 ; Returns the legendary level for the skill
3679 ; -1 indicates the particular skill cannot have a legendary level
3680 ; DEPRECATED
3681 int Function GetSkillLegendaryLevel(string actorValue) global
3682     return ActorValueInfo.GetActorValueInfoByName(actorValue).GetSkillLegendaryLevel()
3683 EndFunction
3684
3685 ; Sets the legendary level for the skill

```



```

3686 ; DEPRECATED
3687 Function SetSkillLegendaryLevel(string actorValue, int level) global
3688     ActorValueInfo.GetActorValueInfoByName(actorValue).SetSkillLegendaryLevel(level)
3689 EndFunction
3690
3691 ; Returns the players experience for this level (not total experience)
3692 float Function GetPlayerExperience() global native
3693
3694 ; Sets the players experience, does not trigger level-up notification
3695 Function SetPlayerExperience(float exp) global native
3696
3697 ; Calculates the experience required for to level-up
3698 ; (fXPLevelUpBase + currentLevel * fXPLevelUpMult)
3699 float Function GetExperienceForLevel(int currentLevel) global native
3700
3701 ; Returns true if in run mode, false if in walk mode
3702 ; Does not reflect actual movement state, only the control mode
3703 bool Function GetPlayerMovementMode() global native
3704
3705 ; Updates the camera when changing Shoulder positions
3706 Function UpdateThirdPerson() global
3707     Camera.UpdateThirdPerson()
3708 EndFunction
3709
3710 ; Hotkeys 0-7 reflect keys 1-8
3711 ; Unbinds a favorited item bound to the specified hotkey
3712 Function UnbindObjectHotkey(int hotkey) global native
3713
3714 ; Returns the base form object that is bound to the specified hotkey
3715 Form Function GetHotkeyBoundObject(int hotkey) global native
3716
3717 ; Returns if base form is favorited by the player
3718 bool Function IsObjectFavorited(Form form) global native
3719
3720 ; Same as GetForm, but also works for formIds >= 0x80000000
3721 Form Function GetFormEx(int formId) global native
3722
3723 ; Returns the object reference the player is in dialogue with
3724 ObjectReference Function GetDialogueTarget() global native
3725
3726 ; Returns the current crosshair ref
3727 ObjectReference Function GetCurrentCrosshairRef() global native
3728
3729 ; Returns the currently selected ref in the console
3730 ObjectReference Function GetCurrentConsoleRef() global native
3731
3732 ; Sets the player level
3733 Function SetPlayerLevel(int level) global native    :: Add a newline between files
3734 Scriptname GameData Hidden
3735
3736 ; Keywords are AND operations, must have all listed keywords
3737 ; IgnoreTemplates will exclude items that are inherited from other items with slightly
    altered stats
3738 ; IgnoreEnchantments will exclude any item with an enchantment
3739 ; WeaponTypes are a bitfield, will filter weapons by type
3740 ; Add together to filter by multiple types
3741 int Property WeaponTypeHandToHand = 1 AutoReadOnly
3742 int Property WeaponTypeOneHandSword = 2 AutoReadOnly
3743 int Property WeaponTypeOneHandDagger = 4 AutoReadOnly
3744 int Property WeaponTypeOneHandAxe = 8 AutoReadOnly
3745 int Property WeaponTypeOneHandMace = 16 AutoReadOnly
3746 int Property WeaponTypeTwoHandSword = 32 AutoReadOnly
3747 int Property WeaponTypeTwoHandAxe = 64 AutoReadOnly
3748 int Property WeaponTypeBow = 128 AutoReadOnly
3749 int Property WeaponTypeStaff = 256 AutoReadOnly
3750 int Property WeaponTypeCrossbow = 512 AutoReadOnly
3751
3752 Form[] Function GetAllWeapons(string modName, Keyword[] keywords = None, bool playable =
    true, bool ignoreTemplates = true, bool ignoreEnchantments = true, bool onlyEnchanted =

```

```

false, int weaponTypes = 0xFFFFFFFF) global native
3753
3754 Form[] Function GetAllArmor(string modName, Keyword[] keywords = None, bool playable =
true, bool ignoreTemplates = true, bool ignoreEnchantments = true, bool onlyEnchanted =
false, bool ignoreSkin = true) global native
3755
3756 Form[] Function GetAllAmmo(string modName, Keyword[] keywords = None, bool playable =
true) global native
3757
3758 Form[] Function GetAllBooks(string modName, Keyword[] keywords = None, bool regular =
true, bool spell = false, bool skill = false) global native
3759
3760 Form[] Function GetAllPotions(string modName, Keyword[] keywords = None, bool potions =
true, bool food = false, bool poison = false) global native
3761
3762 Form[] Function GetAllIngredients(string modName, Keyword[] keywords = None) global
native
3763
3764 Form[] Function GetAllScrolls(string modName, Keyword[] keywords = None) global native
3765
3766 Form[] Function GetAllKeys(string modName, Keyword[] keywords = None) global native
3767
3768 Form[] Function GetAllMiscItems(string modName, Keyword[] keywords = None) global
native    :: Add a newline between files
3769 Scriptname HeadPart extends Form Hidden
3770
3771 int Property Type_Misc = 0 AutoReadOnly
3772 int Property Type_Face = 1 AutoReadOnly
3773 int Property Type_Eyes = 2 AutoReadOnly
3774 int Property Type_Hair = 3 AutoReadOnly
3775 int Property Type_FacialHair = 4 AutoReadOnly
3776 int Property Type_Scar = 5 AutoReadOnly
3777 int Property Type_Brows = 6 AutoReadOnly
3778
3779 HeadPart Function GetHeadPart(string name) native global
3780
3781 ; Returns the head part type
3782 int Function GetType() native
3783
3784 int Function GetNumExtraParts() native
3785 HeadPart Function GetNthExtraPart(int n) native
3786
3787 bool Function HasExtraPart(HeadPart p) native
3788 int Function GetIndexOfExtraPart(HeadPart p) native
3789
3790 ; Returns a formlist of the valid races for this head part
3791 FormList Function GetValidRaces() native
3792 Function SetValidRaces(FormList vRaces) native
3793
3794 ; Returns whether the head part is an extra part
3795 bool Function IsExtraPart() native
3796
3797 ; Returns the EditorID of the HeadPart
3798 string Function GetPartName() native    :: Add a newline between files
3799 Scriptname Ingredient extends Form
3800
3801 ; Is this ingredient classified as hostile?
3802 bool Function IsHostile() native
3803
3804 ; Flags the effect with the given 0 based index as known by the player
3805 Function LearnEffect(int aiIndex) native
3806
3807 ; Flags the next unknown effect as known by the player, returning index of effect learned
3808 int Function LearnNextEffect() native
3809
3810 ; Flags the all effects as known by the player
3811 Function LearnAllEffects() native
3812
3813 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC

```

```

3814 ; return the number of the effects
3815 int Function GetNumEffects() native
3816
3817 ; return the magnitude of the specified effect
3818 float Function GetNthEffectMagnitude(int index) native
3819
3820 ; return the area of the specified effect
3821 int Function GetNthEffectArea(int index) native
3822
3823 ; return the duration of the specified effect
3824 int Function GetNthEffectDuration(int index) native
3825
3826 ; return the magic effect of the specified effect
3827 MagicEffect Function GetNthEffectMagicEffect(int index) native
3828
3829 ; return the index of the costliest effect
3830 int Function GetCostliestEffectIndex() native
3831
3832 ; sets the magnitude of the specified effect
3833 Function SetNthEffectMagnitude(int index, float value) native
3834
3835 ; sets the area of the specified effect
3836 Function SetNthEffectArea(int index, int value) native
3837
3838 ; sets the duration of the specified effect
3839 Function SetNthEffectDuration(int index, int value) native
3840
3841 ; determines whether the player knows this effect
3842 bool Function GetIsNthEffectKnown(int index) native
3843
3844 ; Returns all the magnitudes of this object in order
3845 float[] Function GetEffectMagnitudes() native
3846
3847 ; Returns all the areas of this object in order
3848 int[] Function GetEffectAreas() native
3849
3850 ; Returns all the durations of this object in order
3851 int[] Function GetEffectDurations() native
3852
3853 ; Returns all the magic effects of this object in order
3854 MagicEffect[] Function GetMagicEffects() native :: Add a newline between files
3855 Scriptname Input Hidden
3856
3857 ; returns whether a key is pressed
3858 bool Function IsKeyPressed(Int dxKeycode) global native
3859
3860 ; taps the specified key
3861 Function TapKey(Int dxKeycode) global native
3862
3863 ; holds down the specified key until released
3864 Function HoldKey(Int dxKeycode) global native
3865
3866 ; releases the specified key
3867 Function ReleaseKey(Int dxKeycode) global native
3868
3869 ; how many keys are pressed
3870 int Function GetNumKeysPressed() global native
3871
3872 ; for walking over the pressed keys
3873 int Function GetNthKeyPressed(int n) global native
3874
3875 ; returns keycode bound to a control for given device
3876 ;
3877 ; Valid controls:
3878 ; "Forward", "Back", "Strafe Left", "Strafe Right", "Move", "Look", "Left
Attack/Block", "Right Attack/Block"
3879 ; "Activate", "Ready Weapon", "Tween Menu", "Toggle POV", "Zoom Out", "Zoom In",
"Jump", "Sprint", "Shout",
3880 ; "Sneak", "Run", "Toggle Always Run", "Auto-Move", "Favorites", "Hotkey1", "Hotkey2",

```

```

3881 "Hotkey3", "Hotkey4",
3882 ; "Hotkey5", "Hotkey6", "Hotkey7", "Hotkey8", "Quicksave", "Quickload", "Wait",
3883 ; "Journal", "Pause", "Screenshot",
3884 ; "Multi-Screenshot", "Console", "CameraPath", "Quick Inventory", "Quick Magic",
3885 ; "Quick Stats", "Quick Map"
3886 ;
3887 ; Valid device types:
3888 ; (default) auto detect
3889 ; 0 keyboard
3890 ; 1 mouse
3891 ; 2 gamepad
3892 int Function GetMappedKey(string control, int deviceType = 0xFF) global native
3893 ; returns name of control bound to given keycode, or "" if unbound
3894 string Function GetMappedControl(int keycode) global native :: Add a newline between
3895 files
3896 Scriptname Keyword Extends Form Hidden
3897 ; Sends this keyword as a story event to the story manager
3898 Function SendStoryEvent(Location akLoc = None, ObjectReference akRef1 = None,
3899 ObjectReference akRef2 = None, int aiValue1 = 0, \
3900 int aiValue2 = 0) native
3901 ; Sends this keyword as a story event to the story manager and waits for it to be
3902 processed. Returns true if a quest was started.
3903 bool Function SendStoryEventAndWait(Location akLoc = None, ObjectReference akRef1 =
3904 None, ObjectReference akRef2 = None, \
3905 int aiValue1 = 0, int aiValue2 = 0) native
3906 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
3907 ; return the keyword with the specified key
3908 Keyword Function GetKeyword(string key) global native
3909 ; return the string value of the keyword
3910 string Function GetString() native :: Add a newline between files
3911 Scriptname LevelledActor extends Form Hidden
3912 ; Adds the given count of the given form to the under the given level in this leveled
3913 list
3914 Function AddForm(Form apForm, int aiLevel) native
3915 ; Removes all script added forms from this leveled list
3916 Function Revert() native
3917 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
3918 int Function GetNumForms() native
3919 Form Function GetNthForm(int n) native
3920 int Function GetNthLevel(int n) native
3921 Function SetNthLevel(int n, int level) native
3922 int Function GetNthCount(int n) native
3923 Function SetNthCount(int n, int count) native :: Add a newline between files
3924 Scriptname LevelledItem extends Form Hidden
3925 ; Adds the given count of the given form to the under the given level in this leveled
3926 list
3927 Function AddForm(Form apForm, int aiLevel, int aiCount) native
3928 ; Removes all script added forms from this leveled list
3929 Function Revert() native
3930 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
3931 int function GetChanceNone() native
3932 Function SetChanceNone(int chance) native
3933 GlobalVariable Function GetChanceGlobal() native
3934 Function SetChanceGlobal(GlobalVariable glob) native
3935
3936
3937
3938
3939
3940

```

```

3941 int Function GetNumForms() native
3942 Form Function GetNthForm(int n) native
3943
3944 int Function GetNthLevel(int n) native
3945 Function SetNthLevel(int n, int level) native
3946
3947 int Function GetNthCount(int n) native
3948 Function SetNthCount(int n, int count) native    :: Add a newline between files
3949 Scriptname LeveledSpell extends Form Hidden
3950
3951 ; Adds the given count of the given form to the under the given level in this leveled
    list
3952 Function AddForm(Form apForm, int aiLevel) native
3953
3954 ; Removes all script added forms from this leveled list
3955 Function Revert() native
3956
3957 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
3958 int function GetChanceNone() native
3959 Function SetChanceNone(int chance) native
3960
3961 int Function GetNumForms() native
3962 Form Function GetNthForm(int n) native
3963
3964 int Function GetNthLevel(int n) native
3965 Function SetNthLevel(int n, int level) native
3966     :: Add a newline between files
3967 Scriptname Location extends Form Hidden
3968
3969 ; Returns the float value attached to the specified keyword attached to this location
3970 float Function GetKeywordData(Keyword akKeyword) native
3971
3972 ; Returns the number of alive references matching the specified reference type
3973 int Function GetRefTypeAliveCount(LocationRefType akRefType) native
3974
3975 ; Returns the number of dead references matching the specified reference type
3976 int Function GetRefTypeDeadCount(LocationRefType akRefType) native
3977
3978 ; Returns if these two locations have a common parent - filtered with the keyword, if
    provided
3979 bool Function HasCommonParent(Location akOther, Keyword akFilter = None) native
3980
3981 ; Returns if this location has the specified reference type
3982 bool Function HasRefType(LocationRefType akRefType) native
3983
3984 ; Returns whether this location is flagged as "cleared" or not
3985 bool Function IsCleared() native
3986
3987 ; Returns whether the other location is a child of this one
3988 bool Function IsChild(Location akOther) native
3989
3990 ; Is this location loaded in game?
3991 bool Function IsLoaded() native
3992
3993 bool Function IsSameLocation(Location akOtherLocation, Keyword akKeyword = None)
3994 {Returns true if the calling location is the same as the supplied location - if an
    optional keyword is supplied, it also returns true if the locations share a parent with
    that keyword, or if either location is a child of the other and the other has that
    keyword.}
3995 ;jduvall
3996     bool bmatching = self == akOtherLocation
3997     if !bmatching && akKeyword
3998         bmatching = HasCommonParent(akOtherLocation, akKeyword)
3999
4000     if !bmatching && akOtherLocation.HasKeyword(akKeyword)
4001         bmatching = akOtherLocation.IsChild(self)
4002     elseif !bmatching && self.HasKeyword(akKeyword)
4003         bmatching = self.IsChild(akOtherLocation)
4004     endif

```

