TES3MP Documentation

Release 0.8.1

TES3MP Team

CONTENTS

1	TES3MP's Lua API reference		
	1.1	Actor functions	
	1.2	Book functions	
	1.3	Cell functions	
	1.4	Char class functions	
	1.5	Chat functions	
	1.6	Dialogue functions	
	1.7	Faction functions	
	1.8	GUI functions	
	1.9	Item functions	
	1.10	Mechanics functions	
	1.11	Miscellaneous functions	
	1.12	Object functions	
	1.13	Position functions	
	1.14	Quest functions	
	1.15	Records Dynamic functions	
	1.16	Server functions	
	1.17	Setting functions	
	1.18	Shapeshift functions	
	1.19	Spell functions	
	1.20	Stats functions	
	1.21	Worldstate functions	
Ind	lex	13	

Contents:

CONTENTS 1

2 CONTENTS

TES3MP'S LUA API REFERENCE

1.1 Actor functions

class ActorFunctions

Public Static Functions

static void ReadReceivedActorList() noexcept

Use the last actor list received by the server as the one being read.

Returns

void

static void ReadCellActorList(const char *cellDescription) noexcept

Use the temporary actor list stored for a cell as the one being read.

This type of actor list is used to store actor positions and dynamic stats and is deleted when the cell is unloaded.

Parameters

cellDescription – The description of the cell whose actor list should be read.

Returns

void

static void **ClearActorList()** noexcept

Clear the data from the actor list stored on the server.

Returns

void

static void SetActorListPid(unsigned short pid) noexcept

Set the pid attached to the ActorList.

Parameters

pid – The player ID to whom the actor list should be attached.

Returns

void

static void CopyReceivedActorListToStore() noexcept

Take the contents of the read-only actor list last received by the server from a player and move its contents to the stored object list that can be sent by the server.

Returns

void

static unsigned int GetActorListSize() noexcept

Get the number of indexes in the read actor list.

Returns

The number of indexes.

static unsigned char **GetActorListAction()** noexcept

Get the action type used in the read actor list.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE, 3 for REQUEST).

static const char *GetActorCell(unsigned int index) noexcept

Get the cell description of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The cell description.

static const char *GetActorRefId(unsigned int index) noexcept

Get the refId of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The refId.

static unsigned int GetActorRefNum(unsigned int index) noexcept

Get the refNum of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The refNum.

static unsigned int GetActorMpNum (unsigned int index) noexcept

Get the mpNum of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The mpNum.

static double **GetActorPosX**(unsigned int index) noexcept

Get the X position of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The X position.

static double GetActorPosY(unsigned int index) noexcept

Get the Y position of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The Y position.

static double GetActorPosZ(unsigned int index) noexcept

Get the Z position of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The Z position.

static double **GetActorRotX** (unsigned int index) noexcept

Get the X rotation of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The X rotation.

static double **GetActorRotY**(unsigned int index) noexcept

Get the Y rotation of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The Y rotation.

static double ${\tt GetActorRotZ}(unsigned\ int\ index)\ noexcept$

Get the Z rotation of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The Z rotation.

static double **GetActorHealthBase**(unsigned int index) noexcept

Get the base health of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The base health.

static double GetActorHealthCurrent(unsigned int index) noexcept

Get the current health of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The current health.

static double **GetActorHealthModified**(unsigned int index) noexcept

Get the modified health of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The modified health.

static double ${\tt GetActorMagickaBase}$ (unsigned int index) no except

Get the base magicka of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The base magicka.

static double **GetActorMagickaCurrent** (unsigned int index) noexcept

Get the current magicka of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The current magicka.

static double ${\tt GetActorMagickaModified}$ (unsigned int index) no except

Get the modified magicka of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The modified magicka.

static double GetActorFatigueBase(unsigned int index) noexcept

Get the base fatigue of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The base fatigue.

$static\ double\ \textbf{GetActorFatigueCurrent} (unsigned\ int\ index)\ no except$

Get the current fatigue of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The current fatigue.

static double **GetActorFatigueModified**(unsigned int index) noexcept

Get the modified fatigue of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The modified fatigue.

static const char *GetActorEquipmentItemRefId(unsigned int index, unsigned short slot) noexcept

Get the refId of the item in a certain slot of the equipment of the actor at a certain index in the read actor list.

Parameters

- index The index of the actor.
- **slot** The slot of the equipment item.

Returns

The refId.

 $static \ int \ \textbf{GetActorEquipmentItemCount} (unsigned \ int \ index, \ unsigned \ short \ slot) \ no except$

Get the count of the item in a certain slot of the equipment of the actor at a certain index in the read actor list.

Parameters

- index The index of the actor.
- **slot** The slot of the equipment item.

Returns

The item count.

static int GetActorEquipmentItemCharge (unsigned int index, unsigned short slot) noexcept

Get the charge of the item in a certain slot of the equipment of the actor at a certain index in the read actor list.

Parameters

- index The index of the actor.
- **slot** The slot of the equipment item.

Returns

The charge.

static double **GetActorEquipmentItemEnchantmentCharge**(unsigned int index, unsigned short slot) noexcept

Get the enchantment charge of the item in a certain slot of the equipment of the actor at a certain index in the read actor list.

Parameters

- index The index of the actor.
- **slot** The slot of the equipment item.

Returns

The enchantment charge.

static bool DoesActorHavePlayerKiller (unsigned int index) noexcept

Check whether the killer of the actor at a certain index in the read actor list is a player.

Parameters

index – The index of the actor.

Returns

Whether the actor was killed by a player.

static int GetActorKillerPid(unsigned int index) noexcept

Get the player ID of the killer of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The player ID of the killer.

static const char *GetActorKillerRefId(unsigned int index) noexcept

Get the refId of the actor killer of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The refld of the killer.

static unsigned int GetActorKillerRefNum(unsigned int index) noexcept

Get the refNum of the actor killer of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The refNum of the killer.

static unsigned int GetActorKillerMpNum(unsigned int index) noexcept

Get the mpNum of the actor killer of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The mpNum of the killer.

static const char *GetActorKillerName (unsigned int index) noexcept

Get the name of the actor killer of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The name of the killer.

static unsigned int GetActorDeathState(unsigned int index) noexcept

Get the deathState of the actor at a certain index in the read actor list.

Parameters

index – The index of the actor.

Returns

The deathState.

static unsigned int GetActorSpellsActiveChangesSize(unsigned int actorIndex) noexcept

Get the number of indexes in an actor's latest spells active changes.

Parameters

actorIndex – The index of the actor.

Returns

The number of indexes for spells active changes.

static unsigned int GetActorSpellsActiveChangesAction (unsigned int actorIndex) noexcept

Get the action type used in an actor's latest spells active changes.

Parameters

actorIndex – The index of the actor.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE).

static const char ***GetActorSpellsActiveId**(unsigned int actorIndex, unsigned int spellIndex) noexcept Get the spell id at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The spell id.

static const char *GetActorSpellsActiveDisplayName(unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the spell display name at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The spell display name.

static bool **GetActorSpellsActiveStackingState**(unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the spell stacking state at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The spell stacking state.

static unsigned int **GetActorSpellsActiveEffectCount** (unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the number of effects at an index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The number of effects.

 $static \ unsigned \ int \ \textbf{GetActorSpellsActiveEffectId} (unsigned \ int \ actorIndex, \ unsigned \ int \ spellIndex, \ unsigned \ int \ effectIndex) \ no except$

Get the id for an effect index at a spell index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The id of the effect.

static int **GetActorSpellsActiveEffectArg**(unsigned int actorIndex, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the arg for an effect index at a spell index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The arg of the effect.

static double **GetActorSpellsActiveEffectMagnitude**(unsigned int actorIndex, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the magnitude for an effect index at a spell index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The magnitude of the effect.

static double **GetActorSpellsActiveEffectDuration**(unsigned int actorIndex, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the duration for an effect index at a spell index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The duration of the effect.

 $static\ double\ \textbf{GetActorSpellsActiveEffectTimeLeft} (unsigned\ int\ actorIndex,\ unsigned\ int\ spellIndex,\ unsigned\ int\ effectIndex)\ noexcept$

Get the time left for an effect index at a spell index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

• **effectIndex** – The index of the effect.

Returns

The time left for the effect.

static bool **DoesActorSpellsActiveHavePlayerCaster** (unsigned int actorIndex, unsigned int spellIndex) noexcept

Check whether the spell at a certain index in an actor's latest spells active changes has a player as its caster.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

Whether a player is the caster of the spell.

static int **GetActorSpellsActiveCasterPid**(unsigned int actorIndex, unsigned int spellIndex) noexcept Get the player ID of the caster of the spell at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The player ID of the caster.

static const char *GetActorSpellsActiveCasterRefId(unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the refld of the actor caster of the spell at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The refld of the caster.

static unsigned int **GetActorSpellsActiveCasterRefNum**(unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the refNum of the actor caster of the spell at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The refNum of the caster.

static unsigned int **GetActorSpellsActiveCasterMpNum**(unsigned int actorIndex, unsigned int spellIndex) noexcept

Get the mpNum of the actor caster of the spell at a certain index in an actor's latest spells active changes.

Parameters

- actorIndex The index of the actor.
- **spellIndex** The index of the spell.

Returns

The mpNum of the caster.

static bool DoesActorHavePosition(unsigned int index) noexcept

Check whether there is any positional data for the actor at a certain index in the read actor list.

This is only useful when reading the actor list data recorded for a particular cell.

Parameters

index – The index of the actor.

Returns

Whether the read actor list contains positional data.

static bool DoesActorHaveStatsDynamic(unsigned int index) noexcept

Check whether there is any dynamic stats data for the actor at a certain index in the read actor list.

This is only useful when reading the actor list data recorded for a particular cell.

Parameters

index – The index of the actor.

Returns

Whether the read actor list contains dynamic stats data.

static void **SetActorListCell**(const char *cellDescription) noexcept

Set the cell of the temporary actor list stored on the server.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

cellDescription – The description of the cell.

Returns

void

static void **SetActorListAction** (unsigned char action) noexcept

Set the action type of the temporary actor list stored on the server.

Parameters

action – The action type (0 for SET, 1 for ADD, 2 for REMOVE, 3 for REQUEST).

Returns

void

static void **SetActorCell**(const char *cellDescription) noexcept

Set the cell of the temporary actor stored on the server.

Used for ActorCellChange packets, where a specific actor's cell now differs from that of the actor list.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

cellDescription – The description of the cell.

Returns

void

static void **SetActorRefId**(const char *refId) noexcept

Set the refId of the temporary actor stored on the server.

```
Parameters
```

refId - The refId.

Returns

void

static void **SetActorRefNum**(int refNum) noexcept

Set the refNum of the temporary actor stored on the server.