```

4005
4006     endif
4007     return bmatching
4008 endFunction
4009
4010
4011 ; Sets the specified keyword's data on the location
4012 Function SetKeywordData(Keyword akKeyword, float afData) native
4013
4014 ; Sets this location as cleared or not
4015 Function SetCleared(bool abCleared = true) native
4016
4017 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
4018 Location Function GetParent() native    :: Add a newline between files
4019 Scriptname MagicEffect extends Form Hidden
4020 ; Get the Associated Skill for this MagicEffect
4021 string Function GetAssociatedSkill() native
4022
4023 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
4024 Function SetAssociatedSkill(string skill) native
4025
4026 string Function GetResistance() native
4027 Function SetResistance(string skill) native
4028
4029 ; Hostile           0x00000001
4030 ; Recover           0x00000002
4031 ; Detrimental       0x00000004
4032 ; NoHitEvent         0x00000010
4033 ; DispelKeywords     0x00000100
4034 ; NoDuration         0x00000200
4035 ; NoMagnitude        0x00000400
4036 ; NoArea             0x00000800
4037 ; FXPersist          0x00001000
4038 ; GloryVisuals       0x00004000
4039 ; HideInUI           0x00008000
4040 ; NoRecast           0x00020000
4041 ; Magnitude          0x00200000
4042 ; Duration           0x00400000
4043 ; Painless           0x04000000
4044 ; NoHitEffect        0x08000000
4045 ; NoDeathDispel      0x10000000
4046
4047 bool Function IsEffectFlagSet(int flag) native
4048 Function SetEffectFlag(int flag) native
4049 Function ClearEffectFlag(int flag) native
4050
4051 float Function GetCastTime() native
4052 Function SetCastTime(float castTime) native
4053
4054 int Function GetSkillLevel() native
4055 Function SetSkillLevel(int level) native
4056
4057 int Function GetArea() native
4058 Function SetArea(int area) native
4059
4060 float Function GetSkillUsageMult() native
4061 Function SetSkillUsageMult(float usageMult) native
4062
4063 float Function GetBaseCost() native
4064 Function SetBaseCost(float cost) native
4065
4066 Light Function GetLight() native
4067 Function SetLight(Light obj) native
4068
4069 EffectShader Function GetHitShader() native
4070 Function SetHitShader(EffectShader obj) native
4071
4072 EffectShader Function GetEnchantShader() native
4073 Function SetEnchantShader(EffectShader obj) native

```

```

4074
4075 Projectile Function GetProjectile() native
4076 Function SetProjectile(Projectile obj) native
4077
4078 Explosion Function GetExplosion() native
4079 Function SetExplosion(Explosion obj) native
4080
4081 Art Function GetCastingArt() native
4082 Function SetCastingArt(Art obj) native
4083
4084 Art Function GetHitEffectArt() native
4085 Function SetHitEffectArt(Art obj) native
4086
4087 Art Function GetEnchantArt() native
4088 Function SetEnchantArt(Art obj) native
4089
4090 ImpactDataSet Function GetImpactDataSet() native
4091 Function SetImpactDataSet(ImpactDataSet obj) native
4092
4093 Spell Function GetEquipAbility() native
4094 Function SetEquipAbility(Spell obj) native
4095
4096 ImageSpaceModifier Function GetImageSpaceMod() native
4097 Function SetImageSpaceMod(ImageSpaceModifier obj) native
4098
4099 Perk Function GetPerk() native
4100 Function SetPerk(Perk obj) native
4101
4102 int Function GetCastingType() native
4103 ; Constant Effect      0
4104 ; Fire And Forget      1
4105 ; Concentration        2
4106
4107 int Function GetDeliveryType() native
4108 ; Self                  0
4109 ; Contact                1
4110 ; Aimed                 2
4111 ; Target Actor          3
4112 ; Target Location       4
4113
4114 ; Entries will be None if there is no sound
4115 ; will always return an array of size 6
4116 Sound[] Function GetSounds() native
4117 ; Draw Sheathe         0
4118 ; Charge                1
4119 ; Ready                 2
4120 ; Release               3
4121 ; Loop                  4
4122 ; Hit                   5    :: Add a newline between files
4123 Scriptname Math Hidden
4124
4125 ; Calculates the absolute value of the passed in value - N for N, and N for (-N)
4126 float Function abs(float afValue) global native
4127
4128 ; Calculates the arccosine of the passed in value, returning degrees
4129 float Function acos(float afValue) global native
4130
4131 ; Calculates the arcsine of the passed in value, returning degrees
4132 float Function asin(float afValue) global native
4133
4134 ; Calculates the arctangent of the passed in value, returning degrees
4135 float Function atan(float afValue) global native
4136
4137 ; Calculates the ceiling of the passed in value - the smallest integer greater than or
equal to the value
4138 int Function Ceiling(float afValue) global native
4139
4140 ; Calculates the cosine of the passed in value (in degrees)
4141 float Function cos(float afValue) global native

```

```

4142
4143 ; Converts degrees to radians
4144 float Function DegreesToRadians(float afDegrees) global native
4145
4146 ; Calculates the floor of the passed in value - the largest integer less than or equal
to the value
4147 int Function Floor(float afValue) global native
4148
4149 ; Calculates x raised to the y power
4150 float Function pow(float x, float y) global native
4151
4152 ; Converts radians to degrees
4153 float Function RadiansToDegrees(float afRadians) global native
4154
4155 ; Calculates the sine of the passed in value (in degrees)
4156 float Function sin(float afValue) global native
4157
4158 ; Calculate the square root of the passed in value
4159 float Function sqrt(float afValue) global native
4160
4161 ; Calculates the tangent of the passed in value (in degrees)
4162 float Function tan(float afValue) global native
4163
4164 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
4165 int Function LeftShift(int value, int shiftBy) global native
4166 int Function RightShift(int value, int shiftBy) global native
4167 int Function LogicalAnd(int arg1, int arg2) global native
4168 int Function LogicalOr(int arg1, int arg2) global native
4169 int Function LogicalXor(int arg1, int arg2) global native
4170 int Function LogicalNot(int arg1) global native
4171 float Function Log(float arg1) global native    :: Add a newline between files
4172 Scriptname ModEvent Hidden
4173
4174 ; ModEvent allows sending mod events with any number/type of arguments, unlike the more
limited Form.SendModEvent.
4175 ;
4176 ; Example:
4177 ;
4178 ; (Sender)
4179 ;
4180 ;     int handle = ModEvent.Create("MYPREFIX_myCustomEvent")
4181 ;     if (handle)
4182 ;         ModEvent.PushForm(handle, self)
4183 ;         ModEvent.PushForm(handle, someOtherForm)
4184 ;         ModEvent.PushInt(handle, 1000)
4185 ;         ModEvent.PushString(handle, "It worked!")
4186 ;         UIDelegate.Send(handle)
4187 ;     endif
4188 ;
4189 ; (Receiver)
4190 ;
4191 ;     function OnInit()
4192 ;         RegisterForModEvent("MYPREFIX_myCustomEvent", "OnMyCustomEvent")
4193 ;     endFunction
4194 ;
4195 ;     event OnMyCustomEvent(Form sender, Form theForm, int theInt, string theString)
4196 ;         ; sender == (Sender)
4197 ;         ; theForm == someOtherForm
4198 ;         ; theInt == 1000
4199 ;         ; theString == "It worked!"
4200 ;     endEvent
4201
4202 ; Creates a new ModEvent and returns the handle.
4203 int Function Create(string eventName) global native
4204
4205 ; Sends the ModEvent and releases it.
4206 ; Returns true, if it was sent successfully, false if an error happened.
4207 bool Function Send(int handle) global native
4208

```



```

4209 ; Releases the ModEvent without sending it.
4210 Function Release(int handle) global native
4211
4212 ; Push single parameter.
4213 ;
4214 ; For arguments 1 .. N, the signature of the receiving event callback has to look like
this:
4215 ;
4216 ;   event MyCallback(TYPE_1 PARAM_1, ... , TYPE_N PARAM_N)
4217 ;
4218 Function PushBool(int handle, bool value) global native
4219 Function PushInt(int handle, int value) global native
4220 Function PushFloat(int handle, float value) global native
4221 Function PushString(int handle, string value) global native
4222 Function PushForm(int handle, Form value) global native
4223 :: Add a newline between files
4224 Scriptname NetImmerse Hidden
4225
4226 ; Note that only local transforms can be set as the world transform
4227 ; is computed based on the entire hierarchy rather than a single node
4228
4229 ; Return whether the object has the particular node
4230 bool Function HasNode(ObjectReference ref, string node, bool firstPerson) native global
4231
4232 ; NiNode Manipulation
4233 float Function GetNodeWorldPositionX(ObjectReference ref, string node, bool firstPerson)
native global
4234 float Function GetNodeWorldPositionY(ObjectReference ref, string node, bool firstPerson)
native global
4235 float Function GetNodeWorldPositionZ(ObjectReference ref, string node, bool firstPerson)
native global
4236
4237 ; Returns nodeB - nodeA
4238 float Function GetRelativeNodePositionX(ObjectReference ref, string nodeA, string nodeB,
bool firstPerson) native global
4239 float Function GetRelativeNodePositionY(ObjectReference ref, string nodeA, string nodeB,
bool firstPerson) native global
4240 float Function GetRelativeNodePositionZ(ObjectReference ref, string nodeA, string nodeB,
bool firstPerson) native global
4241
4242 float Function GetNodeLocalPositionX(ObjectReference ref, string node, bool firstPerson)
native global
4243 float Function GetNodeLocalPositionY(ObjectReference ref, string node, bool firstPerson)
native global
4244 float Function GetNodeLocalPositionZ(ObjectReference ref, string node, bool firstPerson)
native global
4245
4246 Function SetNodeLocalPositionX(ObjectReference ref, string node, float x, bool
firstPerson) native global
4247 Function SetNodeLocalPositionY(ObjectReference ref, string node, float y, bool
firstPerson) native global
4248 Function SetNodeLocalPositionZ(ObjectReference ref, string node, float z, bool
firstPerson) native global
4249
4250 ; Sets the scale of a particular Nif node
4251 float Function GetNodeScale(ObjectReference ref, string node, bool firstPerson) native
global
4252 Function SetNodeScale(ObjectReference ref, string node, float scale, bool firstPerson)
native global
4253
4254 ; Sets a NiTriShape's textures by name of the Nif node
4255 Function SetNodeTextureSet(ObjectReference ref, string node, TextureSet tSet, bool
firstPerson) native global
4256
4257
4258 ; Array based functions, return true when successful, false when unsuccessful (node did
not exist, or array wrong size)
4259
4260 ; returns the node's world position into the specify array, must be size of 3

```

```

4261 bool Function GetNodeWorldPosition(ObjectReference ref, string node, float[] in, bool
    firstPerson) native global
4262
4263 ; returns the node's relative world position of nodeB minus nodeA into the specify
    array, must be size of 3
4264 bool Function GetRelativeNodePosition(ObjectReference ref, string nodeA, string nodeB,
    float[] in, bool firstPerson) native global
4265
4266 ; returns the node's local position into the specify array, must be size of 3
4267 bool Function GetNodeLocalPosition(ObjectReference ref, string node, float[] in, bool
    firstPerson) native global
4268
4269 ; sets the node's local position of the specified array, must be size of 3
4270 bool Function SetNodeLocalPosition(ObjectReference ref, string node, float[] in, bool
    firstPerson) native global
4271
4272 ; Euler Rotation in DEGREES (heading, attitude, bank)
4273 ; returns the euler rotation of the node into the specified array, must be size of 3
4274 bool Function GetNodeWorldRotationEuler(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4275
4276 ; returns the euler rotation of the node into the specified array, must be size of 3
4277 bool Function GetNodeLocalRotationEuler(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4278
4279 ; sets the euler rotation for the node of the specified array, must be size of 3
4280 bool Function SetNodeLocalRotationEuler(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4281
4282 ; Matrix Rotation in RADIANS
4283 ; returns the matrix rotation of the node into the specified array, must be size of 9
4284 bool Function GetNodeWorldRotationMatrix(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4285
4286 ; returns the matrix rotation of the node into the specified array, must be size of 9
4287 bool Function GetNodeLocalRotationMatrix(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4288
4289 ; sets the matrix rotation for the node of the specified array, must be size of 9
4290 bool Function SetNodeLocalRotationMatrix(ObjectReference ref, string node, float[] in,
    bool firstPerson) native global
4291
4292
4293 ; DEPRECATED FUNCTIONS
4294 Function SetNodePositionX(ObjectReference ref, string node, float x, bool firstPerson)
    global
4295     NetImmerse.SetNodeLocalPositionX(ref, node, x, firstPerson)
4296 EndFunction
4297 Function SetNodePositionY(ObjectReference ref, string node, float y, bool firstPerson)
    global
4298     NetImmerse.SetNodeLocalPositionY(ref, node, y, firstPerson)
4299 EndFunction
4300 Function SetNodePositionZ(ObjectReference ref, string node, float z, bool firstPerson)
    global
4301     NetImmerse.SetNodeLocalPositionZ(ref, node, z, firstPerson)
4302 EndFunction
4303
4304 float Function GetNodePositionX(ObjectReference ref, string node, bool firstPerson)
    global
4305     return NetImmerse.GetNodeWorldPositionX(ref, node, firstPerson)
4306 EndFunction
4307 float Function GetNodePositionY(ObjectReference ref, string node, bool firstPerson)
    global
4308     return NetImmerse.GetNodeWorldPositionY(ref, node, firstPerson)
4309 EndFunction
4310 float Function GetNodePositionZ(ObjectReference ref, string node, bool firstPerson)
    global
4311     return NetImmerse.GetNodeWorldPositionZ(ref, node, firstPerson)
4312 EndFunction    :: Add a newline between files

```

```

4313 Scriptname ObjectReference extends Form Hidden
4314
4315 bool FUNCTION rampRumble(float power = 0.5, float duration = 0.25, float falloff =
1600.0)
4316     ; Function to shake cam/controller based on distance from player
4317     ; should always be called on the source of the rumble,
4318     ; as final intensity is relevant to player
4319     if power > 1.0 || power <= 0
4320     ;     debug.traceStack(self + " called rampRumble() but parameter 'power' was
invalid. Must be a non-zero float less than 1.0",1)
4321     ;     throw the warning, but don't return false - value gets clamped anyway
4322     endif
4323     float playerDist = game.getplayer().getDistance(self)
4324     ; ignore if the player is too far away
4325     if playerDist < falloff
4326         float intensity = (1 - (playerDist / falloff))
4327         ; ramp actual intensity down based on parameter value
4328         intensity = intensity*power
4329         if intensity > 1.0
4330             ; clamp to prevent invalid values
4331             ;     debug.traceStack(self + " called for too much controller/camera shake.
Clamped to 1.0", 0)
4332             intensity = 1.0
4333             elseif intensity <= 0
4334                 ; clamp to prevent invalid values
4335                 ;     debug.traceStack(self + " called for too little controller/camera shake", 0)
4336                 intensity = 0
4337                 return false
4338             endif
4339             game.shakeCamera(game.getPlayer(), intensity)
4340             game.shakeController(intensity, intensity, duration)
4341             return true
4342         else
4343             ;     debug.traceStack(self + "called for rampedRumble(), but player is too far away",
0)
4344             return False
4345         endif
4346     endif
4347 endFUNCTION
4348
4348 ; Function to know if I'm near the player (whether I can be safely enabled or disabled)
4349 bool Function IsNearPlayer()
4350     Actor player = Game.GetPlayer()
4351     Cell targetCell = self.GetParentCell()
4352     Cell playerCell = player.GetParentCell()
4353
4354     if (targetCell != playerCell)
4355         ; player and target are in different cells
4356         if (targetCell && targetCell.IsInterior() || playerCell &&
playerCell.IsInterior())
4357             ; in different cells and at least one is an interior
4358             ; -- we can safely enable or disable
4359             return false
4360         else
4361             ; both in an exterior -- no means of testing
4362             ; worldspace at the moment, so this will do.
4363             if (player.GetDistance(self) > 3000)
4364                 ; pretty darned far away -- safe
4365                 return false
4366             else
4367                 ; too close for comfort
4368                 return true
4369             endif
4370         endif
4371     else
4372         ; in the same cell -- err on the side of caution
4373         return true
4374     endif
4375 endFunction
4376

```

```

4377 ;jduvall
4378 bool Function IsInInterior()
4379 {Returns !IsInExterior() }
4380     Cell parentCell = GetParentCell()
4381     Return parentCell && parentCell.IsInterior()
4382 EndFunction
4383
4384 ;kkuhlmann:
4385 bool function MoveToIfUnloaded(ObjectReference akTarget, float afXOffset = 0.0, float
afYOffset = 0.0, float afZOffset = 0.0)
4386 {Calls MoveTo if the calling ObjectReference is currently unloaded. Doesn't do anything
if it IS loaded. No waiting or while loops. Returns true if it does the moveto}
4387     if !Is3DLoaded()
4388         MoveTo(akTarget, afXOffset, afYOffset, afZOffset)
4389         return true
4390     else
4391         return false
4392     endif
4393 EndFunction
4394
4395 ;jduvall:
4396 function MoveToWhenUnloaded(ObjectReference akTarget, float afXOffset = 0.0, float
afYOffset = 0.0, float afZOffset = 0.0)
4397 {DEPRECATED: DO NOT USE. Calls MoveTo if both the calling ObjectReference and the
akTarget ObjectReference have current locations that are not loaded.}
4398     while self.GetCurrentLocation().IsLoaded() ||
akTarget.GetCurrentLocation().IsLoaded()
4399         ;do nothing
4400 ;         debug.trace(self + "MoveToWhenUnloaded() waiting for current location and target
location to be unloaded before moving. If called by a quest stage fragment, this may
cause that quest stage to not complete until this function finishes (and if it's a
startup stage, the quest will not report itself as running until the stage finishes.).",
1)
4401         Utility.Wait(5) ;when this function is threaded we can increase this wait
time... I set it lower for testing purposes so it reevaluates faster when I need
to purge cell buffers in the Civil War when calling moveto on the player between
Civil War campaigns
4402     EndWhile
4403     self.MoveTo(akTarget, afXOffset, afYOffset, afZOffset)
4404 EndFunction
4405
4406 ;jduvall
4407 Function DeleteWhenAble()
4408 {This will become a native function... it will wait until the object is not persisting,
then delete itself.}
4409     While GetParentCell() && GetParentCell().IsAttached()
4410         ;do nothing
4411 ;         debug.trace(self + "DeleteWhenAble() waiting for current location to be unloaded
before deleting. If called by a quest stage fragment, this may cause that quest stage to
not complete until this function finishes (and if it's a startup stage, the quest will
not report itself as running until the stage finishes.).", 1)
4412         Utility.Wait(5) ;when this function is threaded we can increase this wait
time... I set it lower for testing purposes so it reevaluates faster when I need
to purge cell buffers in the Civil War when calling moveto on the player between
Civil War campaigns
4413     EndWhile
4414     Delete()
4415 EndFunction
4416
4417
4418
4419 ;jduvall
4420 Function AddKeyIfNeeded(ObjectReference ObjectWithNeededKey)
4421 {Should only be called by ObjectReferences that have/are containers (ie Containers and
Actors). Checks to see if self has the key to ObjectWithNeededKey, and if not, creates a
copy of the key and puts it in self.}
4422     key NeededKey = ObjectWithNeededKey.GetKey()
4423     if NeededKey != None
4424         if GetItemCount(NeededKey) == 0

```

```

4425         AddItem(NeededKey)
4426     EndIf
4427 EndIf
4428 EndFunction
4429
4430
4431 ; Property to obtain the current X position of the object
4432 float Property X
4433     float Function get()
4434         return GetPositionX()
4435     EndFunction
4436 EndProperty
4437
4438 ; Property to obtain the current Y position of the object
4439 float Property Y
4440     float Function get()
4441         return GetPositionY()
4442     EndFunction
4443 EndProperty
4444
4445 ; Property to obtain the current Z position of the object
4446 float Property Z
4447     float Function get()
4448         return GetPositionZ()
4449     EndFunction
4450 EndProperty
4451
4452 ; Have akActivator activate this reference. If abDefaultProcessingOnly is true then any
block will be bypassed
4453 ; and no OnActivate event will be sent. The function returns true if default processing
ran, and succeeded. If
4454 ; default processing has been blocked, will always return false.
4455 bool Function Activate(ObjectReference akActivator, bool abDefaultProcessingOnly =
false) native
4456
4457 ; Sets up a dependent animated object
4458 ; This function should be used only with a coder supervision. It is left undocumented
because it can cause dangling pointers as well as very broken functionality
4459 ; for the dependent object if used improperly.
4460 bool Function AddDependentAnimatedObjectReference( ObjectReference akDependent ) native
4461
4462 ; Add an inventory event filter to this reference. Item added/removed events matching the
4463 ; specified form (or in the specified form list) will now be let through.
4464 Function AddInventoryEventFilter(Form akFilter) native
4465
4466 ; Adds the specified base object or object reference to this object reference's
container/inventory
4467 ; Note that you cannot add more then one copy of a reference to a container (a warning
will be printed if you try)
4468 Function AddItem(Form akItemToAdd, int aiCount = 1, bool abSilent = false) native
4469
4470 ; Adds this reference (which is a map marker) to the map, optionally making it available
for fast travel
4471 Function AddToMap(bool abAllowFastTravel = false) native
4472
4473 ; Apply an impulse to this reference
4474 Function ApplyHavokImpulse(float afX, float afY, float afZ, float afMagnitude) native
4475
4476 ; Turns on and off blocking of normal activation - OnActivate events will still be sent
4477 Function BlockActivation(bool abBlocked = true) native
4478
4479 ; Calculate's this references encounter level based on the requested difficulty level
4480 ; 0 - Easy
4481 ; 1 - Medium
4482 ; 2 - Hard
4483 ; 3 - Very Hard
4484 ; 4 - None
4485 int Function CalculateEncounterLevel(int aiDifficulty = 4) native
4486

```

```
4487 ; Can the map marker be fast traveled to?
4488 bool Function CanFastTravelToMarker() native
4489
4490 ; Clears all effects of destruction from this object
4491 Function ClearDestruction() native
4492
4493 ; Create a detection event at this reference, with the specified owner. Sound level is
4494 between 0 and 100
4495 Function CreateDetectionEvent(Actor akOwner, int aiSoundLevel = 0 ) native
4496
4497 ; Damages this object and advances the destruction stage - does not return until the
4498 object is damaged
4499 Function DamageObject(float afDamage) native
4500
4501 ; Deletes this object
4502 Function Delete() native
4503
4504 ; Disables this object - fading out if requested
4505 Function Disable(bool abFadeOut = false) native
4506
4507 ; Disables this object - fading out if requested. Does NOT wait for the fade or disable
4508 to finish
4509 Function DisableNoWait(bool abFadeOut = false) native
4510
4511 ; Drops the specified object from this object's inventory
4512 ObjectReference Function DropObject(Form akObject, int aiCount = 1) native
4513
4514 ; Enables this object - fading in if requested
4515 Function Enable(bool abFadeIn = false) native
4516
4517 ; Enables the ability to fast travel to this marker - or disables it. Note that if you
4518 disable
4519 ; fast travel the player will see "You haven't discovered this location" as an error
4520 message
4521 Function EnableFastTravel(bool abEnable = true) native
4522
4523 ; Enables this object - fading in if requested. Does NOT wait for the fade or enable to
4524 finish
4525 Function EnableNoWait(bool abFadeIn = false) native
4526
4527 ; Forcibly adds / removes the ragdoll for a reference to the world
4528 Function ForceAddRagdollToWorld() native
4529 Function ForceRemoveRagdollFromWorld() native
4530
4531 ; Gets the actor that owns this object (or None if not owned by an Actor)
4532 ActorBase Function GetActorOwner() native
4533
4534 ; Get the current X angle of this object
4535 float Function GetAngleX() native
4536
4537 ; Get the current Y angle of this object
4538 float Function GetAngleY() native
4539
4540 ; Get the current Z angle of this object
4541 float Function GetAngleZ() native
4542
4543 ; Get a variable from the reference's animation graph (if applicable). Bool version.
4544 bool Function GetAnimationVariableBool(string arVariableName) native
4545
4546 ; Get a variable from the reference's animation graph (if applicable). Int version.
4547 int Function GetAnimationVariableInt(string arVariableName) native
4548
4549 ; Get a variable from the reference's animation graph (if applicable). Float version.
4550 float Function GetAnimationVariableFloat(string arVariableName) native
4551
4552 ; Returns the base object this reference represents
4553 Form Function GetBaseObject() native
4554
```

```

4550 ; Returns the object's current destruction stage
4551 int Function GetCurrentDestructionStage() native
4552
4553 ; Returns this reference's current location
4554 Location Function GetCurrentLocation() native
4555
4556 ; Returns the scene this reference is currently in - if any
4557 Scene Function GetCurrentScene() native
4558
4559 ; Calculates the distance between this reference and another - both must either be in
the same interior, or same worldspace
4560 float Function GetDistance(ObjectReference akOther) native
4561
4562 ; Returns this reference's editor location
4563 Location Function GetEditorLocation() native
4564
4565 ; Gets the faction that owns this object (or None if not owned by a Faction)
4566 Faction Function GetFactionOwner() native
4567
4568 ; Gets the angle between this object's heading and the other object in degrees - in the
range from -180 to 180
4569 float Function GetHeadingAngle(ObjectReference akOther) native
4570
4571 ; Get the current height of the object
4572 float Function GetHeight() native
4573
4574 ; Returns how many of the specified item is in this object reference's inventory
4575 int Function GetItemCount(Form akItem) native
4576
4577 ; Returns the smithed health of this object reference (1.0 == 100%)
4578 float Function GetItemHealthPercent() native
4579
4580 ; Returns the key base object that will unlock this object
4581 Key Function GetKey() native
4582
4583 ; Get the current length of the object
4584 float Function GetLength() native
4585
4586 ; Get our linked reference
4587 ObjectReference Function GetLinkedRef(Keyword apKeyword = NONE) native
4588
4589 ; Get the level of the lock on this object
4590 int Function GetLockLevel() native
4591
4592 ;jtucker, jduvall
4593 ;This function counts the number of linked refs that are in a linked Ref chain (ie
object is linked to A, A is linked to B, etc. this then counts all the linked refs.)
4594 ;Often used in conjunction with GetNthLinkedRef()
4595 ;*** WARNING: Having a link ref chain that at any point loops back on itself and calling
this function will result in very bad things. Don't do that!***
4596 int Function countLinkedRefChain(keyword apKeyword = None, int maxExpectedLinkedRefs =
100)
4597 ;Don't use this on a loop of linked refs.
4598 ObjectReference CurrentLink = self
4599 ObjectReference NewLink
4600 int NumLinkedRefs = 0
4601
4602 while(currentLink) && NumLinkedRefs <= maxExpectedLinkedRefs
4603
4604     NewLink = currentLink.getLinkedRef(apKeyword)
4605
4606     if NewLink != self
4607         currentLink = NewLink
4608         NumLinkedRefs = NumLinkedRefs + 1
4609     Else
4610         currentLink = None
4611 ;         debug.trace( self + "countLinkedRefs() found itself. This suggests it was
linked back to itself. This will create an infinite loop, so we are killing the function
now. NumLinkedRefs =" + NumLinkedRefs)

```

```

4612         EndIf
4613
4614     endwhile
4615
4616     if NumLinkedRefs >= maxExpectedLinkedRefs
4617 ;       debug.trace( self + "countLinkedRefs() bailing out early because it found more
linked refs than maxExpectedLinkRefs (suggesting an infinite loop). LinkedRefs found:" +
NumLinkedRefs + ", maxExpectedLinkedRefs:" + maxExpectedLinkedRefs)
4618     EndIf
4619
4620
4621     return NumLinkedRefs
4622
4623 endFunction
4624
4625
4626 ; Returns the Nth linked ref from this reference (0 = self, 1 = GetLinkedRef, 2 =
GetLinkedRef.GetLinkedRef, etc)
4627 ObjectReference Function GetNthLinkedRef(int aiLinkedRef) native
4628
4629
4630 ; Enables all of the references that are linked, in a chain, to this one.
4631 Function EnableLinkChain(Keyword apKeyword = None)
4632     ObjectReference CurrentLink = GetLinkedRef(apKeyword)
4633     While CurrentLink
4634         CurrentLink.Enable()
4635         CurrentLink = CurrentLink.GetLinkedRef(apKeyword)
4636     endwhile
4637 endFunction
4638
4639
4640 ; Disables all of the references that are linked, in a chain, to this one.
4641 Function DisableLinkChain(Keyword apKeyword = None, bool abFadeOut = false)
4642     ObjectReference CurrentLink = GetLinkedRef(apKeyword)
4643     While CurrentLink
4644         CurrentLink.Disable(abFadeOut)
4645         CurrentLink = CurrentLink.GetLinkedRef(apKeyword)
4646     endwhile
4647 endFunction
4648
4649
4650 ; Get this object's mass
4651 float Function GetMass() native
4652
4653 ; Gets the open state of this object. Which can be one of the following:
4654 ; 0 - None
4655 ; 1 - Open
4656 ; 2 - Opening
4657 ; 3 - Closed
4658 ; 4 - Closing
4659 int Function GetOpenState() native
4660
4661 ; Gets the cell this object is in
4662 Cell Function GetParentCell() native
4663
4664 ; Get the current X position of the object
4665 float Function GetPositionX() native
4666
4667 ; Get the current Y position of the object
4668 float Function GetPositionY() native
4669
4670 ; Get the current Z position of the object
4671 float Function GetPositionZ() native
4672
4673 ; Get the current scale of the object
4674 float Function GetScale() native
4675
4676 ; Get the number of objects inside this trigger (throws warning if not a trigger)
4677 int Function GetTriggerObjectCount() native

```



```

4678
4679 ; Gets the voice type for this reference. Will return None if not an actor or a talking
activator
4680 VoiceType Function GetVoiceType() native
4681
4682 ; Get the current width of the object
4683 float Function GetWidth() native
4684
4685 ; Get this objects worldspace
4686 WorldSpace Function GetWorldSpace() native
4687
4688 ; Returns self cast as an actor
4689 actor Function GetSelfAsActor()
4690     return self as Actor
4691 endFunction
4692
4693 ; Returns if this reference has an active effect coming from a magic effect with the
specified keyword attached
4694 bool Function HasEffectKeyword(Keyword akKeyword) native
4695
4696 ; Returns whether the reference has the given node
4697 bool Function HasNode(string asNodeName) native
4698
4699 ; Returns if this reference has the specified location ref type
4700 bool Function HasRefType(LocationRefType akRefType) native
4701
4702 ; Flags this reference as ignoring (or not ignoring) friendly hits
4703 Function IgnoreFriendlyHits(bool abIgnore = true) native
4704
4705 ; Interrupts any spell-casting this object may be doing
4706 Function InterruptCast() native
4707
4708 ; Checks to see if the passed in reference is the activate child of this one
4709 bool Function IsActivateChild(ObjectReference akChild) native
4710
4711 ; Checks to see if activation is currently blocked on this object
4712 bool Function IsActivationBlocked() native
4713
4714 ; Returns if the 3d for this object is loaded or not
4715 bool Function Is3DLoaded() native
4716
4717 ; Is this object currently flagged for delete?
4718 bool Function IsDeleted() native
4719
4720 ; Is this object currently disabled?
4721 bool Function IsDisabled() native
4722
4723 ; Because Shane got tired of remembering which way to call this
4724 bool Function IsEnabled()
4725     return !IsDisabled()
4726 EndFunction
4727
4728 ; Is any marker on this furniture in use?
4729 bool Function IsFurnitureInUse(bool abIgnoreReserved = false) native
4730
4731 ; Is a particular marker on this furniture in use?
4732 bool Function IsFurnitureMarkerInUse(int aiMarker, bool abIgnoreReserved = false) native
4733
4734 ; Is this object ignoring friendly hits?
4735 bool Function IsIgnoringFriendlyHits() native
4736
4737 ; Is this actor or talking activator currently talking to the player?
4738 bool Function IsInDialogueWithPlayer() native
4739
4740 ; Is the lock on this object broken?
4741 bool Function IsLockBroken() native
4742
4743 ; Is the lock on this object locked?
4744 bool Function IsLocked() native

```