Parameters

refNum - The refNum.

Returns

void

static void **SetActorMpNum**(int mpNum) noexcept

Set the mpNum of the temporary actor stored on the server.

Parameters

mpNum - The mpNum.

Returns

void

static void **SetActorPosition**(double x, double y, double z) noexcept

Set the position of the temporary actor stored on the server.

Parameters

- \mathbf{x} The X position.
- **y** The Y position.
- **z** The Z position.

Returns

void

static void **SetActorRotation**(double x, double y, double z) noexcept

Set the rotation of the temporary actor stored on the server.

Parameters

- **x** The X rotation.
- y The Y rotation.
- **z** The Z rotation.

Returns

void

static void SetActorHealthBase(double value) noexcept

Set the base health of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void **SetActorHealthCurrent**(double value) noexcept

Set the current health of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void SetActorHealthModified(double value) noexcept

Set the modified health of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void SetActorMagickaBase(double value) noexcept

Set the base magicka of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

$static\ void\ \textbf{SetActorMagickaCurrent}$ (double value) no except

Set the current magicka of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void **SetActorMagickaModified**(double value) noexcept

Set the modified magicka of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void **SetActorFatigueBase**(double value) noexcept

Set the base fatigue of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void **SetActorFatigueCurrent**(double value) noexcept

Set the current fatigue of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

static void SetActorFatigueModified(double value) noexcept

Set the modified fatigue of the temporary actor stored on the server.

Parameters

value – The new value.

Returns

void

static void **SetActorSound**(const char *sound) noexcept

Set the sound of the temporary actor stored on the server.

Parameters

sound – The sound.

Returns

void

static void **SetActorDeathState**(unsigned int deathState) noexcept

Set the deathState of the temporary actor stored on the server.

Parameters

deathState – The deathState.

Returns

void

static void **SetActorDeathInstant**(bool isInstant) noexcept

Set whether the death of the temporary actor stored on the server should be instant or not.

Parameters

isInstant – Whether the death should be instant.

Returns

void

static void **SetActorSpellsActiveAction**(unsigned char action) noexcept

Set the action type in the spells active changes of the temporary actor stored on the server.

Parameters

action – The action (0 for SET, 1 for ADD, 2 for REMOVE).

Returns

void

static void **SetActorAlAction**(unsigned int action) noexcept

Set the AI action of the temporary actor stored on the server.

Parameters

action - The new action.

Returns

void

$static\ void\ \textbf{SetActorAITargetToPlayer} (unsigned\ short\ pid)\ no except$

Set a player as the AI target of the temporary actor stored on the server.

Parameters

pid - The player ID.

Returns

void

static void **SetActorAITargetToObject**(int refNum, int mpNum) noexcept

Set another object as the AI target of the temporary actor stored on the server.

Parameters

- **refNum** The refNum of the target object.
- mpNum The mpNum of the target object.

Returns

void

static void **SetActorAICoordinates**(double x, double y, double z) noexcept

Set the coordinates for the AI package associated with the current AI action.

Parameters

- **x** The X coordinate.
- **y** The Y coordinate.
- **z** The Z coordinate.

Returns

void

static void **SetActorAIDistance** (unsigned int distance) noexcept

Set the distance of the AI package associated with the current AI action.

Parameters

distance – The distance of the package.

Returns

void

static void **SetActorAIDuration**(unsigned int duration) noexcept

Set the duration of the AI package associated with the current AI action.

Parameters

duration – The duration of the package.

Returns

void

static void **SetActorAIRepetition**(bool shouldRepeat) noexcept

Set whether the current AI package should be repeated.

Note: This only has an effect on the WANDER package.

Parameters

shouldRepeat – Whether the package should be repeated.

Returns

void

static void **EquipActorItem**(unsigned short slot, const char *refId, unsigned int count, int charge, double enchantmentCharge = -1) noexcept

Equip an item in a certain slot of the equipment of the temporary actor stored on the server.

Parameters

- **slot** The equipment slot.
- **refId** The refId of the item.

- **count** The count of the item.
- **charge** The charge of the item.
- **enchantmentCharge** The enchantment charge of the item.

Returns

void

static void **UnequipActorItem**(unsigned short slot) noexcept

Unequip the item in a certain slot of the equipment of the temporary actor stored on the server.

Parameters

slot – The equipment slot.

Returns

void

static void **AddActorSpellActive**(const char *spellId, const char *displayName, bool stackingState) noexcept

Add a new active spell to the spells active changes for the temporary actor stored, on the server, using the temporary effect values stored so far.

Parameters

- **spellId** The spellId of the spell.
- **displayName** The displayName of the spell.
- **stackingState** Whether the spell should stack with other instances of itself.

Returns

void

static void **AddActorSpellActiveEffect**(int effectId, double magnitude, double duration, double timeLeft, int arg) noexcept

Add a new effect to the next active spell that will be added to the temporary actor stored on the server.

Parameters

- **effectId** The id of the effect.
- magnitude The magnitude of the effect.
- **duration** The duration of the effect.
- timeLeft The timeLeft for the effect.
- **arg** The arg of the effect when applicable, e.g. the skill used for Fortify Skill or the attribute used for Fortify Attribute.

Returns

void

static void AddActor() noexcept

Add a copy of the server's temporary actor to the server's temporary actor list.

In the process, the server's temporary actor will automatically be cleared so a new one can be set up.

Returns

void

static void **SendActorList()** noexcept

Send an ActorList packet.

It is sent only to the player for whom the current actor list was initialized.

Returns

void

static void SendActorAuthority() noexcept

Send an ActorAuthority packet.

The player for whom the current actor list was initialized is recorded in the server memory as the new actor authority for the actor list's cell.

The packet is sent to that player as well as all other players who have the cell loaded.

Returns

void

static void **SendActorPosition**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorPosition packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorStatsDynamic**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorStatsDynamic packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorEquipment** (bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorEquipment packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

static void **SendActorSpellsActiveChanges**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorSpellsActive packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorSpeech**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorSpeech packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorDeath**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorDeath packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorAI** (bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorAI packet.

Parameters

- **sendToOtherVisitors** Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendActorCellChange**(bool sendToOtherVisitors, bool skipAttachedPlayer) noexcept Send an ActorCellChange packet.

Parameters

• **sendToOtherVisitors** – Whether this packet should be sent to cell visitors other than the player attached to the packet (false by default).

• **skipAttachedPlayer** – Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.2 Book functions

class BookFunctions

Public Static Functions

static void ClearBookChanges (unsigned short pid) noexcept

Clear the last recorded book changes for a player.

This is used to initialize the sending of new PlayerBook packets.

Parameters

pid – The player ID whose book changes should be used.

Returns

void

static unsigned int GetBookChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest book changes.

Parameters

pid – The player ID whose book changes should be used.

Returns

The number of indexes.

static void ${\bf AddBook} ({\bf unsigned\ short\ pid},\, {\bf const\ char\ *bookId})\ no except$

Add a new book to the book changes for a player.

Parameters

- **pid** The player ID whose book changes should be used.
- **bookId** The bookId of the book.

Returns

void

static const char *GetBookId(unsigned short pid, unsigned int index) noexcept

Get the bookId at a certain index in a player's latest book changes.

Parameters

- pid The player ID whose book changes should be used.
- **index** The index of the book.

Returns

The bookId.

static void **SendBookChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerBook packet with a player's recorded book changes.

Parameters

- **pid** The player ID whose book changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.3 Cell functions

class CellFunctions

Public Static Functions

static unsigned int **GetCellStateChangesSize**(unsigned short pid) noexcept Get the number of indexes in a player's latest cell state changes.

Parameters

pid – The player ID whose cell state changes should be used.

Returns

The number of indexes.

static unsigned int **GetCellStateType**(unsigned short pid, unsigned int index) noexcept Get the cell state type at a certain index in a player's latest cell state changes.

Parameters

- pid The player ID whose cell state changes should be used.
- **index** The index of the cell state.

Returns

The cell state type (0 for LOAD, 1 for UNLOAD).

static const char *GetCellStateDescription(unsigned short pid, unsigned int index) noexcept Get the cell description at a certain index in a player's latest cell state changes.

Parameters

- **pid** The player ID whose cell state changes should be used.
- index The index of the cell state.

Returns

The cell description.

1.3. Cell functions 21

static const char *GetCell(unsigned short pid) noexcept

Get the cell description of a player's cell.

Parameters

pid – The player ID.

Returns

The cell description.

static int **GetExteriorX**(unsigned short pid) noexcept

Get the X coordinate of the player's exterior cell.

Parameters

pid – The player ID.

Returns

The X coordinate of the cell.

static int GetExteriorY(unsigned short pid) noexcept

Get the Y coordinate of the player's exterior cell.

Parameters

pid – The player ID.

Returns

The Y coordinate of the cell.

static bool IsInExterior(unsigned short pid) noexcept

Check whether the player is in an exterior cell or not.

Parameters

pid – The player ID.

Returns

Whether the player is in an exterior cell.

static const char *GetRegion(unsigned short pid) noexcept

Get the region of the player's exterior cell.

A blank value will be returned if the player is in an interior.

Parameters

pid – The player ID.

Returns

The region.

static bool IsChangingRegion (unsigned short pid) noexcept

Check whether the player's last cell change has involved a region change.

Parameters

pid – The player ID.

Returns

Whether the player has changed their region.

static void **SetCell** (unsigned short pid, const char *cellDescription) noexcept

Set the cell of a player.

This changes the cell recorded for that player in the server memory, but does not by itself send a packet.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

- **pid** The player ID.
- **cellDescription** The cell description.

Returns

void

static void **SetExteriorCell**(unsigned short pid, int x, int y) noexcept

Set the cell of a player to an exterior cell.

This changes the cell recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- \mathbf{x} The X coordinate of the cell.
- y The Y coordinate of the cell.

Returns

void

static void SendCell(unsigned short pid) noexcept

Send a PlayerCellChange packet about a player.

It is only sent to the affected player.

Parameters

pid – The player ID.

Returns

void

1.4 Char class functions

class CharClassFunctions

Public Static Functions

static const char *GetDefaultClass(unsigned short pid) noexcept

Get the default class used by a player.

Parameters

pid – The player ID.

Returns

The ID of the default class.

static const char *GetClassName(unsigned short pid) noexcept

Get the name of the custom class used by a player.

Parameters

pid – The player ID.

Returns

The name of the custom class.

static const char *GetClassDesc(unsigned short pid) noexcept

Get the description of the custom class used by a player.

Parameters

pid – The player ID.

Returns

The description of the custom class.

static int GetClassMajorAttribute(unsigned short pid, unsigned char slot)

Get the ID of one of the two major attributes of a custom class used by a player.