```

4745
4746 ; Is the map marker visible?
4747 bool Function IsMapMarkerVisible() native
4748
4749 ; Executes a knock effect to an area
4750 Function KnockAreaEffect(float afMagnitude, float afRadius) native
4751
4752 ; Lock/unlock this object. If told to lock it, it will add a lock if it doesn't have
4753 ; one. If locked/unlocked as the owner on a door,
4754 ; the adjoining cell will be made public/private as appropriate
4755 Function Lock(bool abLock = true, bool abAsOwner = false) native
4756
4757 ; Moves this object to the position of the specified object, with an offset, and
4758 ; optionally matching its rotation
4759 Function MoveTo(ObjectReference akTarget, float afXOffset = 0.0, float afYOffset = 0.0,
4760 float afZOffset = 0.0, bool abMatchRotation = true) native
4761
4762 ; Moves this object to the position (and rotation) of the specified object's interaction
4763 ; position
4764 Function MoveToInteractionLocation(ObjectReference akTarget) native
4765
4766 ; Moves this object to its editor location
4767 Function MoveToMyEditorLocation() native
4768
4769 ; Moves this object to the position (and rotation) of the specified node on the
4770 ; specified object's 3D
4771 Function MoveToNode(ObjectReference akTarget, string asNodeName) native
4772
4773 ; Create x copies of the passed in form (forcing them to persist if desired) and place
4774 ; them at our location, returning the last object created
4775 ObjectReference Function PlaceAtMe(Form akFormToPlace, int aiCount = 1, bool
4776 abForcePersist = false, bool abInitiallyDisabled = false) native
4777
4778 ; Create an actor at this object's location. Level mod is one of the following:
4779 ; 0 - Easy
4780 ; 1 - Medium
4781 ; 2 - Hard
4782 ; 3 - Boss
4783 ; 4 - None
4784 Actor Function PlaceActorAtMe(ActorBase akActorToPlace, int aiLevelMod = 4,
4785 EncounterZone akZone = None) native
4786
4787 ; Start the specified animation playing - returns true if it succeeds
4788 bool Function PlayAnimation(string asAnimation) native
4789
4790 ; Start the specified animation playing and wait for the specified event - returns true
4791 ; if succeeds
4792 bool Function PlayAnimationAndWait(string asAnimation, string asEventName) native
4793
4794 ; Start the specified Gamebryo animation playing - returns true if it succeeds
4795 bool Function PlayGamebryoAnimation(string asAnimation, bool abStartOver = false, float
4796 afEaseInTime = 0.0) native
4797
4798 ; Play the specified impact effect - returns true if it succeeds
4799 bool Function PlayImpactEffect(ImpactDataSet akImpactEffect, string asNodeName = "",
4800 float afPickDirX = 0.0, float afPickDirY = 0.0, float afPickDirZ = -1.0, float
4801 afPickLength = 512.0, bool abApplyNodeRotation = false, bool abUseNodeLocalRotation =
4802 false) native
4803
4804 ; Play two animations at once - one on this object, one on another object
4805 bool Function PlaySyncedAnimationSS(string asAnimation1, ObjectReference akObj2, string
4806 asAnimation2) native
4807
4808 ; Play two animations at once - one on this object, one on another object - and wait for
4809 ; both
4810 bool Function PlaySyncedAnimationAndWaitSS(string asAnimation1, string asEvent1,
4811 ObjectReference akObj2, string asAnimation2, string asEvent2) native
4812
4813 ; Play a terrain effect that is attached to the specified bone of this object.

```

```

4798 Function PlayTerrainEffect(string asEffectModelName, string asAttachBoneName) native
4799
4800 ; Tells this object to process a trap hitting it
4801 Function ProcessTrapHit(ObjectReference akTrap, float afDamage, float afPushback, float
afXVel, float afYVel, float afZVel, float afXPos, float afYPos, float afZPos, int
aeMaterial, float afStagger) native
4802
4803 ; Pushes the passed-in actor away from this object, using the passed in knockback force
to determine the speed
4804 Function PushActorAway(Actor akActorToPush, float aiKnockbackForce) native
4805
4806 ; Remove all inventory event filters from this reference - all item added/removed events
will now be received
4807 Function RemoveAllInventoryEventFilters() native
4808
4809 ; Removes all items from this container, transferring it to the other object if passed
4810 Function RemoveAllItems(ObjectReference akTransferTo = None, bool abKeepOwnership =
false, bool abRemoveQuestItems = false) native
4811
4812 ; Remove an inventory event filter from this reference. Item added/removed events
matching the
4813 ; specified form (or in the specified form list) will no longer be let through.
4814 Function RemoveInventoryEventFilter(Form akFilter) native
4815
4816 ; Removes the specified item from this object reference's inventory
4817 Function RemoveItem(Form akItemToRemove, int aiCount = 1, bool abSilent = false,
ObjectReference akOtherContainer = None) native
4818
4819 ; Removes a previously added dependent object
4820 ; This function should be used only with a coder supervision. It is left undocumented
because it can cause dangling pointers as well as very broken functionality
4821 ; for the dependent object if used improperly.
4822 bool Function RemoveDependentAnimatedObjectReference( ObjectReference akDependent )
native
4823
4824 ; Resets this object, optional place the object at the new target
4825 Function Reset(ObjectReference akTarget = None) native
4826
4827 ; Has this object "say" the specified topic, as if spoken by the specified actor (if one
is
4828 ; provided, and potentially "speaking" in the player's head.
4829 Function Say(Topic akTopicToSay, Actor akActorToSpeakAs = None, bool
abSpeakInPlayersHead = false) native
4830
4831 ; Has this object behave as if the specified actor attempted to steal it
4832 Function SendStealAlarm(Actor akThief) native
4833
4834 ; Sets this object's actor cause to the specified actor
4835 Function SetActorCause(Actor akActor) native
4836
4837 ; Sets this object's owner to the specified actor base - None means to remove ownership
4838 Function SetActorOwner(ActorBase akActorBase) native
4839
4840 ; Set the orientation of the object (angles are in degrees)
4841 Function SetAngle(float afXAngle, float afYAngle, float afZAngle) native
4842
4843 ; Set a variable on the reference's animation graph (if applicable). Bool version.
4844 Function SetAnimationVariableBool(string arVariableName, bool abNewValue) native
4845
4846 ; Set a variable on the reference's animation graph (if applicable). Int version.
4847 Function SetAnimationVariableInt(string arVariableName, int aiNewValue) native
4848
4849 ; Set a variable on the reference's animation graph (if applicable). Float version.
4850 Function SetAnimationVariableFloat(string arVariableName, float afNewValue) native
4851
4852 ; Sets this object as destroyed or not
4853 Function SetDestroyed(bool abDestroyed = true) native
4854
4855 ; Sets this object's owner to the specified faction

```

```

4856 Function SetFactionOwner(Faction akFaction) native
4857
4858 ; Sets the lock level on this object. Will add an unlocked lock to it if it doesn't have
one
4859 Function SetLockLevel(int aiLockLevel) native
4860
4861 ; Sets the motion type of the reference
4862 ; aeMotionType: The type of motion (see properties at end of file)
4863 ; abAllowActivate: When setting to a dynamic type, allows the simulation to be activated
4864 Function SetMotionType(int aeMotionType, bool abAllowActivate = true) native
4865
4866 ; Sets this object reference as one that teammates will refuse to do favors on
4867 Function SetNoFavorAllowed(bool abNoFavor = true) native
4868
4869 ; Opens/closes this object
4870 Function SetOpen(bool abOpen = true) native
4871
4872 ; Set the position of the object
4873 Function SetPosition(float afX, float afY, float afZ) native
4874
4875 ; Set the current scale of the object
4876 Function SetScale(float afScale) native
4877
4878 ; Makes the reference translate to the given position/orientation
4879 ; Note: Rotation speed is entirely dependent on the length of the path and the movement
speed
4880 ; that is, the rotation will happen such that the reference reaches the goal orientation
at the end
4881 ; of the translation.
4882 Function TranslateTo(float afX, float afY, float afZ, float afXAngle, float afYAngle,
float afZAngle, float afSpeed, float afMaxRotationSpeed = 0.0) native
4883
4884 ; Makes the reference translate to the given position/orientation on a spline
4885 Function SplineTranslateTo(float afX, float afY, float afZ, float afXAngle, float
afYAngle, float afZAngle, float afTangentMagnitude, float afSpeed, float
afMaxRotationSpeed = 0.0) native
4886
4887 ; Makes the reference translate to the target node's ref/orient on a spline at the given
speed
4888 Function SplineTranslateToRefNode(ObjectReference arTarget, string arNodeName, float
afTangentMagnitude, float afSpeed, float afMaxRotationSpeed = 0.0) native
4889
4890 ; Stops the reference from moving
4891 Function StopTranslation() native
4892
4893 ; Makes the reference translate to the target ref position/orient at the given speed
4894 Function TranslateToRef(ObjectReference arTarget, float afSpeed, float
afMaxRotationSpeed = 0.0)
4895     TranslateTo(arTarget.X, arTarget.Y, arTarget.Z, arTarget.GetAngleX(),
arTarget.GetAngleY(), arTarget.GetAngleZ(), afSpeed, afMaxRotationSpeed)
4896 endFunction
4897
4898 ; Makes the reference translate to the target ref position/orient on a spline at the
given speed
4899 Function SplineTranslateToRef(ObjectReference arTarget, float afTangentMagnitude, float
afSpeed, float afMaxRotationSpeed = 0.0)
4900     SplineTranslateTo(arTarget.X, arTarget.Y, arTarget.Z, arTarget.GetAngleX(),
arTarget.GetAngleY(), arTarget.GetAngleZ(), afTangentMagnitude, afSpeed,
afMaxRotationSpeed)
4901 endFunction
4902
4903 ; Tether a prisoner cart to the given horse.
4904 Function TetherToHorse(ObjectReference akHorse) native
4905
4906 ; Waits for the animation graph to send the specified event
4907 bool Function WaitForAnimationEvent(string asEventName) native
4908
4909 ; Convenience function to check if I'm in a location or any of its children
4910 bool Function IsInLocation(Location akLocation)

```

```

4911      ; cache current location to avoid changing location while this function is running
4912      (surprisingly that seems to be happening occasionally)
4913      Location currLoc = GetCurrentLocation()
4914      if currLoc == None
4915          return false
4916      else
4917          return akLocation.IsChild(currLoc) || currLoc == akLocation
4918      endif
4919  endFunction
4920
4921  ; Event received when this reference is activated
4922  Event OnActivate(ObjectReference akActionRef)
4923  EndEvent
4924
4925  ; Event received when this object has moved to an attached cell from a detached one
4926  Event OnAttachedToCell()
4927  EndEvent
4928
4929  ; Event received when this object's parent cell is attached
4930  Event OnCellAttach()
4931  EndEvent
4932
4933  ; Event received when this object's parent cell is detached
4934  Event OnCellDetach()
4935  EndEvent
4936
4937  ; Event received when every object in this object's parent cell is loaded (TODO: Find
4938  restrictions)
4939  Event OnCellLoad()
4940  EndEvent
4941
4942  ; Event received when this object is closed
4943  Event OnClose(ObjectReference akActionRef)
4944  EndEvent
4945
4946  ; Event received when this object enters, exits, or changes containers
4947  Event OnContainerChanged(ObjectReference akNewContainer, ObjectReference akOldContainer)
4948  EndEvent
4949
4950  ; Event received when this reference's destruction stage has changed
4951  Event OnDestructionStageChanged(int aiOldStage, int aiCurrentStage)
4952  EndEvent
4953
4954  ; Event recieved when this object moves to a detached cell from an attached one
4955  Event OnDetachedFromCell()
4956  EndEvent
4957
4958  ; Event received when this object is equipped by an actor
4959  Event OnEquipped(Actor akActor)
4960  EndEvent
4961
4962  ; Event received when this object is grabbed by the player
4963  Event OnGrab()
4964  EndEvent
4965
4966  ; Event received when a this trigger is tripped
4967  Event OnTrigger(ObjectReference akActionRef)
4968  EndEvent
4969
4970  ; Event received when this trigger volume is entered
4971  Event OnTriggerEnter(ObjectReference akActionRef)
4972  EndEvent
4973
4974  ; Event received when this trigger volume is left
4975  Event OnTriggerLeave(ObjectReference akActionRef)
4976  EndEvent
4977
4978  ; Event received when this object is hit by a source (weapon, spell, explosion) or
4979  projectile attack

```

```
4977 Event OnHit(ObjectReference akAggressor, Form akSource, Projectile akProjectile, bool
4978 abPowerAttack, bool abSneakAttack, bool abBashAttack, bool abHitBlocked)
4979 EndEvent
4980 ; Event received when an item is added to this object's inventory. If the item is a
4981 persistant reference, akItemReference will
4982 ; point at it - otherwise the parameter will be None
4983 Event OnItemAdded(Form akBaseItem, int aiItemCount, ObjectReference akItemReference,
4984 ObjectReference akSourceContainer)
4985 EndEvent
4986 ; Event received when an item is removed from this object's inventory. If the item is a
4987 persistant reference, akItemReference
4988 ; will point at it - otherwise the parameter will be None
4989 Event OnItemRemoved(Form akBaseItem, int aiItemCount, ObjectReference akItemReference,
4990 ObjectReference akDestContainer)
4991 EndEvent
4992 ; Event recieved when this object is completely loaded - will be fired every time this
4993 object is loaded
4994 Event OnLoad()
4995 EndEvent
4996 ; Event received when the lock on this object changes
4997 Event OnLockStateChanged()
4998 EndEvent
4999 ; Event received when a magic affect is being applied to this object
5000 Event OnMagicEffectApply(ObjectReference akCaster, MagicEffect akEffect)
5001 EndEvent
5002 ; Event received when this object is opened
5003 Event OnOpen(ObjectReference akActionRef)
5004 EndEvent
5005 ; Event received when this object, if a book, is read
5006 Event OnRead()
5007 EndEvent
5008 ; Event received when this object is released by the player
5009 Event OnRelease()
5010 EndEvent
5011 ; Event received when this reference is reset
5012 Event OnReset()
5013 EndEvent
5014 ; Event received when this reference is sold by an actor
5015 Event OnSell(Actor akSeller)
5016 EndEvent
5017 ; Event received when a spell is cast by this object
5018 Event OnSpellCast(Form akSpell)
5019 EndEvent
5020 ; Event received when translation is almost complete (from a call to TranslateTo,
5021 "almost" is determined by a gamesetting, default is 90% of the way)
5022 Event OnTranslationAlmostComplete()
5023 EndEvent
5024 ; Event received when translation is complete (from a call to TranslateTo)
5025 Event OnTranslationComplete()
5026 EndEvent
5027 ; Event received when translation is aborted (from a call to StopTranslateTo)
5028 Event OnTranslationFailed()
5029 EndEvent
5030 ; Event recieved when this reference hits a target
```

```

5039 Event OnTrapHit(ObjectReference akTarget, float afXVel, float afYVel, float afZVel,
5040 float afXPos, float afYPos, float afZPos, \
5041 int aeMaterial, bool abInitialHit, int aeMotionType)
5042 EndEvent
5043 ; Event recieved when this starts hitting a target
5044 Event OnTrapHitStart(ObjectReference akTarget, float afXVel, float afYVel, float afZVel,
5045 float afXPos, float afYPos, float afZPos, \
5046 int aeMaterial, bool abInitialHit, int aeMotionType)
5047 EndEvent
5048 ; Event recieved when this stops hitting a target
5049 Event OnTrapHitStop(ObjectReference akTarget)
5050 EndEvent
5051
5052 ; Event received when this object is unequipped by an actor
5053 Event OnUnequipped(Actor akActor)
5054 EndEvent
5055
5056 ; Event recieved when this object is being unloaded - will be fired every time this
5057 object is unloaded
5058 Event OnUnload()
5059 EndEvent
5060
5061 ; Event received when this object's Ward is hit by a spell
5062 Event OnWardHit(ObjectReference akCaster, Spell akSpell, int aiStatus)
5063 EndEvent
5064
5065 ; Set of read-only properties to essentially make a fake enum for motion types passed in
5066 to the trap hit
5067 int Property Motion_Dynamic = 1 AutoReadOnly
5068 int Property Motion_SphereIntertia = 2 AutoReadOnly
5069 int Property Motion_BoxIntertia = 3 AutoReadOnly
5070 int Property Motion_Keyframed = 4 AutoReadOnly
5071 int Property Motion_Fixed = 5 AutoReadOnly
5072 int Property Motion_ThinBoxIntertia = 6 AutoReadOnly
5073 int Property Motion_Character = 7 AutoReadOnly
5074
5075 ; added in 1.6.1126
5076 Bool Function IsContainerEmpty() Native
5077 Function RemoveAllStolenItems(ObjectReference akTransferTo) Native
5078 Function SetContainerAllowStolenItems(Bool setAllowStolenItems) Native
5079 Int Function GetAllItemsCount() Native
5080
5081 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5082
5083 ; Container-only functions
5084 int Function GetNumItems() native
5085 Form Function GetNthForm(int index) native
5086 float Function GetTotalItemWeight() native
5087 float Function GetTotalArmorWeight() native
5088
5089 ; Tree and Flora only functions
5090 bool Function IsHarvested() native
5091 Function SetHarvested(bool harvested) native
5092
5093 ; Tempering
5094 Function SetItemHealthPercent(float health) native
5095
5096 ; Charges
5097
5098 ; Only works on ObjectReferences that have user-enchants
5099 Function SetItemMaxCharge(float maxCharge) native
5100 ; Works on any enchanted item
5101 float Function GetItemMaxCharge() native
5102
5103 float Function GetItemCharge() native
5104 Function SetItemCharge(float charge) native

```

```

5104
5105 Function ResetInventory() native
5106
5107 bool Function IsOffLimits() native
5108
5109 ; Returns the name of this reference
5110 ; this is the name that is displayed
5111 string Function GetDisplayName() native
5112
5113 ; Sets a reference's display name
5114 ; returns false if force is false and the reference
5115 ; is held by an alias using 'Stored Text' or 'Uses Stored Text'
5116 ; Text Replacement does not use this name and may be lost if forced
5117 bool Function SetDisplayName(string name, bool force = false) native
5118
5119 ; Returns the enable parent object
5120 ObjectReference Function GetEnableParent() native
5121
5122 ; Returns the player-made enchantment if there is one
5123 Enchantment Function GetEnchantment() native
5124
5125 ; Changes an item's player-made enchantment to something else
5126 ; None enchantment will remove the existing enchantment
5127 ; does not delete the custom enchantment, only removes it
5128 Function SetEnchantment(Enchantment source, float maxCharge) native
5129
5130 ; Creates a new enchantment on the item given the specified parameters
5131 ; all arrays must be the same size
5132 ; created enchantments are not purged from the save when removed or overwritten
5133 ; exact same enchantments are re-used by the game
5134 Function CreateEnchantment(float maxCharge, MagicEffect[] effects, float[] magnitudes,
5135 int[] areas, int[] durations) native
5136
5137 ; Returns the number of ref aliases holding this reference
5138 int Function GetNumReferenceAliases() native
5139
5140 ; Returns the nth ReferenceAlias holding this reference
5141 ReferenceAlias Function GetNthReferenceAlias(int n) native
5142
5143 ; Returns the poison applied to the weapon
5144 Potion Function GetPoison() native
5145
5146 ; Returns all base forms in the inventory/container into the specified FormList
5147 Function GetAllForms(FormList toFill) native
5148
5149 ; Returns all base forms from the container into a new array
5150 Form[] Function GetContainerForms() native
5151
5152 ; Returns all of the aliases holding this reference
5153 ReferenceAlias[] Function GetReferenceAliases() native      :: Add a newline between files
5154 Scriptname Outfit extends Form Hidden
5155
5156 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5157
5158 int Function GetNumParts() native
5159 Form Function GetNthPart(int n) native      :: Add a newline between files
5160 Scriptname Perk extends Form Hidden
5161
5162 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5163 Perk Function GetNextPerk() native
5164
5165 int Function GetNumEntries() native
5166
5167 int Function GetNthEntryRank(int n) native
5168 bool Function SetNthEntryRank(int n, int rank) native
5169
5170 int Function GetNthEntryPriority(int n) native

```



```

5172 bool Function SetNthEntryPriority(int n, int priority) native
5173
5174 Quest Function GetNthEntryQuest(int n) native
5175 bool Function SetNthEntryQuest(int n, Quest newQuest) native
5176
5177 int Function GetNthEntryStage(int n) native
5178 bool Function SetNthEntryStage(int n, int stage) native
5179
5180 Spell Function GetNthEntrySpell(int n) native
5181 bool Function SetNthEntrySpell(int n, Spell newSpell) native
5182
5183 LeveledItem Function GetNthEntryLeveledList(int n) native
5184 bool Function SetNthEntryLeveledList(int n, LeveledItem lList) native
5185
5186 string Function GetNthEntryText(int n) native
5187 bool Function SetNthEntryText(int n, string newText) native
5188
5189 float Function GetNthEntryValue(int n, int i) native
5190 bool Function SetNthEntryValue(int n, int i, float value) native    :: Add a newline
    between files
5191 Scriptname Potion extends Form
5192
5193 ; Is this postion classified as hostile?
5194 bool Function IsHostile() native
5195
5196 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5197 ; Is this potion classified as Food?
5198 bool Function IsFood() native
5199
5200 ; Is this potion classified as Poison?
5201 bool Function IsPoison() native
5202
5203 ; return the number of the effects
5204 int Function GetNumEffects() native
5205
5206 ; return the magnitude of the specified effect
5207 float Function GetNthEffectMagnitude(int index) native
5208
5209 ; return the area of the specified effect
5210 int Function GetNthEffectArea(int index) native
5211
5212 ; return the duration of the specified effect
5213 int Function GetNthEffectDuration(int index) native
5214
5215 ; return the magic effect of the specified effect
5216 MagicEffect Function GetNthEffectMagicEffect(int index) native
5217
5218 ; return the index of the costliest effect
5219 int Function GetCostliestEffectIndex() native
5220
5221 ; sets the magnitude of the specified effect
5222 Function SetNthEffectMagnitude(int index, float value) native
5223
5224 ; sets the area of the specified effect
5225 Function SetNthEffectArea(int index, int value) native
5226
5227 ; sets the duration of the specified effect
5228 Function SetNthEffectDuration(int index, int value) native
5229
5230 ; gets the use sound of this potion
5231 SoundDescriptor Function GetUseSound() native
5232
5233 ; Returns all the magnitudes of this object in order
5234 float[] Function GetEffectMagnitudes() native
5235
5236 ; Returns all the areas of this object in order
5237 int[] Function GetEffectAreas() native
5238
5239 ; Returns all the durations of this object in order

```

```

5240 int[] Function GetEffectDurations() native
5241
5242 ; Returns all the magic effects of this object in order
5243 MagicEffect[] Function GetMagicEffects() native    :: Add a newline between files
5244 Scriptname Quest extends Form Hidden
5245
5246 ; non-native functions
5247
5248 ; thread-safe way to modify a global value
5249 ; optional parameters:
5250 ; aiObjectiveID = objective ID to redisplay
5251 ; afTargetValue = value you're counting up (or down) towards -- if included, function
5252 will return TRUE when the global reaches the target value
5253 ; abCountingUp = by default, function assumes you're counting up towards the target
5254 value; make this false to count DOWN towards target value
5255 ; abCompleteObjective = by default, function assumes you're completing the objective
5256 once you reach the target value; make this false to FAIL the objective
5257 ; abRedisplayObjective = by default, function assume you want to redisplay the
5258 objective every time the global is incremeneted; make this FALSE to only display the
5259 objectives on complete or failure
5260 bool Function ModObjectiveGlobal(float afModValue, GlobalVariable aModGlobal, int
5261 aiObjectiveID = -1, float afTargetValue = -1.0, bool abCountingUp = true, bool
5262 abCompleteObjective = true, bool abRedisplayObjective = true)
5263     aModGlobal.Mod(afModValue)
5264     UpdateCurrentInstanceGlobal(aModGlobal)
5265     if aiObjectiveID >= 0
5266         ; display/complete objectives automatically
5267         if afTargetValue > -1
5268             if (abCountingUp && aModGlobal.value >= afTargetValue) || (!abCountingUp &&
5269 aModGlobal.value <= afTargetValue)
5270                 if (abCompleteObjective)
5271                     ; complete objective
5272                     SetObjectiveCompleted(aiObjectiveID)
5273                     return true
5274                 Else
5275                     ; fail objective
5276                     SetObjectiveFailed(aiObjectiveID)
5277                     return true
5278             Endif
5279         elseif (abRedisplayObjective)
5280             ; redisplay objective
5281             SetObjectiveDisplayed(aiObjectiveID, true, true)
5282         Else
5283             SetObjectiveDisplayed(aiObjectiveID, true, false)
5284         endif
5285     elseif (abRedisplayObjective)
5286         ; no target value, always redisplay objective
5287         SetObjectiveDisplayed(aiObjectiveID, true, true)
5288     Else
5289         SetObjectiveDisplayed(aiObjectiveID, true, false)
5290     endif
5291     endif
5292     return false
5293 endFunction
5294
5295 ; native functions
5296
5297 ; Flags all objectives as complete
5298 Function CompleteAllObjectives() native
5299
5300 ; Flags this quest as completed
5301 Function CompleteQuest() native
5302
5303 ; Flags all objectives as failed
5304 Function FailAllObjectives() native
5305
5306 ; Obtains the specified alias on the quest
5307 Alias Function GetAlias(int aiAliasID) native

```

```
5301
5302 ; Obtains the id of the highest completed stage on this quest
5303 int Function GetCurrentStageID() native
5304
5305 ; Alias for GetCurrentStage - obtains the highest completed stage on this quest
5306 int Function GetStage()
5307     return GetCurrentStageID()
5308 EndFunction
5309
5310 ; Alias for IsStageDone - checks to see whether the given stage is done or not
5311 bool Function GetStageDone(int aiStage)
5312     return IsStageDone(aiStage)
5313 EndFunction
5314
5315 ; Is this quest "active" (tracked by the player)?
5316 bool Function IsActive() native
5317
5318 ; Checks to see if the quest is completed
5319 bool Function IsCompleted() native
5320
5321 ; Checks to see if the specified objective is completed
5322 bool Function IsObjectiveCompleted(int aiObjective) native
5323
5324 ; Checks to see if the specified objective is displayed
5325 bool Function IsObjectiveDisplayed(int aiObjective) native
5326
5327 ; Checks to see if the specified objective is failed
5328 bool Function IsObjectiveFailed(int aiObjective) native
5329
5330 ; Checks to see if the quest is running
5331 bool Function IsRunning() native
5332
5333 ; Obtains whether the specified stage is done or not
5334 bool Function IsStageDone(int aiStage) native
5335
5336 ; Checks to see if the quest is enabled but not running yet
5337 bool Function IsStarting() native
5338
5339 ; Checks to see if the quest is not enabled anymore but still shutting down
5340 bool Function IsStopping() native
5341
5342 ; Checks to see if the quest is no longer enabled or running
5343 bool Function IsStopped() native
5344
5345 ; Resets the quest
5346 Function Reset() native
5347
5348 ; Flags this quest as "active" (tracked by the player)
5349 Function SetActive(bool abActive = true) native
5350
5351 ; Set the quest to the requested stage ID - returns true if stage exists and was set.
5352 ; This function is latent and will wait for the quest to start up before returning (if
    it needed to be started)
5353 bool Function SetCurrentStageID(int aiStageID) native
5354
5355 ; Sets the specified objective to completed or not
5356 Function SetObjectiveCompleted(int aiObjective, bool abCompleted = true) native
5357
5358 ; Sets the specified objective to displayed or hidden - if abForce is true, will display
    the objective even if it has already been displayed
5359 Function SetObjectiveDisplayed(int aiObjective, bool abDisplayed = true, bool abForce =
    false) native
5360
5361 ; Sets the specified objective to failed or not
5362 Function SetObjectiveFailed(int aiObjective, bool abFailed = true) native
5363
5364 ; Alias of SetCurrentStage - Set the quest to the requested stage
5365 ; This function is latent and will wait for the quest to start up before returning (if
    it needed to be started)
```

```
5366 bool Function SetStage(int aiStage)
5367     return SetCurrentStageID(aiStage)
5368 EndFunction
5369
5370 ; Starts the quest - returns whether the quest was able to be started or not
5371 ; This function is latent and will wait for the quest to start up before returning
5372 bool Function Start() native
5373
5374 ; Stops the quest
5375 Function Stop() native
5376
5377 ; Updates current instance's value for the given global
5378 bool Function UpdateCurrentInstanceGlobal( GlobalVariable aUpdateGlobal ) native
5379
5380 ; Story manager events - fired in parallel with the quest startup stage
5381
5382 Event OnStoryAddToPlayer(ObjectReference akOwner, ObjectReference akContainer, \
5383     Location akLocation, Form akItemBase, int aiAcquireType)
5384 EndEvent
5385
5386 Event OnStoryArrest(ObjectReference akArrestingGuard, ObjectReference akCriminal, \
5387     Location akLocation, int aiCrime)
5388 EndEvent
5389
5390 Event OnStoryAssaultActor(ObjectReference akVictim, ObjectReference akAttacker, \
5391     Location akLocation, int aiCrime)
5392 EndEvent
5393
5394 Event OnStoryBribeNPC(ObjectReference akActor)
5395 EndEvent
5396
5397 Event OnStoryCastMagic(ObjectReference akCastingActor, ObjectReference akSpellTarget, \
5398     Location akLocation, Form akSpell)
5399 EndEvent
5400
5401 Event OnStoryChangeLocation(ObjectReference akActor, Location akOldLocation, \
5402     Location akNewLocation)
5403 EndEvent
5404
5405 Event OnStoryCrimeGold(ObjectReference akVictim, ObjectReference akCriminal, \
5406     Form akFaction, int aiGoldAmount, int aiCrime)
5407 EndEvent
5408
5409 Event OnStoryCure(Form akInfection)
5410 EndEvent
5411
5412 Event OnStoryDialogue(Location akLocation, ObjectReference akActor1, ObjectReference
akActor2)
5413 EndEvent
5414
5415 Event OnStoryDiscoverDeadBody(ObjectReference akActor, ObjectReference akDeadActor, \
5416     Location akLocation)
5417 EndEvent
5418
5419 Event OnStoryEscapeJail(Location akLocation, Form akCrimeGroup)
5420 EndEvent
5421
5422 Event OnStoryActivateActor(Location akLocation, ObjectReference akActor)
5423 EndEvent
5424
5425 Event OnStoryFlatterNPC(ObjectReference akActor)
5426 EndEvent
5427
5428 Event OnStoryHello(Location akLocation, ObjectReference akActor1, ObjectReference
akActor2)
5429 EndEvent
5430
5431 Event OnStoryIncreaseLevel(int aiNewLevel)
5432 EndEvent
```