Parameters

- pid The player ID.
- **slot** The slot of the major attribute (0 or 1).

Returns

The ID of the major attribute.

static int GetClassSpecialization(unsigned short pid) noexcept

Get the specialization ID of the custom class used by a player.

Parameters

pid – The player ID.

Returns

The specialization ID of the custom class (0 for Combat, 1 for Magic, 2 for Stealth).

static int **GetClassMajorSkill** (unsigned short pid, unsigned char slot)

Get the ID of one of the five major skills of a custom class used by a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the major skill (0 to 4).

Returns

The ID of the major skill.

static int **GetClassMinorSkill** (unsigned short pid, unsigned char slot)

Get the ID of one of the five minor skills of a custom class used by a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the minor skill (0 to 4).

Returns

The ID of the minor skill.

static int ${\tt IsClassDefault}$ (unsigned short pid) noexcept

Check whether the player is using a default class instead of a custom one.

Parameters

pid – The player ID.

Returns

Whether the player is using a default class.

static void SetDefaultClass (unsigned short pid, const char *id) noexcept

Set the default class used by a player.

If this is left blank, the custom class data set for the player will be used instead.

Parameters

- **pid** The player ID.
- id The ID of the default class.

Returns

void

static void SetClassName (unsigned short pid, const char *name) noexcept

Set the name of the custom class used by a player.

Parameters

- **pid** The player ID.
- name The name of the custom class.

Returns

void

static void **SetClassDesc**(unsigned short pid, const char *desc) noexcept

Set the description of the custom class used by a player.

Parameters

- **pid** The player ID.
- **desc** The description of the custom class.

Returns

void

static void SetClassMajorAttribute(unsigned short pid, unsigned char slot, int attrId)

Set the ID of one of the two major attributes of the custom class used by a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the major attribute (0 or 1).
- attrId The ID to use for the attribute.

Returns

void

static void **SetClassSpecialization** (unsigned short pid, int spec) noexcept

Set the specialization of the custom class used by a player.

Parameters

- pid The player ID.
- **spec** The specialization ID to use (0 for Combat, 1 for Magic, 2 for Stealth).

Returns

static void SetClassMajorSkill (unsigned short pid, unsigned char slot, int skillId)

Set the ID of one of the five major skills of the custom class used by a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the major skill (0 to 4).
- **skillId** The ID to use for the skill.

Returns

void

static void SetClassMinorSkill (unsigned short pid, unsigned char slot, int skillId)

Set the ID of one of the five minor skills of the custom class used by a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the minor skill (0 to 4).
- **skillId** The ID to use for the skill.

Returns

void

static void SendClass (unsigned short pid) noexcept

Send a PlayerCharClass packet about a player.

It is only sent to the affected player.

Parameters

pid – The player ID.

Returns

void

1.5 Chat functions

class ChatFunctions

Public Static Functions

static void **SendMessage** (unsigned short pid, const char *message, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a message to a certain player.

Parameters

- pid The player ID.
- **message** The contents of the message.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **CleanChatForPid**(unsigned short pid)

Remove all messages from chat for a certain player.

Parameters

pid – The player ID.

Returns

void

static void CleanChat()

Remove all messages from chat for everyone on the server.

Returns

void

1.6 Dialogue functions

class DialogueFunctions

Public Static Functions

static void ClearTopicChanges (unsigned short pid) noexcept

Clear the last recorded topic changes for a player.

This is used to initialize the sending of new PlayerTopic packets.

Parameters

pid – The player ID whose topic changes should be used.

Returns

void

static unsigned int **GetTopicChangesSize** (unsigned short pid) noexcept

Get the number of indexes in a player's latest topic changes.

Parameters

pid – The player ID whose topic changes should be used.

Returns

The number of indexes.

static void **AddTopic** (unsigned short pid, const char *topicId) noexcept

Add a new topic to the topic changes for a player.

Parameters

- **pid** The player ID whose topic changes should be used.
- **topicId** The topicId of the topic.

Returns

static const char *GetTopicId(unsigned short pid, unsigned int index) noexcept

Get the topicId at a certain index in a player's latest topic changes.

Parameters

- pid The player ID whose topic changes should be used.
- **index** The index of the topic.

Returns

The topicId.

static void **SendTopicChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerTopic packet with a player's recorded topic changes.

Parameters

- pid The player ID whose topic changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **PlayAnimation**(unsigned short pid, const char *groupname, int mode, int count, bool persist) noexcept

Play a certain animation on a player's character by sending a PlayerAnimation packet.

Parameters

- pid The player ID of the character playing the animation.
- **groupname** The groupname of the animation.
- **mode** The mode of the animation.
- **count** The number of times the animation should be played.
- **persist** Whether the animation should persist or not.

Returns

void

static void PlaySpeech (unsigned short pid, const char *sound) noexcept

Play a certain sound for a player as spoken by their character by sending a PlayerSpeech packet.

Parameters

- **pid** The player ID of the character playing the sound.
- **sound** The path of the sound file.

Returns

1.7 Faction functions

class FactionFunctions

Public Static Functions

static void ClearFactionChanges (unsigned short pid) noexcept

Clear the last recorded faction changes for a player.

This is used to initialize the sending of new PlayerFaction packets.

Parameters

pid – The player ID whose faction changes should be used.

Returns

void

static unsigned int GetFactionChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest faction changes.

Parameters

pid – The player ID whose faction changes should be used.

Returns

The number of indexes.

static unsigned char **GetFactionChangesAction**(unsigned short pid) noexcept

Get the action type used in a player's latest faction changes.

Parameters

pid – The player ID whose faction changes should be used.

Returns

The action type (0 for RANK, 1 for EXPULSION, 2 for REPUTATION).

static const char ***GetFactionId**(unsigned short pid, unsigned int index) noexcept Get the factionId at a certain index in a player's latest faction changes.

Parameters

- pid The player ID whose faction changes should be used.
- index The index of the faction.

Returns

The factionId.

static int GetFactionRank (unsigned short pid, unsigned int index) noexcept

Get the rank at a certain index in a player's latest faction changes.

Parameters

- pid The player ID whose faction changes should be used.
- index The index of the faction.

Returns

The rank.

1.7. Faction functions 29

static bool **GetFactionExpulsionState** (unsigned short pid, unsigned int index) noexcept Get the expulsion state at a certain index in a player's latest faction changes.

Parameters

- pid The player ID whose faction changes should be used.
- index The index of the faction.

Returns

The expulsion state.

static int ${\tt GetFactionReputation}$ (unsigned short pid, unsigned int index) no except

Get the reputation at a certain index in a player's latest faction changes.

Parameters

- **pid** The player ID whose faction changes should be used.
- index The index of the faction.

Returns

The reputation.

static void **SetFactionChangesAction** (unsigned short pid, unsigned char action) noexcept Set the action type in a player's faction changes.

Parameters

- pid The player ID whose faction changes should be used.
- action The action (0 for RANK, 1 for EXPULSION, 2 for REPUTATION).

Returns

void

static void ${f SetFactionId}$ (const char *factionId) noexcept

Set the factionId of the temporary faction stored on the server.

Parameters

factionId – The factionId.

Returns

void

static void **SetFactionRank** (unsigned int rank) noexcept

Set the rank of the temporary faction stored on the server.

Parameters

rank – The rank.

Returns

void

 $static\ void\ \textbf{SetFactionExpulsionState} (bool\ expulsionState)\ no except$

Set the expulsion state of the temporary faction stored on the server.

Parameters

expulsionState – The expulsion state.

Returns

static void **SetFactionReputation**(int reputation) noexcept

Set the reputation of the temporary faction stored on the server.

Parameters

reputation – The reputation.

Returns

void

static void AddFaction (unsigned short pid) noexcept

Add the server's temporary faction to the faction changes for a player.

In the process, the server's temporary faction will automatically be cleared so a new one can be set up.

Parameters

pid – The player ID whose faction changes should be used.

Returns

void

static void **SendFactionChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerFaction packet with a player's recorded faction changes.

Parameters

- **pid** The player ID whose faction changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.8 GUI functions

class **GUIFunctions**

Public Static Functions

static void _MessageBox(unsigned short pid, int id, const char *label) noexcept

Display a simple messagebox at the bottom of the screen that vanishes after a few seconds.

Note for C++ programmers: do not rename into MessageBox so as to not conflict with WINAPI's MessageBox.

Parameters

- **pid** The player ID for whom the messagebox should appear.
- id The numerical ID of the messagebox.
- label The text in the messagebox.

Returns

void

1.8. GUI functions 31

static void **CustomMessageBox** (unsigned short pid, int id, const char *label, const char *buttons) noexcept Display an interactive messagebox at the center of the screen that vanishes only when one of its buttons is clicked.

Parameters

- **pid** The player ID for whom the messagebox should appear.
- id The numerical ID of the messagebox.
- label The text in the messagebox. \parm buttons The captions of the buttons, separated by semicolons (e.g. "Yes;No;Maybe").

Returns

void

static void **InputDialog**(unsigned short pid, int id, const char *label, const char *note) noexcept Display an input dialog at the center of the screen.

Parameters

- **pid** The player ID for whom the input dialog should appear.
- id The numerical ID of the input dialog.
- label The text at the top of the input dialog. \parm note The text at the bottom of the input dialog.

Returns

void

static void **PasswordDialog** (unsigned short pid, int id, const char *label, const char *note) noexcept Display a password dialog at the center of the screen.

Although similar to an input dialog, the password dialog replaces all input characters with asterisks.

Parameters

- pid The player ID for whom the password dialog should appear.
- id The numerical ID of the password dialog.
- label The text at the top of the password dialog. \parm note The text at the bottom of the password dialog.

Returns

void

static void ListBox (unsigned short pid, int id, const char *label, const char *items)

Display a listbox at the center of the screen where each item takes up a row and is selectable, with the listbox only vanishing once the Ok button is pressed.

Parameters

- **pid** The player ID for whom the listbox should appear.
- **id** The numerical ID of the listbox.
- label The text at the top of the listbox. \parm items The items in the listbox, separated by newlines (e.g. "Item 1\nItem 2").

Returns

static void ClearQuickKeyChanges (unsigned short pid) noexcept

Clear the last recorded quick key changes for a player.

This is used to initialize the sending of new PlayerQuickKeys packets.

Parameters

pid – The player ID whose quick key changes should be used.

Returns

void

static unsigned int GetQuickKeyChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest quick key changes.

Parameters

pid – The player ID whose quick key changes should be used.

Returns

The number of indexes.

static void **AddQuickKey** (unsigned short pid, unsigned short slot, int type, const char *itemId = "") noexcept Add a new quick key to the quick key changes for a player.