```
5433
5434 Event OnStoryIncreaseSkill(string asSkill)
5435 EndEvent
5436
5437 Event OnStoryInfection(ObjectReference akTransmittingActor, Form akInfection)
5438 EndEvent
5439
5440 Event OnStoryIntimidateNPC(ObjectReference akActor)
5441 EndEvent
5442
5443 Event OnStoryJail(ObjectReference akGuard, Form akCrimeGroup, Location akLocation, \
5444     int aiCrimeGold)
5445 EndEvent
5446
5447 Event OnStoryKillActor(ObjectReference akVictim, ObjectReference akKiller, \
5448     Location akLocation, int aiCrimeStatus, int aiRelationshipRank)
5449 EndEvent
5450
5451 Event OnStoryCraftItem(ObjectReference akBench, Location akLocation, Form akCreatedItem)
5452 EndEvent
5453
5454 Event OnStoryNewVoicePower(ObjectReference akActor, Form akVoicePower)
5455 EndEvent
5456
5457 Event OnStoryPickLock(ObjectReference akActor, ObjectReference akLock)
5458 EndEvent
5459
5460 Event OnStoryPayFine(ObjectReference akCriminal, ObjectReference akGuard, \
5461     Form akCrimeGroup, int aiCrimeGold)
5462 EndEvent
5463
5464 Event OnStoryPlayerGetsFavor(ObjectReference akActor)
5465 EndEvent
5466
5467 Event OnStoryRelationshipChange(ObjectReference akActor1, ObjectReference akActor2, \
5468     int aiOldRelationship, int aiNewRelationship)
5469 EndEvent
5470
5471 Event OnStoryRemoveFromPlayer(ObjectReference akOwner, ObjectReference akItem, \
5472     Location akLocation, Form akItemBase, int aiRemoveType)
5473 EndEvent
5474
5475 Event OnStoryScript(Keyword akKeyword, Location akLocation, ObjectReference akRef1, \
5476     ObjectReference akRef2, int aiValue1, int aiValue2)
5477 EndEvent
5478
5479 Event OnStoryServedTime(Location akLocation, Form akCrimeGroup, int aiCrimeGold, \
5480     int aiDaysJail)
5481 EndEvent
5482
5483 Event OnStoryTrespass(ObjectReference akVictim, ObjectReference akTrespasser, \
5484     Location akLocation, int aiCrime)
5485 EndEvent
5486
5487 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5488
5489 ; returns the quest with the specified editor id
5490 Quest Function GetQuest(string editorId) global native
5491
5492 ; returns the editor ID of the quest
5493 string Function GetID() native
5494
5495 ; returns the priority of the quest
5496 int Function GetPriority() native
5497
5498 ; returns the number of aliases associated with the quest
5499 int Function GetNumAliases() native
5500
5501 ; returns the specified alias associated with the quest
```

```

5502 Alias Function GetNthAlias(int index) native
5503
5504 ; returns the alias associated with the quest by name
5505 Alias Function GetAliasByName(string name) native
5506
5507 ; returns the alias by AlisID
5508 Alias Function GetAliasById(int aliasId) native
5509
5510 ; Returns all the aliases of this quest
5511 Alias[] Function GetAliases() native      :: Add a newline between files
5512 Scriptname Race extends Form Hidden
5513
5514 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5515 ; returns the number of spells for the race
5516 int Function GetSpellCount() native
5517
5518 ; returns the specified spell from the race
5519 Spell Function GetNthSpell(int n) native
5520
5521 ; returns whether the specified race flag is set
5522 bool Function IsRaceFlagSet(int n) native
5523
5524 ; sets the specified race flag
5525 Function SetRaceFlag(int n) native
5526
5527 ; clears the specified race flag
5528 Function ClearRaceFlag(int n) native
5529
5530 ; Returns the races default voice type
5531 VoiceType Function GetDefaultVoiceType(bool female) native
5532
5533 ; Sets the races default voice type
5534 Function SetDefaultVoiceType(bool female, VoiceType voice) native
5535
5536 ; Gets/sets the skin of the race
5537 Armor Function GetSkin() native
5538 Function SetSkin(Armor skin) native
5539
5540 ; Returns the number of playable races
5541 int Function GetNumPlayableRaces() native global
5542
5543 ; Returns the nth playable race
5544 Race Function GetNthPlayableRace(int n) native global
5545
5546 ; Returns a race by it's editorId name
5547 Race Function GetRace(string editorId) native global
5548
5549 ; race flags for previous functions
5550 int property kRace_Playable = 0x00000001 AutoReadOnly
5551 int property kRace_FaceGenHead = 0x00000002 AutoReadOnly
5552 int property kRace_Child = 0x00000004 AutoReadOnly
5553 int property kRace_TiltFrontBack = 0x00000008 AutoReadOnly
5554 int property kRace_TiltLeftRight = 0x00000010 AutoReadOnly
5555 int property kRace_NoShadow = 0x00000020 AutoReadOnly
5556 int property kRace_Swims = 0x00000040 AutoReadOnly
5557 int property kRace_Flies = 0x00000080 AutoReadOnly
5558 int property kRace_Walks = 0x00000100 AutoReadOnly
5559 int property kRace_Immobile = 0x00000200 AutoReadOnly
5560 int property kRace_NotPushable = 0x00000400 AutoReadOnly
5561 int property kRace_NoCombatInWater = 0x00000800 AutoReadOnly
5562 int property kRace_NoRotatingToHeadTrack = 0x00001000 AutoReadOnly
5563 int property kRace_UseHeadTrackAnim = 0x00008000 AutoReadOnly
5564 int property kRace_SpellsAlignWithMagicNode = 0x00010000 AutoReadOnly
5565 int property kRace_UseWorldRaycasts = 0x00020000 AutoReadOnly
5566 int property kRace_AllowRagdollCollision = 0x00040000 AutoReadOnly
5567 int property kRace_CantOpenDoors = 0x00100000 AutoReadOnly
5568 int property kRace_AllowPCDialogue = 0x00200000 AutoReadOnly
5569 int property kRace_NoKnockdowns = 0x00400000 AutoReadOnly
5570 int property kRace_AllowPickpocket = 0x00800000 AutoReadOnly

```

```
5571 int property kRace_AlwaysUseProxyController = 0x01000000 AutoReadOnly
5572 int property kRace_AllowMultipleMembraneShaders = 0x20000000 AutoReadOnly
5573 int property kRace_AvoidsRoads = 0x80000000 AutoReadOnly
5574
5575 bool Function IsPlayable()
5576     return IsRaceFlagSet(self.kRace_Playable)
5577 endFunction
5578
5579 Function MakePlayable()
5580     SetRaceFlag(self.kRace_Playable)
5581 endFunction
5582
5583 Function MakeUnplayable()
5584     ClearRaceFlag(self.kRace_Playable)
5585 endFunction
5586
5587 bool Function IsChildRace()
5588     return IsRaceFlagSet(self.kRace_Child)
5589 endFunction
5590
5591 Function MakeChildRace()
5592     SetRaceFlag(self.kRace_Child)
5593 endFunction
5594
5595 Function MakeNonChildRace()
5596     ClearRaceFlag(self.kRace_Child)
5597 endFunction
5598
5599 bool Function CanFly()
5600     return IsRaceFlagSet(self.kRace_Flies)
5601 endFunction
5602
5603 Function MakeCanFly()
5604     SetRaceFlag(self.kRace_Flies)
5605 endFunction
5606
5607 Function MakeNonFlying()
5608     ClearRaceFlag(self.kRace_Flies)
5609 endFunction
5610
5611 bool Function CanSwim()
5612     return IsRaceFlagSet(self.kRace_Swims)
5613 endFunction
5614
5615 Function MakeCanSwim()
5616     SetRaceFlag(self.kRace_Swims)
5617 endFunction
5618
5619 Function MakeNonSwimming()
5620     ClearRaceFlag(self.kRace_Swims)
5621 endFunction
5622
5623 bool Function CanWalk()
5624     return IsRaceFlagSet(self.kRace_Walks)
5625 endFunction
5626
5627 Function MakeCanWalk()
5628     SetRaceFlag(self.kRace_Walks)
5629 endFunction
5630
5631 Function MakeNonWalking()
5632     ClearRaceFlag(self.kRace_Walks)
5633 endFunction
5634
5635 bool Function IsImmobile()
5636     return IsRaceFlagSet(self.kRace_Immobile)
5637 endFunction
5638
5639 Function MakeImmobile()
```

```
5640         SetRaceFlag(self.kRace_Immobile)
5641     endFunction
5642
5643     Function MakeMobile()
5644         ClearRaceFlag(self.kRace_Immobile)
5645     endFunction
5646
5647     bool Function IsNotPushable()
5648         return IsRaceFlagSet(self.kRace_NotPushable)
5649     endFunction
5650
5651     Function MakeNotPushable()
5652         SetRaceFlag(self.kRace_NotPushable)
5653     endFunction
5654
5655     Function MakePushable()
5656         ClearRaceFlag(self.kRace_NotPushable)
5657     endFunction
5658
5659     bool Function NoKnockdowns()
5660         return IsRaceFlagSet(self.kRace_AllowPickpocket)
5661     endFunction
5662
5663     Function MakeNoKnockdowns()
5664         SetRaceFlag(self.kRace_AllowPickpocket)
5665     endFunction
5666
5667     Function ClearNoKnockdowns()
5668         ClearRaceFlag(self.kRace_AllowPickpocket)
5669     endFunction
5670
5671     bool Function NoCombatInWater()
5672         return IsRaceFlagSet(self.kRace_NoCombatInWater)
5673     endFunction
5674
5675     Function SetNoCombatInWater()
5676         SetRaceFlag(self.kRace_NoCombatInWater)
5677     endFunction
5678
5679     Function ClearNoCombatInWater()
5680         ClearRaceFlag(self.kRace_NoCombatInWater)
5681     endFunction
5682
5683     bool Function AvoidsRoads()
5684         return IsRaceFlagSet(self.kRace_AvoidsRoads)
5685     endFunction
5686
5687     Function SetAvoidsRoads()
5688         SetRaceFlag(self.kRace_AvoidsRoads)
5689     endFunction
5690
5691     Function ClearAvoidsRoads()
5692         ClearRaceFlag(self.kRace_AvoidsRoads)
5693     endFunction
5694
5695     bool Function AllowPickpocket()
5696         return IsRaceFlagSet(self.kRace_AllowPickpocket)
5697     endFunction
5698
5699     Function SetAllowPickpocket()
5700         SetRaceFlag(self.kRace_AllowPickpocket)
5701     endFunction
5702
5703     Function ClearAllowPickpocket()
5704         ClearRaceFlag(self.kRace_AllowPickpocket)
5705     endFunction
5706
5707     bool Function AllowPCDialogue()
5708         return IsRaceFlagSet(self.kRace_AllowPCDialogue)
```



```

5709     endFunction
5710
5711     Function SetAllowPCDialogue()
5712         SetRaceFlag(self.kRace_AllowPCDialogue)
5713     endFunction
5714
5715     Function ClearAllowPCDialogue()
5716         ClearRaceFlag(self.kRace_AllowPCDialogue)
5717     endFunction
5718
5719     bool Function CantOpenDoors()
5720         return IsRaceFlagSet(self.kRace_CantOpenDoors)
5721     endFunction
5722
5723     Function SetCantOpenDoors()
5724         SetRaceFlag(self.kRace_CantOpenDoors)
5725     endFunction
5726
5727     Function ClearCantOpenDoors()
5728         ClearRaceFlag(self.kRace_CantOpenDoors)
5729     endFunction
5730
5731     bool Function NoShadow()
5732         return IsRaceFlagSet(self.kRace_NoShadow)
5733     endFunction
5734
5735     Function SetNoShadow()
5736         SetRaceFlag(self.kRace_NoShadow)
5737     endFunction
5738
5739     Function ClearNoShadow()
5740         ClearRaceFlag(self.kRace_NoShadow)
5741     endFunction
5742
5743     :: Add a newline between files
5744     Scriptname Scroll extends Form Hidden
5745
5746     ; Cast this scroll from an ObjectReference, optionally toward another.
5747     Function Cast(ObjectReference akSource, ObjectReference akTarget=NONE) native
5748
5749     ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5750     ; return the casting time
5751     float Function GetCastTime() native
5752
5753     ; return the perk associated with the spell
5754     Perk Function GetPerk() native
5755
5756     ; return the number of the effects
5757     int Function GetNumEffects() native
5758
5759     ; return the magnitude of the specified effect
5760     float Function GetNthEffectMagnitude(int index) native
5761
5762     ; return the area of the specified effect
5763     int Function GetNthEffectArea(int index) native
5764
5765     ; return the duration of the specified effect
5766     int Function GetNthEffectDuration(int index) native
5767
5768     ; return the magic effect of the specified effect
5769     MagicEffect Function GetNthEffectMagicEffect(int index) native
5770
5771     ; return the index of the costliest effect
5772     int Function GetCostliestEffectIndex() native
5773
5774     ; sets the magnitude of the specified effect
5775     Function SetNthEffectMagnitude(int index, float value) native
5776
5777     ; sets the area of the specified effect

```

```

5778 Function SetNthEffectArea(int index, int value) native
5779
5780 ; sets the duration of the specified effect
5781 Function SetNthEffectDuration(int index, int value) native
5782
5783 ; Returns the particular equipslot type
5784 EquipSlot Function GetEquipType() native
5785 Function SetEquipType(EquipSlot type) native
5786
5787 ; Returns all the magnitudes of this object in order
5788 float[] Function GetEffectMagnitudes() native
5789
5790 ; Returns all the areas of this object in order
5791 int[] Function GetEffectAreas() native
5792
5793 ; Returns all the durations of this object in order
5794 int[] Function GetEffectDurations() native
5795
5796 ; Returns all the magic effects of this object in order
5797 MagicEffect[] Function GetMagicEffects() native    :: Add a newline between files
5798 Scriptname Shout extends Form Hidden
5799
5800 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5801 WordOfPower Function GetNthWordOfPower(int n) native
5802 Spell Function GetNthSpell(int n) native
5803 float Function GetNthRecoveryTime(int n) native
5804
5805 Function SetNthWordOfPower(int n, WordOfPower aWoop) native
5806 Function SetNthSpell(int n, Spell aSpell) native
5807 Function SetNthRecoveryTime(int n, float time) native    :: Add a newline between files
5808 Scriptname SKSE Hidden
5809 ; General SKSE-specific information
5810
5811 ; get the major version of SKSE
5812 int Function GetVersion() global native
5813 ; get the minor version of SKSE
5814 int Function GetVersionMinor() global native
5815 ; get the beta version of SKSE
5816 int Function GetVersionBeta() global native
5817 ; get the release index of SKSE. This number is incremented every time
5818 ; SKSE is released outside of the development team
5819 int Function GetVersionRelease() global native
5820 ; get the release index of this script file.
5821 ; Can be used to detect a script/runtime version mismatch
5822 int Function GetScriptVersionRelease() global
5823     return 72
5824 endFunction
5825
5826 ; get a plugins version number, -1 if the plugin is not loaded
5827 int Function GetPluginVersion(string name) global native    :: Add a newline between files
5828 Scriptname SoulGem extends MiscObject Hidden
5829
5830 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5831
5832 int Function GetSoulSize() native
5833 int Function GetGemSize() native    :: Add a newline between files
5834 Scriptname Sound extends Form Hidden
5835 import ObjectReference
5836
5837 ; Play this sound base object from the specified source
5838 int Function Play(ObjectReference akSource) native
5839
5840 ; Play this sound from the specified source, and wait for it to finish
5841 bool Function PlayAndWait(ObjectReference akSource) native
5842
5843 ; Stops a given playback instance of a sound
5844 Function StopInstance(int aiPlaybackInstance) native global
5845
5846 ; Set the volume of a given playback instance of a sound. Clamped between 0 and 1.

```