Parameters

- **pid** The player ID whose quick key changes should be used.
- **slot** The slot to be used.
- **type** The type of the quick key (0 for ITEM, 1 for ITEM_MAGIC, 2 for MAGIC, 3 for UNASSIGNED).
- **itemId** The itemId of the item.

Returns

void

static int GetQuickKeySlot(unsigned short pid, unsigned int index) noexcept

Get the slot of the quick key at a certain index in a player's latest quick key changes.

Parameters

- **pid** The player ID whose quick key changes should be used.
- **index** The index of the quick key in the quick key changes vector.

Returns

The slot.

static int GetQuickKeyType(unsigned short pid, unsigned int index) noexcept

Get the type of the quick key at a certain index in a player's latest quick key changes.

Parameters

- pid The player ID whose quick key changes should be used.
- **index** The index of the quick key in the quick key changes vector.

Returns

The quick key type.

static const char *GetQuickKeyItemId (unsigned short pid, unsigned int index) noexcept Get the itemId at a certain index in a player's latest quick key changes.

Parameters

1.8. GUI functions 33

- **pid** The player ID whose quick key changes should be used.
- **index** The index of the quick key in the quick key changes vector.

The itemId.

static void SendQuickKeyChanges (unsigned short pid) noexcept

Send a PlayerQuickKeys packet with a player's recorded quick key changes.

Parameters

pid – The player ID whose quick key changes should be used.

Returns

void

static void **SetMapVisibility**(unsigned short targetPid, unsigned short affectedPid, unsigned short state) noexcept

Determine whether a player can see the map marker of another player.

Note: This currently has no effect, and is just an unimplemented stub.

Parameters

- targetPid The player ID whose map marker should be hidden or revealed.
- affectedPid The player ID for whom the map marker will be hidden or revealed.
- **state** The state of the map marker (false to hide, true to reveal).

Returns

void

static void SetMapVisibilityAll (unsigned short targetPid, unsigned short state) noexcept

Determine whether a player's map marker can be seen by all other players.

Note: This currently has no effect, and is just an unimplemented stub.

Parameters

- targetPid The player ID whose map marker should be hidden or revealed.
- **state** The state of the map marker (false to hide, true to reveal).

Returns

void

1.9 Item functions

class ItemFunctions

Public Static Functions

static void ClearInventoryChanges (unsigned short pid) noexcept

Clear the last recorded inventory changes for a player.

This is used to initialize the sending of new PlayerInventory packets.

Parameters

pid – The player ID whose inventory changes should be used.

Returns

void

static int GetEquipmentSize() noexcept

Get the number of slots used for equipment.

The number is 19 before any dehardcoding is done in OpenMW.

Returns

The number of slots.

static unsigned int GetEquipmentChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest equipment changes.

Parameters

pid – The player ID whose equipment changes should be used.

Returns

The number of indexes.

static unsigned int GetInventoryChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest inventory changes.

Parameters

pid – The player ID whose inventory changes should be used.

Returns

The number of indexes.

static unsigned int **GetInventoryChangesAction**(unsigned short pid) noexcept

Get the action type used in a player's latest inventory changes.

Parameters

pid – The player ID whose inventory changes should be used.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE).

static void SetInventoryChangesAction (unsigned short pid, unsigned char action) noexcept

Set the action type in a player's inventory changes.

Parameters

- pid The player ID whose inventory changes should be used.
- action The action (0 for SET, 1 for ADD, 2 for REMOVE).

Returns

void

1.9. Item functions 35

static void **EquipItem**(unsigned short pid, unsigned short slot, const char *refId, unsigned int count, int charge, double enchantmentCharge = -1) noexcept

Equip an item in a certain slot of the equipment of a player.

Parameters

- **pid** The player ID.
- **slot** The equipment slot.
- **refId** The refId of the item.
- **count** The count of the item.
- **charge** The charge of the item.
- **enchantmentCharge** The enchantment charge of the item.

Returns

void

static void **UnequipItem**(unsigned short pid, unsigned short slot) noexcept

Unequip the item in a certain slot of the equipment of a player.

Parameters

- **pid** The player ID.
- **slot** The equipment slot.

Returns

void

static void **AddItemChange** (unsigned short pid, const char *refId, unsigned int count, int charge, double enchantmentCharge, const char *soul) noexcept

Add an item change to a player's inventory changes.

Parameters

- **pid** The player ID.
- **refId** The refId of the item.
- **count** The count of the item.
- **charge** The charge of the item.
- $\bullet \ \ \textbf{enchantmentCharge} The \ enchantment \ charge \ of \ the \ item.$
- **soul** The soul of the item.

Returns

void

static bool **HasItemEquipped**(unsigned short pid, const char *refId)

Check whether a player has equipped an item with a certain refId in any slot.

Parameters

- **pid** The player ID.
- **refId** The refId of the item.

Returns

Whether the player has the item equipped.

static int **GetEquipmentChangesSlot** (unsigned short pid, unsigned int changeIndex) noexcept

Get the slot used for the equipment item at a specific index in the most recent equipment changes.

Parameters

- **pid** The player ID.
- **changeIndex** The index of the equipment change.

Returns

The slot.

static const char *GetEquipmentItemRefId(unsigned short pid, unsigned short slot) noexcept

Get the refId of the item in a certain slot of the equipment of a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the equipment item.

Returns

The refId.

static int **GetEquipmentItemCount** (unsigned short pid, unsigned short slot) noexcept

Get the count of the item in a certain slot of the equipment of a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the equipment item.

Returns

The item count.

static int **GetEquipmentItemCharge** (unsigned short pid, unsigned short slot) noexcept Get the charge of the item in a certain slot of the equipment of a player.

Parameters

- **pid** The player ID.
- **slot** The slot of the equipment item.

Returns

The charge.

static double **GetEquipmentItemEnchantmentCharge** (unsigned short pid, unsigned short slot) noexcept Get the enchantment charge of the item in a certain slot of the equipment of a player.

Parameters

- pid The player ID.
- **slot** The slot of the equipment item.

Returns

The enchantment charge.

static const char *GetInventoryItemRefId (unsigned short pid, unsigned int index) noexcept

Get the refId of the item at a certain index in a player's latest inventory changes.

Parameters

• **pid** – The player ID whose inventory changes should be used.

1.9. Item functions 37

• **index** – The index of the inventory item.

Returns

The refId.

static int **GetInventoryItemCount** (unsigned short pid, unsigned int index) noexcept Get the count of the item at a certain index in a player's latest inventory changes.

Parameters

- **pid** The player ID whose inventory changes should be used.
- **index** The index of the inventory item.

Returns

The item count.

static int **GetInventoryItemCharge** (unsigned short pid, unsigned int index) noexcept Get the charge of the item at a certain index in a player's latest inventory changes.

Parameters

- **pid** The player ID whose inventory changes should be used.
- **index** The index of the inventory item.

Returns

The charge.

static double **GetInventoryItemEnchantmentCharge** (unsigned short pid, unsigned int index) noexcept Get the enchantment charge of the item at a certain index in a player's latest inventory changes.

Parameters

- **pid** The player ID whose inventory changes should be used.
- index The index of the inventory item.

Returns

The enchantment charge.

static const char ***GetInventoryItemSoul** (unsigned short pid, unsigned int index) noexcept Get the soul of the item at a certain index in a player's latest inventory changes.

Parameters

- pid The player ID whose inventory changes should be used.
- **index** The index of the inventory item.

Returns

The soul.

static const char *GetUsedItemRefId(unsigned short pid) noexcept

Get the refId of the item last used by a player.

Parameters

pid – The player ID.

Returns

The refId.

static int GetUsedItemCount (unsigned short pid) noexcept

Get the count of the item last used by a player.

Parameters

pid – The player ID.

Returns

The item count.

static int GetUsedItemCharge (unsigned short pid) noexcept

Get the charge of the item last used by a player.

Parameters

pid – The player ID.

Returns

The charge.

static double **GetUsedItemEnchantmentCharge** (unsigned short pid) noexcept

Get the enchantment charge of the item last used by a player.

Parameters

pid – The player ID.

Returns

The enchantment charge.

static const char *GetUsedItemSoul (unsigned short pid) noexcept

Get the soul of the item last used by a player.

Parameters

pid – The player ID.

Returns

The soul.

static void **SendEquipment** (unsigned short pid) noexcept

Send a PlayerEquipment packet with a player's equipment.

It is always sent to all players.

Parameters

pid – The player ID whose equipment should be sent.

Returns

void

static void **SendInventoryChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerInventory packet with a player's recorded inventory changes.

Parameters

- pid The player ID whose inventory changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.9. Item functions 39

static void **SendItemUse**(unsigned short pid) noexcept

Send a PlayerItemUse causing a player to use their recorded usedItem.

Parameters

pid – The player ID affected.

Returns

void

1.10 Mechanics functions

class MechanicsFunctions

Public Static Functions

static void ClearAlliedPlayersForPlayer (unsigned short pid) noexcept

Clear the list of players who will be regarded as being player's allies.

Parameters

pid – The player ID.

Returns

void

 $static\ unsigned\ char\ \textbf{GetMiscellaneousChangeType} (unsigned\ short\ pid)\ no except$

Get the type of a PlayerMiscellaneous packet.

Parameters

pid – The player ID.

Returns

The type.

static const char *GetMarkCell(unsigned short pid) noexcept

Get the cell description of a player's Mark cell.

Parameters

pid – The player ID.

Returns

The cell description.

static double GetMarkPosX (unsigned short pid) noexcept

Get the X position of a player's Mark.

Parameters

pid – The player ID.

Returns

The X position.

static double GetMarkPosY (unsigned short pid) noexcept

Get the Y position of a player's Mark.

Parameters

pid – The player ID.

The Y position.

static double GetMarkPosZ(unsigned short pid) noexcept

Get the Z position of a player's Mark.

Parameters

pid - The player ID.

Returns

The Z position.

static double **GetMarkRotX** (unsigned short pid) noexcept

Get the X rotation of a player's Mark.

Parameters

pid – The player ID.

Returns

The X rotation.

static double GetMarkRotZ (unsigned short pid) noexcept

Get the Z rotation of a player's Mark.

Parameters

pid – The player ID.

Returns

The X rotation.

static const char *GetSelectedSpellId(unsigned short pid) noexcept

Get the ID of a player's selected spell.

Parameters

pid – The player ID.

Returns

The spell ID.

static bool DoesPlayerHavePlayerKiller (unsigned short pid) noexcept

Check whether the killer of a certain player is also a player.

Parameters

pid – The player ID of the killed player.

Returns

Whether the player was killed by another player.

static int GetPlayerKillerPid(unsigned short pid) noexcept

Get the player ID of the killer of a certain player.

Parameters

pid – The player ID of the killed player.

Returns

The player ID of the killer.

static const char *GetPlayerKillerRefId(unsigned short pid) noexcept

Get the refId of the actor killer of a certain player.

Parameters

pid – The player ID of the killed player.

The refId of the killer.

static unsigned int GetPlayerKillerRefNum(unsigned short pid) noexcept

Get the refNum of the actor killer of a certain player.

Parameters

pid – The player ID of the killed player.

Returns

The refNum of the killer.

static unsigned int GetPlayerKillerMpNum (unsigned short pid) noexcept

Get the mpNum of the actor killer of a certain player.

Parameters

pid – The player ID of the killed player.

Returns

The mpNum of the killer.

static const char *GetPlayerKillerName(unsigned short pid) noexcept

Get the name of the actor killer of a certain player.

Parameters

pid – The player ID of the killed player.

Returns

The name of the killer.

static unsigned int GetDrawState(unsigned short pid) noexcept

Get the draw state of a player (0 for nothing, 1 for drawn weapon, 2 for drawn spell).

Parameters

pid – The player ID.

Returns

The draw state.

static bool GetSneakState(unsigned short pid) noexcept

Get the sneak state of a player.

Parameters

pid – The player ID.

Returns

Whether the player is sneaking.

static void **SetMarkCell** (unsigned short pid, const char *cellDescription) noexcept

Set the Mark cell of a player.

This changes the Mark cell recorded for that player in the server memory, but does not by itself send a packet.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

- **pid** The player ID.
- **cellDescription** The cell description.

void

static void **SetMarkPos** (unsigned short pid, double x, double y, double z) noexcept

Set the Mark position of a player.

This changes the Mark positional coordinates recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **x** The X position.
- y The Y position.
- **z** The Z position.

Returns

void

static void SetMarkRot (unsigned short pid, double x, double z) noexcept

Set the Mark rotation of a player.

This changes the Mark positional coordinates recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **x** The X rotation.
- **z** The Z rotation.

Returns

void

static void SetSelectedSpellId (unsigned short pid, const char *spellId) noexcept

Set the ID of a player's selected spell.

This changes the spell ID recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **spellId** The spell ID.

Returns

void

static void **AddAlliedPlayerForPlayer** (unsigned short pid, unsigned short alliedPlayerPid) noexcept Add an ally to a player's list of allied players.

Parameters

- **pid** The player ID.
- alliedPlayerPid The ally's player ID.

Returns

void

static void **SendMarkLocation**(unsigned short pid)

Send a PlayerMiscellaneous packet with a Mark location to a player.

Parameters

pid – The player ID.

Returns

void

static void **SendSelectedSpell**(unsigned short pid)

Send a PlayerMiscellaneous packet with a selected spell ID to a player.

Parameters

pid – The player ID.

Returns

void

static void **SendAlliedPlayers** (unsigned short pid, bool sendToOtherPlayers)

Send a PlayerAlly packet with a list of team member IDs to a player.

Parameters

- **pid** The player ID.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).

Returns

void

static void **Jail** (unsigned short pid, int jailDays, bool ignoreJailTeleportation, bool ignoreJailSkillIncreases, const char *jailProgressText, const char *jailEndText) noexcept

Send a PlayerJail packet about a player.

This is similar to the player being jailed by a guard, but provides extra parameters for increased flexibility.

It is only sent to the player being jailed, as the other players will be informed of the jailing's actual consequences via other packets sent by the affected client.

Parameters

- **pid** The player ID.
- jailDays The number of days to spend jailed, where each day affects one skill point.
- **ignoreJailTeleportation** Whether the player being teleported to the nearest jail marker should be overridden.
- **ignoreJailSkillIncreases** Whether the player's Sneak and Security skills should be prevented from increasing as a result of the jailing, overriding default behavior.
- **jailProgressText** The text that should be displayed while jailed.
- **jailEndText** The text that should be displayed once the jailing period is over.

Returns

void

static void Resurrect (unsigned short pid, unsigned int type) noexcept

Send a PlayerResurrect packet about a player.

This sends the packet to all players connected to the server.

Parameters

- **pid** The player ID.
- type The type of resurrection (0 for REGULAR, 1 for IMPERIAL_SHRINE, 2 for TRIBUNAL TEMPLE).

void

1.11 Miscellaneous functions

class MiscellaneousFunctions

Public Static Functions

static const char *GenerateRandomString(unsigned int length) noexcept

Generate a random string of a particular length that only contains letters and numbers.

Parameters

length – The length of the generated string.

Returns

The generated string.

static const char *GetSHA256Hash(const char *inputString) noexcept

Get the SHA256 hash corresponding to an input string.

function is not reentrant due to a static variable

Parameters

inputString – The input string.

Returns

The SHA256 hash.

static unsigned int GetLastPlayerId() noexcept

Get the last player ID currently connected to the server.

function is not reentrant due to a static variable

Every player receives a unique numerical index known as their player ID upon joining the server.

Returns

The player ID.

static int GetCurrentMpNum() noexcept

Get the current (latest) mpNum generated by the server.

Every object that did not exist in an .ESM or .ESP data file and has instead been placed or spawned through a server-sent packet has a numerical index known as its mpNum.

When ObjectPlace and ObjectSpawn packets are received from players, their objects lack mpNums, so the server assigns them some based on incrementing the server's current mpNum, with the operation's final mpNum becoming the server's new current mpNum.

Returns

The mpNum.

static void **SetCurrentMpNum**(int mpNum) noexcept

Set the current (latest) mpNum generated by the server.

When restarting a server, it is important to revert to the previous current (latest) mpNum as stored in the server's data, so as to avoid starting over from 0 and ending up assigning duplicate mpNums to objects.

Parameters

mpNum – The number that should be used as the new current mpNum.

Returns

void

1.12 Object functions

class ObjectFunctions

Public Static Functions

static void ReadReceivedObjectList() noexcept

Use the last object list received by the server as the one being read.

Returns

void

static void ClearObjectList() noexcept

Clear the data from the object list stored on the server.

Returns

void

static void **SetObjectListPid**(unsigned short pid) noexcept

Set the pid attached to the ObjectList.

Parameters

pid – The player ID to whom the object list should be attached.

Returns

void

static void CopyReceivedObjectListToStore() noexcept

Take the contents of the read-only object list last received by the server from a player and move its contents to the stored object list that can be sent by the server.

Returns

void

static unsigned int GetObjectListSize() noexcept

Get the number of indexes in the read object list.

Returns

The number of indexes.

static unsigned char GetObjectListOrigin() noexcept

Get the origin of the read object list.

The origin (0 for CLIENT_GAMEPLAY, 1 for CLIENT_CONSOLE, 2 for CLIENT_DIALOGUE, 3 for CLIENT_SCRIPT_LOCAL, 4 for CLIENT_SCRIPT_GLOBAL, 5 for SERVER_SCRIPT).

static const char *GetObjectListClientScript() noexcept

Get the client script that the read object list originated from.

Returns

The ID of the client script.

static unsigned char GetObjectListAction() noexcept

Get the action type used in the read object list.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE, 3 for REQUEST).

static const char *GetObjectListConsoleCommand() noexcept

Get the console command used in the read object list.

Returns

The console command.

$static\ unsigned\ char\ \textbf{GetObjectListContainerSubAction}()\ no except$

Get the container subaction type used in the read object list.

Returns

The action type (0 for NONE, 1 for DRAG, 2 for DROP, 3 for TAKE_ALL).

static bool IsObjectPlayer (unsigned int index) noexcept

Check whether the object at a certain index in the read object list is a player.

Note: Although most player data and events are dealt with in Player packets, object activation is general enough for players themselves to be included as objects in ObjectActivate packets.

Parameters

index – The index of the object.

Returns

Whether the object is a player.

static int GetObjectPid(unsigned int index) noexcept

Get the player ID of the object at a certain index in the read object list, only valid if the object is a player.

Note: Currently, players can only be objects in ObjectActivate and ConsoleCommand packets.

Parameters

index – The index of the object.

Returns

The player ID of the object.

static const char *GetObjectRefId(unsigned int index) noexcept

Get the refId of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refId.

static unsigned int GetObjectRefNum(unsigned int index) noexcept

Get the refNum of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refNum.

static unsigned int GetObjectMpNum(unsigned int index) noexcept

Get the mpNum of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The mpNum.

static int GetObjectCount(unsigned int index) noexcept

Get the count of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The object count.

static int GetObjectCharge(unsigned int index) noexcept

Get the charge of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The charge.

static double GetObjectEnchantmentCharge (unsigned int index) noexcept

Get the enchantment charge of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The enchantment charge.

static const char *GetObjectSoul(unsigned int index) noexcept

Get the soul of the object at a certain index in the read object list.

Parameters

index - The index of the object.

Returns

The soul.

static int GetObjectGoldValue(unsigned int index) noexcept

Get the gold value of the object at a certain index in the read object list.

This is used solely to get the gold value of gold. It is not used for other objects.

Parameters

index – The index of the object.

The gold value.

static double **GetObjectScale**(unsigned int index) noexcept

Get the object scale of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The object scale.

static const char *GetObjectSoundId (unsigned int index) noexcept

Get the object sound ID of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The object sound ID.

static bool **GetObjectState**(unsigned int index) noexcept

Get the object state of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The object state.

static int GetObjectDoorState(unsigned int index) noexcept

Get the door state of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The door state.

static int GetObjectLockLevel (unsigned int index) noexcept

Get the lock level of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The lock level.

static unsigned int GetObjectDialogueChoiceType(unsigned int index) noexcept

Get the dialogue choice type for the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The dialogue choice type.

static const char *GetObjectDialogueChoiceTopic(unsigned int index) noexcept

Get the dialogue choice topic for the object at a certain index in the read object list.

Parameters

index - The index of the object.

The dialogue choice topic.

static unsigned int **GetObjectGoldPool** (unsigned int index) noexcept

Get the gold pool of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The gold pool.

static double GetObjectLastGoldRestockHour(unsigned int index) noexcept

Get the hour of the last gold restock of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The hour of the last gold restock.

static int GetObjectLastGoldRestockDay(unsigned int index) noexcept

Get the day of the last gold restock of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The day of the last gold restock.

static bool DoesObjectHavePlayerActivating(unsigned int index) noexcept

Check whether the object at a certain index in the read object list has been activated by a player.

Parameters

index – The index of the object.

Returns

Whether the object has been activated by a player.

static int GetObjectActivatingPid(unsigned int index) noexcept

Get the player ID of the player activating the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The player ID of the activating player.

 $static\ const\ char\ *\textbf{Get0bjectActivatingRefId} (unsigned\ int\ index)\ no except$

Get the refId of the actor activating the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refld of the activating actor.

static unsigned int GetObjectActivatingRefNum(unsigned int index) noexcept

Get the refNum of the actor activating the object at a certain index in the read object list.