```

5847 Function SetInstanceVolume(int aiPlaybackInstance, float afVolume) native global
5848
5849
5850
5851 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5852 SoundDescriptor Function GetDescriptor() native      :: Add a newline between files
5853 Scriptname SoundDescriptor extends Form Hidden
5854
5855 float Function GetDecibelAttenuation() native
5856 Function SetDecibelAttenuation(float dbAttenuation) native
5857
5858 int Function GetDecibelVariance() native
5859 Function SetDecibelVariance(int dbVariance) native
5860
5861 int Function GetFrequencyVariance() native
5862 Function SetFrequencyVariance(int frequencyVariance) native
5863
5864 int Function GetFrequencyShift() native
5865 Function SetFrequencyShift(int frequencyShift) native      :: Add a newline between files
5866 Scriptname SpawnerTask Hidden
5867
5868 ; SpawnerTask allows to spawn and position an arbitrary number references in game world.
5869 ; It's effectively a batch combination of PlaceAtMe and SetPosition/MoveTo that smoothly
    executes in a single frame.
5870 ;
5871 ; Example:
5872 ;
5873 ;     ObjectReference player    = ...
5874 ;     Form             chair    = ...
5875 ;     float[]          offset   = new float[3]
5876 ;     float[]          rotation = new float[3]
5877 ;
5878 ;     ; Allocate new task
5879 ;     int taskId = SpawnerTask.Create()
5880 ;
5881 ;     ; No rotation
5882 ;     rotation[0] = 0
5883 ;     rotation[1] = 0
5884 ;     rotation[2] = 0
5885 ;
5886 ;     ; Spawn 100 chairs in a grid above the player
5887 ;     int i = 0
5888 ;     while i < 100
5889 ;         offset[0] = -250 + (i / 10) * 50
5890 ;         offset[1] = -250 + (i % 10) * 50
5891 ;         offset[2] = 200
5892 ;
5893 ;         SpawnerTask.AddSpawn(taskId, chair, player, offset, rotation)
5894 ;         i += 1
5895 ;     endwhile
5896 ;
5897 ;     ; Run the task and return all placed references in an array
5898 ;     ObjectReference[] spawned = SpawnerTask.Run(taskId)
5899
5900
5901 ; Creates a new SpawnerTask and returns a handle, which is an identifier for the created
    task.
5902 ; The task handle is valid until the task has been run or canceled, or until the calling
    stack has exited.
5903 ;     (Function type: non-delayed)
5904 ;
5905 int Function Create() global native
5906
5907 ; Adds a spawn to the task identified by the given handle.
5908 ; Running the task places a new instance of formToPlace at target reference with given
    rotation and position offset. Parameters are analogously defined to PlaceAtMe.
5909 ; Multiple spawns can be added to the same task to be executed in a batch (which is the
    purpose).
5910 ;     (Function type: non-delayed)

```

```

5911 ;
5912 Function AddSpawn(int handle, Form formToPlace, ObjectReference target, float[]
positionOffset, float[] rotation, int count = 1, bool bForcePersist = false, bool
bInitiallyDisabled = false) global native
5913
5914 ; Runs the task and returns the spawned references in an array. May return arrays with a
size larger than 128.
5915 ; The resources allocated to the task are freed in the process, so the same task handle
cannot be run twice.
5916 ; (Function type: latent)
5917 ;
5918 ObjectReference[] Function Run(int handle) global native
5919
5920 ; Cancels a task before running it and frees its allocated resources.
5921 ; Tasks cannot be canceled once they have been started with Run, and vice versa.
5922 ;
5923 Function Cancel(int handle) global native
5924 :: Add a newline between files
5925 Scriptname Spell extends Form Hidden
5926
5927 ; Cast this spell from an ObjectReference, optionally toward another.
5928 Function Cast(ObjectReference akSource, ObjectReference akTarget=NONE) native
5929
5930 ; Cast this spell from an ObjectReference, optionally toward another, and blame it on a
particular actor.
5931 Function RemoteCast(ObjectReference akSource, Actor akBlameActor, ObjectReference
akTarget=NONE) native
5932
5933 ; Is this spell classified as hostile?
5934 bool Function IsHostile() native
5935
5936 ; Preload the art for this spell. Useful for spells you equip & unequip on the player.
5937 ; Warning: Misuse of this function can lead to erroneous behavior as well as excessive
5938 ; memory consumption. It's best to avoid using this. This function will likely be
5939 ; deprecated in the future.
5940 Function Preload() native
5941
5942 ; Unload the art for this spell. Call this only if you've previously called Preload.
5943 ; Warning: Misuse of this function can lead to erroneous behavior including spell art
5944 ; being unloaded while in use, and excessive memory consumption. It's best to avoid
using this.
5945 ; This function will likely be deprecated in the future.
5946 Function Unload() native
5947
5948
5949 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
5950 ; return the casting time
5951 float Function GetCastTime() native
5952
5953 ; return the perk associated with the spell
5954 Perk Function GetPerk() native
5955
5956 ; return the number of the effects
5957 int Function GetNumEffects() native
5958
5959 ; return the magnitude of the specified effect
5960 float Function GetNthEffectMagnitude(int index) native
5961
5962 ; return the area of the specified effect
5963 int Function GetNthEffectArea(int index) native
5964
5965 ; return the duration of the specified effect
5966 int Function GetNthEffectDuration(int index) native
5967
5968 ; return the magic effect of the specified effect
5969 MagicEffect Function GetNthEffectMagicEffect(int index) native
5970
5971 ; return the index of the costliest effect
5972 int Function GetCostliestEffectIndex() native

```

```

5973
5974 ; return the base magicka cost of the spell
5975 int Function GetMagickaCost() native
5976
5977 ; return the effective magicka cost of the spell for given caster
5978 int Function GetEffectiveMagickaCost(Actor caster) native
5979
5980 ; sets the magnitude of the specified effect
5981 Function SetNthEffectMagnitude(int index, float value) native
5982
5983 ; sets the area of the specified effect
5984 Function SetNthEffectArea(int index, int value) native
5985
5986 ; sets the duration of the specified effect
5987 Function SetNthEffectDuration(int index, int value) native
5988
5989 ; Returns the particular equipslot type
5990 EquipSlot Function GetEquipType() native
5991 Function SetEquipType(EquipSlot type) native
5992
5993 ; Returns all the magnitudes of this object in order
5994 float[] Function GetEffectMagnitudes() native
5995
5996 ; Returns all the areas of this object in order
5997 int[] Function GetEffectAreas() native
5998
5999 ; Returns all the durations of this object in order
6000 int[] Function GetEffectDurations() native
6001
6002 ; Returns all the magic effects of this object in order
6003 MagicEffect[] Function GetMagicEffects() native    :: Add a newline between files
6004 Scriptname StringUtil Hidden
6005
6006 ; Note about the internal Skyrim implementation of the string classes used for scripting:
6007 ; the strings are case-insensitive. Each BSFixedString is managed in a cache and reused
6008 ; everywhere it is needed. This means that strings like "O" and "o" are technically
6009 ; equivalent;
6010 ; Which string is used depends greatly on which version is found first. We are
6011 ; investigating
6012 ; how to manage this, but for the time being be aware that the distinction between
6013 ; uppercase
6014 ; and lowercase may not exist. It also means that functions below returning an integer
6015 ; for the character may not correspond exactly. Also GetNthChar("Hello Skyrim!", 4) will
6016 ; return a string with either "O" or "o" depending on which might be registered first.
6017 ; All
6018 ; my tests so far have it return the uppercase, eventhough in the string it is lowercase.
6019 ; We may solve this problem by switching back to returning an integer rather than a
6020 ; string
6021 ; for GetNthChar, but this will still have problems.
6022
6023 ; return the length of the string
6024 int Function GetLength(string s) global native
6025
6026 ; returns a single character string with the character at index
6027 string Function GetNthChar(string s, int index) global native
6028
6029 ; Functions to work on Chars
6030 ; returns information about a specific character
6031 ; assumes a single character string. If a multicharacter string is passed
6032 ; the information about the first character is returned
6033 bool Function IsLetter(string c) global native
6034 bool Function IsDigit(string c) global native
6035 bool Function IsPunctuation(string c) global native
6036 bool Function IsPrintable(string c) global native
6037
6038 ; returns the index of the first character of toFind inside string s
6039 ; returns -1 if toFind is not part of the string or if startIndex is invalid
6040 int Function Find(string s, string toFind, int startIndex = 0) global native
6041

```

```
6037 ; returns a substring of the specified string starting at startIndex and going for len
6038 characters
6038 ; or until the end of the string. Default len of 0 means for the entire string
6039 string Function Substring(string s, int startIndex, int len = 0) global native
6040
6041 ; returns the numeric value of the first character as an int
6042 int Function AsOrd(string c) global native
6043
6044 ; returns a single character string interpreting c as a character
6045 string Function AsChar(int c) global native
6046
6047 ; returns array of strings separated by the specified delimiter
6048 string[] Function Split(string s, string delim) global native :: Add a newline between
6048 files
6049 Scriptname TextureSet extends Form Hidden
6050
6051
6052 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
6053
6054 ; Returns the number of texture paths
6055 int Function GetNumTexturePaths() native
6056
6057 ; Returns the path of the texture
6058 string Function GetNthTexturePath(int n) native
6059
6060 ; Sets the path of the texture
6061 Function SetNthTexturePath(int n, string texturePath) native :: Add a newline between
6061 files
6062 Scriptname TreeObject extends Form Hidden
6063
6064 SoundDescriptor Function GetHarvestSound() native
6065 Function SetHarvestSound(SoundDescriptor akSoundDescriptor) native
6066
6067 Form Function GetIngredient() native
6068 Function SetIngredient(Form akIngredient) native :: Add a newline between files
6069 Scriptname UI Hidden
6070
6071 ; For functions that require a menuName, potential values are
6072 ; "InventoryMenu"
6073 ; "Console"
6074 ; "Dialogue Menu"
6075 ; "HUD Menu"
6076 ; "Main Menu"
6077 ; "MessageBoxMenu"
6078 ; "Cursor Menu"
6079 ; "Fader Menu"
6080 ; "MagicMenu"
6081 ; "Top Menu"
6082 ; "Overlay Menu"
6083 ; "Overlay Interaction Menu"
6084 ; "Loading Menu"
6085 ; "TweenMenu"
6086 ; "BarterMenu"
6087 ; "GiftMenu"
6088 ; "Debug Text Menu"
6089 ; "MapMenu"
6090 ; "Lockpicking Menu"
6091 ; "Quantity Menu"
6092 ; "StatsMenu"
6093 ; "ContainerMenu"
6094 ; "Sleep/Wait Menu"
6095 ; "LevelUp Menu"
6096 ; "Journal Menu"
6097 ; "Book Menu"
6098 ; "FavoritesMenu"
6099 ; "RaceSex Menu"
6100 ; "Crafting Menu"
6101 ; "Training Menu"
6102 ; "Mist Menu"
```

```

6103 ; "Tutorial Menu"
6104 ; "Credits Menu"
6105 ; "TitleSequence Menu"
6106 ; "Console Native UI Menu"
6107 ; "Kinect Menu"
6108 ;
6109 ; The target parameter requires one the following prefixes:
6110 ; _global      , for the default namespace;
6111 ; _root        , for the movie root.
6112
6113
6114 ; Returns if the menu is currently open.
6115 bool Function IsMenuOpen(string menuName) global native
6116
6117
6118 ; Sets bool/number/string value at target location.
6119 ; Target value must already exist.
6120 ;
6121 ; Examples:
6122 ;     UI.SetBool("InventoryMenu", "_root.Menu_mc._visible", false)
6123 ;     UI.SetString("FavoritesMenu", "_root.Menu_mc.panel.message.text", "My Text")
6124 ;
6125 Function SetBool(string menuName, string target, bool value) global native
6126 Function SetInt(string menuName, string target, int value) global native
6127 Function SetFloat(string menuName, string target, float value) global native
6128 Function SetString(string menuName, string target, string value) global native
6129 Function SetNumber(string menuName, string target, float value) global ; DEPRECIATED
6130     SetFloat(menuName, target, value)
6131 EndFunction
6132
6133 ; Gets bool/number/string from target location, or false/0/none if the value doesn't
6134 exist.
6135 ;
6136 ; Examples:
6137 ;     bool    visible = UI.GetBool("Inventory Menu", "_root.Menu_mc._visible")
6138 ;     float    height  = UI.GetNumber("Magic Menu", "_root.Menu_mc._height")
6139 ;
6140 bool    Function GetBool(string menuName, string target) global native
6141 int      Function GetInt(string menuName, string target) global native
6142 float    Function GetFloat(string menuName, string target) global native
6143 string    Function GetString(string menuName, string target) global native
6144 float    Function GetNumber(string menuName, string target) global ; DEPRECIATED
6145     return GetFloat(menuName, target)
6146 EndFunction
6147
6148 ; Invokes the ActionScript function at given target location.
6149 ;
6150 ; Examples:
6151 ;     UI.InvokeString("InventoryMenu", "_global.skse.Log", "Printed to logfile")
6152 ;     UI.InvokeStringA("InventoryMenu", "_global.myFunction", myArray)
6153 ;
6154 Function Invoke(string menuName, string target) global
6155     InvokeBool(menuName, target, false)
6156 EndFunction
6157
6158 Function InvokeBool(string menuName, string target, bool arg) global native
6159 Function InvokeInt(string menuName, string target, int arg) global native
6160 Function InvokeFloat(string menuName, string target, float arg) global native
6161 Function InvokeString(string menuName, string target, string arg) global native
6162 Function InvokeNumber(string menuName, string target, float arg) global ; DEPRECIATED
6163     InvokeFloat(menuName, target, arg)
6164 EndFunction
6165
6166 Function InvokeBoolA(string menuName, string target, bool[] args) global native
6167 Function InvokeIntA(string menuName, string target, int[] args) global native
6168 Function InvokeFloatA(string menuName, string target, float[] args) global native
6169 Function InvokeStringA(string menuName, string target, string[] args) global native
6170 Function InvokeNumberA(string menuName, string target, float[] args) global ; DEPRECIATED

```