Parameters

index - The index of the object.

The refNum of the activating actor.

static unsigned int GetObjectActivatingMpNum (unsigned int index) noexcept

Get the mpNum of the actor activating the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The mpNum of the activating actor.

static const char *GetObjectActivatingName(unsigned int index) noexcept

Get the name of the actor activating the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The name of the activating actor.

static bool **GetObjectHitSuccess** (unsigned int index) noexcept

Check whether the object at a certain index in the read object list has been hit successfully.

Parameters

index – The index of the object.

Returns

The success state.

static double GetObjectHitDamage(unsigned int index) noexcept

Get the damage caused to the object at a certain index in the read object list in a hit.

Parameters

index – The index of the object.

Returns

The damage.

static bool GetObjectHitBlock (unsigned int index) noexcept

Check whether the object at a certain index in the read object list has blocked the hit on it.

Parameters

index – The index of the object.

Returns

The block state.

static bool GetObjectHitKnockdown (unsigned int index) noexcept

Check whether the object at a certain index in the read object list has been knocked down.

Parameters

index – The index of the object.

Returns

The knockdown state.

static bool DoesObjectHavePlayerHitting(unsigned int index) noexcept

Check whether the object at a certain index in the read object list has been hit by a player.

Parameters

index - The index of the object.

Whether the object has been hit by a player.

static int GetObjectHittingPid(unsigned int index) noexcept

Get the player ID of the player hitting the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The player ID of the hitting player.

$static\ const\ char\ *\textbf{GetObjectHittingRefId} (unsigned\ int\ index)\ no except$

Get the refId of the actor hitting the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refId of the hitting actor.

static unsigned int GetObjectHittingRefNum(unsigned int index) noexcept

Get the refNum of the actor hitting the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refNum of the hitting actor.

static unsigned int GetObjectHittingMpNum(unsigned int index) noexcept

Get the mpNum of the actor hitting the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The mpNum of the hitting actor.

static const char *GetObjectHittingName(unsigned int index) noexcept

Get the name of the actor hitting the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The name of the hitting actor.

static bool GetObjectSummonState(unsigned int index) noexcept

Check whether the object at a certain index in the read object list is a summon.

Only living actors can be summoned.

Returns

The summon state.

static double **GetObjectSummonEffectId**(unsigned int index) noexcept

Get the summon effect ID of the object at a certain index in the read object list.

Parameters

index – The index of the object.

The summon effect ID.

static const char *GetObjectSummonSpellId(unsigned int index) noexcept

Get the summon spell ID of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The summon spell ID.

static double **GetObjectSummonDuration**(unsigned int index) noexcept

Get the summon duration of the object at a certain index in the read object list.

Note: Returns -1 if indefinite.

Parameters

index – The index of the object.

Returns

The summon duration.

static bool DoesObjectHavePlayerSummoner(unsigned int index) noexcept

Check whether the object at a certain index in the read object list has a player as its summoner.

Only living actors can be summoned.

Parameters

index – The index of the object.

Returns

Whether a player is the summoner of the object.

static int GetObjectSummonerPid(unsigned int index) noexcept

Get the player ID of the summoner of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The player ID of the summoner.

$static\ const\ char\ *\textbf{GetObjectSummonerRefId} (unsigned\ int\ index)\ no except$

Get the refId of the actor summoner of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refId of the summoner.

$static\ unsigned\ int\ \textbf{GetObjectSummonerRefNum} \ (unsigned\ int\ index)\ no except$

Get the refNum of the actor summoner of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The refNum of the summoner.

static unsigned int GetObjectSummonerMpNum (unsigned int index) noexcept

Get the mpNum of the actor summoner of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The mpNum of the summoner.

static double **GetObjectPosX** (unsigned int index) noexcept

Get the X position of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The X position.

static double GetObjectPosY(unsigned int index) noexcept

Get the Y position of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The Y position.

static double GetObjectPosZ(unsigned int index) noexcept

Get the Z position at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The Z position.

static double ${\tt GetObjectRotX}(unsigned\ int\ index)\ noexcept$

Get the X rotation of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The X rotation.

static double **GetObjectRotY**(unsigned int index) noexcept

Get the Y rotation of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The Y rotation.

static double GetObjectRotZ(unsigned int index) noexcept

Get the Z rotation of the object at a certain index in the read object list.

Parameters

index – The index of the object.

Returns

The Z rotation.

static const char *GetVideoFilename(unsigned int index) noexcept

Get the videoFilename of the object at a certain index in the read object list.

Returns

The videoFilename.

static unsigned int **GetClientLocalsSize** (unsigned int objectIndex) noexcept

Get the number of client local variables of the object at a certain index in the read object list.

Parameters

objectIndex – The index of the object.

Returns

The number of client local variables.

static unsigned int **GetClientLocalInternalIndex**(unsigned int objectIndex, unsigned int variableIndex) noexcept

Get the internal script index of the client local variable at a certain variableIndex in the client locals of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- variableIndex The index of the client local.

Returns

The internal script index.

static unsigned short **GetClientLocalVariableType** (unsigned int objectIndex, unsigned int variableIndex) noexcept

Get the type of the client local variable at a certain variableIndex in the client locals of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- variableIndex The index of the client local.

Returns

The variable type (0 for INTEGER, 1 for LONG, 2 for FLOAT).

static int **GetClientLocalIntValue**(unsigned int objectIndex, unsigned int variableIndex) noexcept

Get the integer value of the client local variable at a certain variableIndex in the client locals of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- variableIndex The index of the client local.

Returns

The integer value.

static double **GetClientLocalFloatValue**(unsigned int objectIndex, unsigned int variableIndex) noexcept

Get the float value of the client local variable at a certain variableIndex in the client locals of the object at a certain objectIndex in the read object list.

Parameters

• **objectIndex** – The index of the object.

• variableIndex – The index of the client local.

Returns

The float value.

static unsigned int GetContainerChangesSize(unsigned int objectIndex) noexcept

Get the number of container item indexes of the object at a certain index in the read object list.

Parameters

objectIndex – The index of the object.

Returns

The number of container item indexes.

static const char *GetContainerItemRefId(unsigned int objectIndex, unsigned int itemIndex) noexcept

Get the refId of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- itemIndex The index of the container item.

Returns

The refId.

static int **GetContainerItemCount** (unsigned int objectIndex, unsigned int itemIndex) noexcept

Get the item count of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- itemIndex The index of the container item.

Returns

The item count.

static int **GetContainerItemCharge**(unsigned int objectIndex, unsigned int itemIndex) noexcept

Get the charge of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- **itemIndex** The index of the container item.

Returns

The charge.

 $static\ double\ \textbf{GetContainerItemEnchantmentCharge} (unsigned\ int\ objectIndex,\ unsigned\ int\ itemIndex)\\ no except$

Get the enchantment charge of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- itemIndex The index of the container item.

The enchantment charge.

static const char *GetContainerItemSoul (unsigned int objectIndex, unsigned int itemIndex) noexcept

Get the soul of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- itemIndex The index of the container item.

Returns

The soul.

static int GetContainerItemActionCount (unsigned int objectIndex, unsigned int itemIndex) noexcept

Get the action count of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the read object list.

Parameters

- **objectIndex** The index of the object.
- **itemIndex** The index of the container item.

Returns

The action count.

static bool DoesObjectHaveContainer(unsigned int index) noexcept

Check whether the object at a certain index in the read object list has a container.

Note: Only ObjectLists from ObjectPlace packets contain this information. Objects from received ObjectSpawn packets can always be assumed to have a container.

Parameters

index – The index of the object.

Returns

Whether the object has a container.

static bool IsObjectDroppedByPlayer(unsigned int index) noexcept

Check whether the object at a certain index in the read object list has been dropped by a player.

Note: Only ObjectLists from ObjectPlace packets contain this information.

Parameters

index – The index of the object.

Returns

Whether the object has been dropped by a player.

static void **SetObjectListCell**(const char *cellDescription) noexcept

Set the cell of the temporary object list stored on the server.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

cellDescription – The description of the cell.

Returns

void

static void **SetObjectListAction**(unsigned char action) noexcept

Set the action type of the temporary object list stored on the server.

Parameters

action – The action type (0 for SET, 1 for ADD, 2 for REMOVE, 3 for REQUEST).

Returns

void

static void **SetObjectListContainerSubAction**(unsigned char subAction) noexcept

Set the container subaction type of the temporary object list stored on the server.

Parameters

subAction – The action type (0 for NONE, 1 for DRAG, 2 for DROP, 3 for TAKE_ALL, 4 for REPLY_TO_REQUEST, 5 for RESTOCK_RESULT).

Returns

void

$static\ void\ \textbf{SetObjectListConsoleCommand} (const\ char\ *consoleCommand)\ no except$

Set the console command of the temporary object list stored on the server.

When sent, the command will run once on every object added to the object list. If no objects have been added, it will run once without any object reference.

Parameters

consoleCommand – The console command.

Returns

void

static void **SetObjectRefId**(const char *refId) noexcept

Set the refId of the temporary object stored on the server.

Parameters

refId - The refId.

Returns

void

static void **SetObjectRefNum**(int refNum) noexcept

Set the refNum of the temporary object stored on the server.

Every object loaded from .ESM and .ESP data files has a unique refNum which needs to be retained to refer to it in packets.

On the other hand, objects placed or spawned via the server should always have a refNum of 0.

Parameters

refNum – The refNum.

Returns

void

static void **SetObjectMpNum**(int mpNum) noexcept

Set the mpNum of the temporary object stored on the server.

Every object placed or spawned via the server is assigned an mpNum by incrementing the last mpNum stored on the server. Scripts should take care to ensure that mpNums are kept unique for these objects.

Objects loaded from .ESM and .ESP data files should always have an mpNum of 0, because they have unique refNumes instead.

Parameters

mpNum – The mpNum.

Returns

void

static void **SetObjectCount** (int count) noexcept

Set the object count of the temporary object stored on the server.

This determines the quantity of an object, with the exception of gold.

Parameters

count – The object count.

Returns

void

static void SetObjectCharge(int charge) noexcept

Set the charge of the temporary object stored on the server.

Object durabilities are set through this value.

Parameters

charge – The charge.

Returns

void

$static\ void\ \textbf{SetObjectEnchantmentCharge} (double\ enchantmentCharge)\ no except$

Set the enchantment charge of the temporary object stored on the server.

Object durabilities are set through this value.

Parameters

enchantmentCharge – The enchantment charge.

Returns

void

static void **SetObjectSoul** (const char *soul) noexcept

Set the soul of the temporary object stored on the server.

Parameters

soul – The ID of the soul.

Returns

void

static void SetObjectGoldValue(int goldValue) noexcept

Set the gold value of the temporary object stored on the server.

This is used solely to set the gold value for gold. It has no effect on other objects.

Parameters

goldValue – The gold value.

Returns

void

static void SetObjectScale(double scale) noexcept

Set the scale of the temporary object stored on the server.

Objects are smaller or larger than their default size based on their scale.

Parameters

scale – The scale.

Returns

void

static void SetObjectState(bool objectState) noexcept

Set the object state of the temporary object stored on the server.

Objects are enabled or disabled based on their object state.

Parameters

objectState – The object state.

Returns

void

static void SetObjectLockLevel(int lockLevel) noexcept

Set the lock level of the temporary object stored on the server.

Parameters

lockLevel – The lock level.

Returns

void

static void **SetObjectDialogueChoiceType**(unsigned int dialogueChoiceType) noexcept

Set the dialogue choice type of the temporary object stored on the server.

Parameters

dialogueChoiceType – The dialogue choice type.

Returns

void

$static\ void\ \textbf{SetObjectDialogueChoiceTopic} (const\ char\ *topic)\ noexcept$

Set the dialogue choice topic for the temporary object stored on the server.

Parameters

topic – The dialogue choice topic.

Returns

void

static void **SetObjectGoldPool** (unsigned int goldPool) noexcept

Set the gold pool of the temporary object stored on the server.

Parameters

goldPool – The gold pool.

Returns

void

static void SetObjectLastGoldRestockHour(double hour) noexcept

Set the hour of the last gold restock of the temporary object stored on the server.

Parameters

hour – The hour of the last gold restock.

Returns

void

static void SetObjectLastGoldRestockDay(int day) noexcept

Set the day of the last gold restock of the temporary object stored on the server.

Parameters

day – The day of the last gold restock.

Returns

void

static void SetObjectDisarmState(bool disarmState) noexcept

Set the disarm state of the temporary object stored on the server.

Parameters

disarmState - The disarmState.

Returns

void

static void SetObjectDroppedByPlayerState(bool dropedByPlayerState) noexcept

Set the droppedByPlayer state of the temporary object stored on the server.

Parameters

dropedByPlayerState – Whether the object has been dropped by a player or not.

Returns

void

static void **SetObjectPosition**(double x, double y, double z) noexcept

Set the position of the temporary object stored on the server.

Parameters

- **x** The X position.
- **y** The Y position.
- **z** The Z position.

Returns

void

static void ${f SetObjectRotation}(double\ x,\ double\ y,\ double\ z)$ noexcept

Set the rotation of the temporary object stored on the server.

Parameters

- **x** The X rotation.
- y The Y rotation.
- **z** The Z rotation.

Returns

void

static void SetObjectSummonState(bool summonState) noexcept

Set the summon state of the temporary object stored on the server.

This only affects living actors and determines whether they are summons of another living actor.

Parameters

summonState – The summon state.

Returns

void

static void SetObjectSummonEffectId(int summonEffectId) noexcept

Set the summon effect ID of the temporary object stored on the server.

Parameters

summonEffectId – The summon effect ID.

Returns

void

static void **SetObjectSummonSpellId**(const char *summonSpellId) noexcept

Set the summon spell ID of the temporary object stored on the server.

Parameters

summonSpellId – The summon spell ID.

Returns

void

static void **SetObjectSummonDuration**(double summonDuration) noexcept

Set the summon duration of the temporary object stored on the server.

Parameters

summonDuration – The summon duration.

Returns

void

static void **SetObjectSummonerPid**(unsigned short pid) noexcept

Set the player ID of the summoner of the temporary object stored on the server.

Parameters

pid – The player ID of the summoner.

Returns

void

static void SetObjectSummonerRefNum(int refNum) noexcept

Set the refNum of the actor summoner of the temporary object stored on the server.

Parameters

refNum – The refNum of the summoner.

Returns

void

static void **SetObjectSummonerMpNum**(int mpNum) noexcept

Set the mpNum of the actor summoner of the temporary object stored on the server.

Parameters

mpNum – The mpNum of the summoner.

Returns

void

static void SetObjectActivatingPid(unsigned short pid) noexcept

Set the player ID of the player activating the temporary object stored on the server. Currently only used for ObjectActivate packets.

Parameters

pid – The pid of the player.

void

static void **SetObjectDoorState**(int doorState) noexcept

Set the door state of the temporary object stored on the server.

Doors are open or closed based on their door state.

Parameters

doorState – The door state.

Returns

void

static void **SetObjectDoorTeleportState**(bool teleportState) noexcept

Set the teleport state of the temporary object stored on the server.

If a door's teleport state is true, interacting with the door teleports a player to its destination. If it's false, it opens and closes like a regular door.

Parameters

teleportState - The teleport state.

Returns

void

static void SetObjectDoorDestinationCell(const char *cellDescription) noexcept

Set the door destination cell of the temporary object stored on the server.

The cell is determined to be an exterior cell if it fits the pattern of a number followed by a comma followed by another number.

Parameters

cellDescription – The description of the cell.

Returns

void

static void **SetObjectDoorDestinationPosition**(double x, double y, double z) noexcept

Set the door destination position of the temporary object stored on the server.

Parameters

- \mathbf{x} The X position.
- **y** The Y position.
- **z** The Z position.

Returns

void

$static\ void\ \textbf{SetObjectDoorDestinationRotation}(double\ x,\ double\ z)\ noexcept$

Set the door destination rotation of the temporary object stored on the server.

Note: Because this sets the rotation a player will have upon using the door, and rotation on the Y axis has no effect on players, the Y value has been omitted as an argument.

Parameters

- \mathbf{x} The X rotation.
- **z** The Z rotation.

void

static void SetPlayerAsObject(unsigned short pid) noexcept

Set a player as the object in the temporary object stored on the server. Currently only used for ConsoleCommand packets.

Parameters

pid – The pid of the player.

Returns

void

static void **SetContainerItemRefId**(const char *refId) noexcept

Set the refId of the temporary container item stored on the server.

Parameters

refId - The refId.

Returns

void

static void **SetContainerItemCount** (int count) noexcept

Set the item count of the temporary container item stored on the server.

Parameters

count - The item count.

Returns

void

static void **SetContainerItemCharge**(int charge) noexcept

Set the charge of the temporary container item stored on the server.

Parameters

charge – The charge.

Returns

void

static void SetContainerItemEnchantmentCharge (double enchantmentCharge) noexcept

Set the enchantment charge of the temporary container item stored on the server.

Parameters

enchantmentCharge – The enchantment charge.

Returns

void

static void **SetContainerItemSoul**(const char *soul) noexcept

Set the soul of the temporary container item stored on the server.

Parameters

soul – The soul.

Returns

void

 $static\ void\ \textbf{SetContainerItemActionCountByIndex} (unsigned\ int\ objectIndex,\ unsigned\ int\ itemIndex,\ int\ actionCount)\ noexcept$

Set the action count of the container item at a certain itemIndex in the container changes of the object at a certain objectIndex in the object list stored on the server.

When resending a received Container packet, this allows you to correct the amount of items removed from a container by a player when it conflicts with what other players have already taken.

Parameters

- **objectIndex** The index of the object.
- itemIndex The index of the container item.
- actionCount The action count.

Returns

void

static void AddObject() noexcept

Add a copy of the server's temporary object to the server's currently stored object list.

In the process, the server's temporary object will automatically be cleared so a new one can be set up.

Returns

void

static void **AddClientLocalInteger**(int internalIndex, int intValue, unsigned int variableType) noexcept Add a client local variable with an integer value to the client locals of the server's temporary object.

Parameters

- **internalIndex** The internal script index of the client local.
- variableType The variable type (0 for SHORT, 1 for LONG).
- **intValue** The integer value of the client local.

Returns

void

 $static\ void\ \textbf{AddClientLocalFloat} (int\ internalIndex,\ double\ floatValue)\ no except$

Add a client local variable with a float value to the client locals of the server's temporary object.

Parameters

- internalIndex The internal script index of the client local.
- **floatValue** The float value of the client local.

Returns

void

static void **AddContainerItem()** noexcept

Add a copy of the server's temporary container item to the container changes of the server's temporary object.

In the process, the server's temporary container item will automatically be cleared so a new one can be set up.

Returns

void

static void **SendObjectActivate**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectActivate packet.

Parameters

• **sendToOtherPlayers** – Whether this packet should be sent to players other than the player attached to the packet (false by default).

• **skipAttachedPlayer** – Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectPlace**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectPlace packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectSpawn**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectSpawn packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectDelete**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectDelete packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectLock**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectLock packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectDialogueChoice**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectDialogueChoice packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectMiscellaneous**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectMiscellaneous packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectRestock**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectRestock packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectTrap**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectTrap packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectScale**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectScale packet.

Parameters

• **sendToOtherPlayers** – Whether this packet should be sent to players other than the player attached to the packet (false by default).

• **skipAttachedPlayer** – Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectSound**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectSound packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectState**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectState packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectMove**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectMove packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendObjectRotate**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send an ObjectRotate packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendDoorState**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a DoorState packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendDoorDestination**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a DoorDestination packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendContainer**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a Container packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendVideoPlay**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a VideoPlay packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendClientScriptLocal** (bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a ClientScriptLocal packet.

Parameters

• **sendToOtherPlayers** – Whether this packet should be sent to players other than the player attached to the packet (false by default).

• **skipAttachedPlayer** – Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendConsoleCommand**(bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a ConsoleCommand packet.

Parameters

- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.13 Position functions

class PositionFunctions

Public Static Functions

static double ${\tt GetPosX}(unsigned\ short\ pid)\ no except$

Get the X position of a player.

Parameters

pid – The player ID.

Returns

The X position.

static double **GetPosY**(unsigned short pid) noexcept

Get the Y position of a player.

Parameters

pid – The player ID.

Returns

The Y position.

static double **GetPosZ**(unsigned short pid) noexcept

Get the Z position of a player.

Parameters

pid – The player ID.

Returns

The Z position.

static double **GetPreviousCellPosX** (unsigned short pid) noexcept

Get the X position of a player from before their latest cell change.

Parameters

pid – The player ID.

The X position.

static double **GetPreviousCellPosY** (unsigned short pid) noexcept

Get the Y position of a player from before their latest cell change.

Parameters

pid - The player ID.

Returns

The Y position.

static double **GetPreviousCellPosZ** (unsigned short pid) noexcept

Get the Z position of a player from before their latest cell change.

Parameters

pid - The player ID.

Returns

The Z position.

static double **GetRotX** (unsigned short pid) noexcept

Get the X rotation of a player.

Parameters

pid – The player ID.