```

6171     InvokeFloatA(menuName, target, args)
6172 EndFunction
6173
6174 ; Sends Form data to Scaleform as a Flash object, FormLists included.
6175 Function InvokeForm(string menuName, string target, Form arg) global native
6176
6177 ; returns if scaleform is in 'text input' mode
6178 ; this is useful for ignoring keys that should get swallowed by an editable text box
6179 bool Function IsTextInputEnabled() global native
6180
6181 ; open a custom menu named "CustomMenu" by loading the given swf from the interface
    folder
6182 ; (filename without extension)
6183 ; there can only be a single custom menu open at the same time
6184 Function OpenCustomMenu(string swfPath, int flags = 0) global native
6185
6186 ; close the custom menu if it's currently open.
6187 Function CloseCustomMenu() global native    :: Add a newline between files
6188 Scriptname UICallback Hidden
6189
6190 ; UICallback allows passing arguments of different types to UI functions, unlike
    UI.Invoke*
6191 ;
6192 ; Example:
6193 ;     int handle = UICallback.Create("InventoryMenu", "_global.MyClass.initData")
6194 ;     if (handle)
6195 ;         UICallback.PushBool(handle, true)
6196 ;         UICallback.PushInt(handle, 1000)
6197 ;         UICallback.PushString(handle, "Hello")
6198 ;         UICallback.PushFloat(handle, 1.234)
6199 ;         UIDelegate.Send(handle)
6200 ;     endif
6201 ;
6202 ; Any UICallback allocated by Create must be released later.
6203 ; That happens automatically when passing it to send.
6204 ; Otherwise the handle must be manually released by passing it to Release.
6205 ;
6206 ; Internally, UICallback objects only persist for the duration of the current
6207 ; game session. They are also cleared after each reload.
6208 ;
6209 ; This means that in very rare cases, the execution sequence of several operations
6210 ; on one UICallback might get interrupted, the handle turns invalid and the final Send
6211 ; will fail. If its necessary to detect this, check the return value of Send.
6212
6213 ; Creates a new UICallback and returns the handle.
6214 int Function Create(string menuName, string target) global native
6215
6216 ; Invokes the UICallback and releases it.
6217 ; Returns true, if it was executed, false if an error happened.
6218 bool Function Send(int handle) global native
6219
6220 ; Releases the UICallback without sending it.
6221 Function Release(int handle) global native
6222
6223 ; Push single parameter. Maximum number of parameters per callback is 128.
6224 Function PushBool(int handle, bool value) global native
6225 Function PushInt(int handle, int value) global native
6226 Function PushFloat(int handle, float value) global native
6227 Function PushString(int handle, string value) global native
6228
6229 ; Push parameters from array. Maximum number of parameters per callback is 128.
6230 Function PushBoolA(int handle, bool[] args) global native
6231 Function PushIntA(int handle, int[] args) global native
6232 Function PushFloatA(int handle, float[] args) global native
6233 Function PushStringA(int handle, string[] args) global native    :: Add a newline between
    files
6234 Scriptname Utility Hidden
6235
6236 ; Converts a float game time (in terms of game days passed) to a string detailing the

```



```

date
6237 ; and time it represents in "MM/DD/YYYY HH:MM" format. A 24-hour clock is used, and the
function
6238 ; is latent (due to issues in the current architecture with returning strings from code)
6239 string Function GameTimeToString(float afGameTime) native global
6240
6241 ; Obtains the current game time in terms of game days passed (same as the global
variable)
6242 float Function GetCurrentGameTime() native global
6243
6244 ; Obtains the number of seconds since the application started (the same timer that
WaitMenuMode uses)
6245 ; Does not take into account menu-mode, or VM frozen time
6246 ; Most useful for determining how long something took to run
6247 float Function GetCurrentRealTime() native global
6248
6249 ; Returns whether the game is currently in menu mode or not
6250 bool Function IsInMenuMode() native global
6251
6252 ; Generates a random integer between aiMin and aiMax (inclusive)
6253 int Function RandomInt(int aiMin = 0, int aiMax = 100) native global
6254
6255 ; Generates a random floating point number between afMin and afMax (inclusive)
6256 float Function RandomFloat(float afMin = 0.0, float afMax = 1.0) native global
6257
6258 ; Set the given INI by type
6259 function SetINIFloat(string ini, float value) native global
6260 function SetINIInt(string ini, int value) native global
6261 function SetINIBool(string ini, bool value) native global
6262 function SetINIStr(string ini, string value) native global
6263
6264 ; Waits for the specified amount of time (latent). Timer will not run during menu mode
6265 Function Wait(float afSeconds) native global
6266
6267 ; Waits for the specified amount of game time (latent)
6268 Function WaitGameTime(float afHours) native global
6269
6270 ; Waits for the specified amount of time (latent) - Timer WILL run during menu mode
6271 Function WaitMenuMode(float afSeconds) native global
6272
6273 ; Frame rate capture functions only available in beta version
6274
6275 ; Gets you a string describing the frame rate for a certain number of frames
6276 ; (String will be no longer than 1K characters long, separated by commas)
6277 string Function CaptureFrameRate(int numFrames) native global
6278
6279 ; Starts or ends a frame rate capture -- then you can get the min or max since
6280 ; frame capture started at any time
6281 Function StartFrameRateCapture() native global
6282 Function EndFrameRateCapture() native global
6283 float Function GetAverageFrameRate() native global
6284 float Function GetMinFrameRate() native global
6285 float Function GetMaxFrameRate() native global
6286
6287 ; Memory tracking functions - only available if memory tracking is turned on
6288 int Function GetCurrentMemory() native global ; Must be called first, it sets up the
memory stats used by the other functions
6289 int Function GetBudgetCount() native global
6290 int Function GetCurrentBudget(int aiBudgetNumber) native global
6291 bool Function OverBudget(int aiBudgetNumber) native global
6292 string Function GetBudgetName(int aiBudgetNumber) native global
6293
6294 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
6295
6296 float Function GetINIFloat(string ini) global native
6297 int Function GetINIInt(string ini) global native
6298 bool Function GetINIBool(string ini) global native
6299 string Function GetINIStr(string ini) global native
6300

```

```
6301
6302 ; Size is treated as unsigned, negative numbers will result
6303 ; extremely large positive numbers, USE WITH CARE
6304 float[] Function CreateFloatArray(int size, float fill = 0.0) global native
6305 int[] Function CreateIntArray(int size, int fill = 0) global native
6306 bool[] Function CreateBoolArray(int size, bool fill = false) global native
6307 string[] Function CreateStringArray(int size, string fill = "") global native
6308 Form[] Function CreateFormArray(int size, Form fill = None) global native
6309 Alias[] Function CreateAliasArray(int size, Alias fill = None) global native
6310
6311 float[] Function ResizeFloatArray(float[] source, int size, float fill = 0.0) global
native
6312 int[] Function ResizeIntArray(int[] source, int size, int fill = 0) global native
6313 bool[] Function ResizeBoolArray(bool[] source, int size, bool fill = false) global native
6314 string[] Function ResizeStringArray(string[] source, int size, string fill = "") global
native
6315 Form[] Function ResizeFormArray(Form[] source, int size, Form fill = None) global native
6316 Alias[] Function ResizeAliasArray(Alias[] source, int size, Alias fill = None) global
native    :: Add a newline between files
6317 Scriptname Weapon extends Form Hidden
6318
6319 ; Fire this weapon base object from the specified source
6320 Function Fire(ObjectReference akSource, Ammo akAmmo = None) native
6321
6322
6323 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
6324
6325 int Function GetBaseDamage() native
6326 Function SetBaseDamage(int damage) native
6327
6328 int Function GetCritDamage() native
6329 Function SetCritDamage(int damage) native
6330
6331 float Function GetReach() native
6332 Function SetReach(float reach) native
6333
6334 float Function GetMinRange() native
6335 Function SetMinRange(float minRange) native
6336
6337 float Function GetMaxRange() native
6338 Function SetMaxRange(float maxRange) native
6339
6340 float Function GetSpeed() native
6341 Function SetSpeed(float speed) native
6342
6343 float Function GetStagger() native
6344 Function SetStagger(float stagger) native
6345
6346 int Function GetWeaponType() native
6347 Function SetWeaponType(int type) native
6348
6349 ; works on the path to the nif file representing the in-game model of the weapon
6350 string Function GetModelPath() native
6351 Function SetModelPath(string path) native
6352
6353 ; works on the path to the nif file representing the icon for the weapon in the inventory
6354 string Function GetIconPath() native
6355 Function SetIconPath(string path) native
6356
6357 ; works on the path to the file representing the message icon for the weapon
6358 string Function GetMessageIconPath() native
6359 Function SetMessageIconPath(string path) native
6360
6361 ; works on the enchantment associated with the weapon
6362 Enchantment Function GetEnchantment() native
6363 Function SetEnchantment(Enchantment e) native
6364
6365 ; works on the enchantment value of the associated weapon
6366 int Function GetEnchantmentValue() native
```

```
6367 Function SetEnchantmentValue(int value) native
6368
6369 ; works on the weapon model when equipped of the associated weapon
6370 Static Function GetEquippedModel() native
6371 Function SetEquippedModel(Static model) native
6372
6373 ; Returns the particular equipslot type
6374 EquipSlot Function GetEquipType() native
6375 Function SetEquipType(EquipSlot type) native
6376
6377 string Function GetSkill() native
6378 Function SetSkill(string skill) native
6379
6380 ; DamageResist
6381 ; ElectricResist
6382 ; FireResist
6383 ; FrostResist
6384 ; MagicResist
6385 ; PoisonResist
6386 string Function GetResist() native
6387 Function SetResist(string resist) native
6388
6389 ; works on the spell that applies when critting
6390 Spell Function GetCritEffect() native
6391 Function SetCritEffect(Spell ce) native
6392
6393 ; Gets, sets or unsets whether the the crit effect should only occur on death
6394 bool Function GetCritEffectOnDeath() native
6395 Function SetCritEffectOnDeath(bool ceod) native
6396
6397 ; Gets/sets the weapons crit multiplier
6398 float Function GetCritMultiplier() native
6399 Function SetCritMultiplier(float crit) native
6400
6401 ; returns the weapon template of this weapon
6402 Weapon Function GetTemplate() native
6403
6404 bool Function IsBattleaxe()
6405     return HasKeywordString("WeapTypeBattleaxe")
6406 endFunction
6407
6408 bool Function IsBow()
6409     return HasKeywordString("WeapTypeBow")
6410 endFunction
6411
6412 bool Function IsDagger()
6413     return HasKeywordString("WeapTypeDagger")
6414 endFunction
6415
6416 bool Function IsGreatsword()
6417     return HasKeywordString("WeapTypeGreatsword")
6418 endFunction
6419
6420 bool Function IsMace()
6421     return HasKeywordString("WeapTypeMace")
6422 endFunction
6423
6424 bool Function IsStaff()
6425     return HasKeywordString("WeapTypeStaff")
6426 endFunction
6427
6428 bool Function IsSword()
6429     return HasKeywordString("WeapTypeSword")
6430 endFunction
6431
6432 bool Function IsWarhammer()
6433     return HasKeywordString("WeapTypeWarhammer")
6434 endFunction
6435
```

```

6436 bool Function IsWarAxe()
6437     return HasKeywordString("WeapTypeWarAxe")
6438 endFunction
6439 :: Add a newline between files
6440 Scriptname Weather extends Form Hidden
6441
6442 ; Tells the sky to release its overriding weather.
6443 function ReleaseOverride() native global
6444
6445 ; Forces the active weather on the sky to be this weather.
6446 function ForceActive( bool abOverride=false ) native
6447
6448 ; Sets the active weather on the sky to be this weather.
6449 function SetActive( bool abOverride=false, bool abAccelerate=false ) native
6450
6451 ; Finds a weather from the current region/climate whose classification matches the given
one.
6452 ; 0 - Pleasant
6453 ; 1 - Cloudy
6454 ; 2 - Rainy
6455 ; 3 - Snow
6456 Weather function FindWeather( int auType ) native global
6457
6458 ; Gets this weather's classification
6459 ; -1 - No classification
6460 ; 0 - Pleasant
6461 ; 1 - Cloudy
6462 ; 2 - Rainy
6463 ; 3 - Snow
6464 int function GetClassification() native
6465
6466 ; Gets the sky's current weather
6467 Weather function GetCurrentWeather() native global
6468
6469 ; Gets the sky's outgoing weather
6470 Weather function GetOutgoingWeather() native global
6471
6472 ; Gets the transition percentage of the current weather
6473 float function GetCurrentWeatherTransition() native global
6474
6475 ; Gets the sky's current mode
6476 ; 0 - No sky (SM_NONE)
6477 ; 1 - Interior (SM_INTERIOR)
6478 ; 2 - Skydome only (SM_SKYDOME_ONLY)
6479 ; 3 - Full sky (SM_FULL)
6480 int function GetSkyMode() native global
6481
6482 ; SKSE64 additions built 2024-01-17 20:01:40.731000 UTC
6483
6484 ; Returns the sun glare percentage
6485 float Function GetSunGlare() native
6486
6487 ; Returns the sun damage percentage
6488 float Function GetSunDamage() native
6489
6490 ; Returns the wind direction in degrees (0-360)
6491 float Function GetWindDirection() native
6492
6493 ; Returns the wind direction range in degrees (0-180)
6494 float Function GetWindDirectionRange() native
6495
6496 ; 0 - Near
6497 ; 1 - Far
6498 ; 2 - Power
6499 ; 3 - Max
6500 float Function GetFogDistance(bool day, int type) native
6501 :: Add a newline between files
6502 Scriptname WornObject Hidden
6503

```

```

6504 ; These functions operate directly on
6505 ; worn items within the inventory
6506 ; Valid Hand Slot:
6507 ; 0 - Left
6508 ; 1 - Right
6509 ; Valid Slot Masks:
6510 ; See Armor.psc
6511 ; Use zero when using hand slot
6512
6513 ; Tempering
6514 float Function GetItemHealthPercent(Actor akActor, int handSlot, int slotMask) global
native
6515 Function SetItemHealthPercent(Actor akActor, int handSlot, int slotMask, float health)
global native
6516
6517 ; Charges
6518 ; Only works on items that have user-enchants
6519 Function SetItemMaxCharge(Actor akActor, int handSlot, int slotMask, float maxCharge)
global native
6520
6521 ; Works on any enchanted item
6522 float Function GetItemMaxCharge(Actor akActor, int handSlot, int slotMask) global native
6523
6524 float Function GetItemCharge(Actor akActor, int handSlot, int slotMask) global native
6525
6526 ; Use LeftItemCharge/RightItemCharge ActorValues instead
6527 ;Function SetItemCharge(Actor akActor, int handSlot, int slotMask, float charge) global
native
6528
6529 ; Returns the name of this reference
6530 ; this is the name that is displayed
6531 string Function GetDisplayName(Actor akActor, int handSlot, int slotMask) global native
6532
6533 ; Sets a reference's display name
6534 ; returns false if force is false and the reference
6535 ; is held by an alias using 'Stored Text' or 'Uses Stored Text'
6536 ; Text Replacement does not use this name and may be lost if forced
6537 bool Function SetDisplayName(Actor akActor, int handSlot, int slotMask, string name,
bool force = false) global native
6538
6539 ; Returns the player-made enchantment if there is one
6540 Enchantment Function GetEnchantment(Actor akActor, int handSlot, int slotMask) global
native
6541
6542 ; Changes an item's player-made enchantment to something else
6543 ; None enchantment will remove the existing enchantment
6544 ; does not delete the custom enchantment, only removes it
6545 Function SetEnchantment(Actor akActor, int handSlot, int slotMask, Enchantment source,
float maxCharge) global native
6546
6547 ; Creates a new enchantment on the item given the specified parameters
6548 ; all arrays must be the same size
6549 ; created enchantments are not purged from the save when removed or overwritten
6550 ; exact same enchantments are re-used by the game
6551 Function CreateEnchantment(Actor akActor, int handSlot, int slotMask, float maxCharge,
MagicEffect[] effects, float[] magnitudes, int[] areas, int[] durations) global native
6552
6553 ; Returns the number of ref aliases holding this reference
6554 int Function GetNumReferenceAliases(Actor akActor, int handSlot, int slotMask) global
native
6555
6556 ; Returns the nth ReferenceAlias holding this reference
6557 ReferenceAlias Function GetNthReferenceAlias(Actor akActor, int handSlot, int slotMask,
int n) global native
6558
6559 ; Returns the poison on the specified item
6560 Potion Function GetPoison(Actor akActor, int handSlot, int slotMask) global native
6561
6562 ; Returns all of the ReferenceAlias holding this reference

```

```
6563 ReferenceAlias[] Function GetReferenceAliases(Actor akActor, int handSlot, int slotMask)
global native    :: Add a newline between files
6564
```