Returns

The X rotation.

static double **GetRotZ**(unsigned short pid) noexcept

Get the Z rotation of a player.

Parameters

pid – The player ID.

Returns

The Z rotation.

static void **SetPos** (unsigned short pid, double x, double y, double z) noexcept

Set the position of a player.

This changes the positional coordinates recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **x** The X position.
- y The Y position.
- z The Z position.

Returns

void

static void **SetRot** (unsigned short pid, double x, double z) noexcept

Set the rotation of a player.

This changes the rotational coordinates recorded for that player in the server memory, but does not by itself send a packet.

A player's Y rotation is always 0, which is why there is no Y rotation parameter.

Parameters

- pid The player ID.
- \mathbf{x} The X position.
- **z** The Z position.

Returns

void

static void **SetMomentum** (unsigned short pid, double x, double y, double z) noexcept

Set the momentum of a player.

This changes the coordinates recorded for that player's momentum in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- \mathbf{x} The X momentum.
- **y** The Y momentum.
- **z** The Z momentum.

Returns

void

static void **SendPos** (unsigned short pid) noexcept

Send a PlayerPosition packet about a player.

It is only sent to the affected player.

Parameters

pid – The player ID.

Returns

void

static void SendMomentum (unsigned short pid) noexcept

Send a PlayerMomentum packet about a player.

It is only sent to the affected player.

Parameters

pid – The player ID.

Returns

1.14 Quest functions

class QuestFunctions

Public Static Functions

static void ClearJournalChanges (unsigned short pid) noexcept

Clear the last recorded journal changes for a player.

This is used to initialize the sending of new PlayerJournal packets.

Parameters

pid – The player ID whose journal changes should be used.

Returns

void

static unsigned int GetJournalChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest journal changes.

Parameters

pid – The player ID whose journal changes should be used.

Returns

The number of indexes.

static void **AddJournalEntry** (unsigned short pid, const char *quest, unsigned int index, const char *actorRefId) noexcept

Add a new journal item of type ENTRY to the journal changes for a player, with a specific timestamp.

Parameters

- pid The player ID whose journal changes should be used.
- **quest** The quest of the journal item.
- **index** The quest index of the journal item.
- actorRefId The actor refId of the journal item.

Returns

void

static void **AddJournalEntryWithTimestamp** (unsigned short pid, const char *quest, unsigned int index, const char *actorRefId, unsigned int daysPassed, unsigned int month, unsigned int day) noexcept

Add a new journal item of type ENTRY to the journal changes for a player, with a specific timestamp.

Parameters

- **pid** The player ID whose journal changes should be used.
- **quest** The quest of the journal item.
- index The quest index of the journal item.
- actorRefId The actor refId of the journal item.
- daysPassed The daysPassed for the journal item.
- **month** The month for the journal item.

1.14. Quest functions 73

• day – The day of the month for the journal item.

Returns

void

static void **AddJournalIndex** (unsigned short pid, const char *quest, unsigned int index) noexcept Add a new journal item of type INDEX to the journal changes for a player.

Parameters

- pid The player ID whose journal changes should be used.
- **quest** The quest of the journal item.
- index The quest index of the journal item.

Returns

void

 $static\ void\ \textbf{SetReputation} (unsigned\ short\ pid,\ int\ value)\ no except$

Set the reputation of a certain player.

Parameters

- **pid** The player ID.
- **value** The reputation.

Returns

void

static const char *GetJournalItemQuest(unsigned short pid, unsigned int index) noexcept Get the quest at a certain index in a player's latest journal changes.

Parameters

- pid The player ID whose journal changes should be used.
- **index** The index of the journalItem.

Returns

The quest.

static int **GetJournalItemIndex** (unsigned short pid, unsigned int index) noexcept Get the quest index at a certain index in a player's latest journal changes.

Parameters

- pid The player ID whose journal changes should be used.
- **index** The index of the journalItem.

Returns

The quest index.

static int **GetJournalItemType** (unsigned short pid, unsigned int index) noexcept

Get the journal item type at a certain index in a player's latest journal changes.

Parameters

- **pid** The player ID whose journal changes should be used.
- **index** The index of the journalItem.

Returns

The type (0 for ENTRY, 1 for INDEX).

static const char *GetJournalItemActorRefId(unsigned short pid, unsigned int index) noexcept

Get the actor refId at a certain index in a player's latest journal changes.

Every journal change has an associated actor, which is usually the quest giver.

Parameters

- pid The player ID whose journal changes should be used.
- **index** The index of the journalItem.

Returns

The actor refId.

static int GetReputation (unsigned short pid) noexcept

Get the a certain player's reputation.

Parameters

pid – The player ID.

Returns

The reputation.

static void **SendJournalChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerJournal packet with a player's recorded journal changes.

Parameters

- **pid** The player ID whose journal changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendReputation**(unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerReputation packet with a player's recorded reputation.

Parameters

- **pid** The player ID whose reputation should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

1.14. Quest functions 75

1.15 Records Dynamic functions

class RecordsDynamicFunctions

Public Static Functions

static void ClearRecords() noexcept

Clear the data from the records stored on the server.

Returns

void

static unsigned short **GetRecordType()** noexcept

Get the type of records in the read worldstate's dynamic records.

Returns

The type of records (0 for SPELL, 1 for POTION, 2 for ENCHANTMENT, 3 for NPC).

static unsigned int **GetRecordCount()** noexcept

Get the number of records in the read worldstate's dynamic records.

Returns

The number of records.

static unsigned int **GetRecordEffectCount** (unsigned int recordIndex) noexcept

Get the number of effects for the record at a certain index in the read worldstate's current records.

Parameters

recordIndex – The index of the record.

Returns

The number of effects.

static const char *GetRecordId (unsigned int index) noexcept

Get the id of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The id of the record.

 $static\ const\ char\ *\textbf{GetRecordBaseId} (unsigned\ int\ index)\ no except$

Get the base id (i.e. the id this record should inherit default values from) of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The base id of the record.

static int **GetRecordSubtype** (unsigned int index) noexcept

Get the subtype of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

The type of the record.

static const char *GetRecordName(unsigned int index) noexcept

Get the name of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The name of the record.

static const char *GetRecordModel (unsigned int index) noexcept

Get the model of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index - The index of the record.

Returns

The model of the record.

static const char *GetRecordIcon(unsigned int index) noexcept

Get the icon of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The icon of the record.

static const char *GetRecordScript(unsigned int index) noexcept

Get the script of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index - The index of the record.

Returns

The script of the record.

static const char *GetRecordEnchantmentId(unsigned int index) noexcept

Get the enchantment id of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The enchantment id of the record.

static int GetRecordEnchantmentCharge (unsigned int index) noexcept

Get the enchantment charge of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The enchantment charge of the record.

static int GetRecordAutoCalc(unsigned int index) noexcept

Get the auto-calculation flag value of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The auto-calculation flag value of the record.

static int GetRecordCharge (unsigned int index) noexcept

Get the charge of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The charge of the record.

static int **GetRecordCost** (unsigned int index) noexcept

Get the cost of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The cost of the record.

static int GetRecordFlags (unsigned int index) noexcept

Get the flags of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The flags of the spell as an integer.

static int **GetRecordValue**(unsigned int index) noexcept

Get the value of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The value of the record.

static double **GetRecordWeight** (unsigned int index) noexcept

Get the weight of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The weight of the record.

static unsigned int **GetRecordQuantity**(unsigned int index) noexcept

Get the quantity of the record at a certain index in the read worldstate's dynamic records of the current type.

Parameters

index – The index of the record.

Returns

The brewed count of the record.

static unsigned int **GetRecordEffectId**(unsigned int recordIndex, unsigned int effectIndex) noexcept Get the ID of the effect at a certain index in the read worldstate's current records.

Parameters

- **recordIndex** The index of the record.
- **effectIndex** The index of the effect.

Returns

The ID of the effect.

static int **GetRecordEffectAttribute**(unsigned int recordIndex, unsigned int effectIndex) noexcept

Get the ID of the attribute modified by the effect at a certain index in the read worldstate's current records.

Parameters

- recordIndex The index of the record.
- **effectIndex** The index of the effect.

Returns

The attribute ID for the effect.

static int **GetRecordEffectSkill** (unsigned int recordIndex, unsigned int effectIndex) noexcept

Get the ID of the skill modified by the effect at a certain index in the read worldstate's current records.

Parameters

- recordIndex The index of the record.
- **effectIndex** The index of the effect.

Returns

The skill ID for the effect.

 $static \ unsigned \ int \ \textbf{GetRecordEffectRangeType} (unsigned \ int \ recordIndex, \ unsigned \ int \ effectIndex) \\ no except$

Get the range type of the effect at a certain index in the read worldstate's current records (0 for self, 1 for touch, 2 for target).

Parameters

- **recordIndex** The index of the record.
- **effectIndex** The index of the effect.

Returns

The range of the effect.

static int **GetRecordEffectArea** (unsigned int recordIndex, unsigned int effectIndex) noexcept

Get the area of the effect at a certain index in the read worldstate's current records.

Parameters

- recordIndex The index of the record.
- **effectIndex** The index of the effect.

Returns

The area of the effect.

static int **GetRecordEffectDuration** (unsigned int recordIndex, unsigned int effectIndex) noexcept Get the duration of the effect at a certain index in the read worldstate's current records.

Parameters

- **recordIndex** The index of the record.
- **effectIndex** The index of the effect.

Returns

The duration of the effect.

static int **GetRecordEffectMagnitudeMax** (unsigned int recordIndex, unsigned int effectIndex) noexcept Get the maximum magnitude of the effect at a certain index in the read worldstate's current records.

Parameters

- recordIndex The index of the record.
- **effectIndex** The index of the effect.

Returns

The maximum magnitude of the effect.

static int **GetRecordEffectMagnitudeMin**(unsigned int recordIndex, unsigned int effectIndex) noexcept

Get the minimum magnitude of the effect at a certain index in the read worldstate's current records.

Parameters

- recordIndex The index of the record.
- **effectIndex** The index of the effect.

Returns

The minimum magnitude of the effect.

static void **SetRecordType** (unsigned int type) noexcept

Set which type of temporary records stored on the server should have their data changed via setter functions.

Parameters

type – The type of records.

Returns

void

static void **SetRecordId**(const char *id) noexcept

Set the id of the temporary record stored on the server for the currently specified record type.

Parameters

id – The id of the record.

Returns

void

static void ${f SetRecordBaseId}$ (const char *baseId) noexcept

Set the base id (i.e. the id this record should inherit default values from) of the temporary record stored on the server for the currently specified record type.

Parameters

baseId – The baseId of the record.

Returns

static void **SetRecordInventoryBaseId**(const char *inventoryBaseId) noexcept

Set the inventory base id (i.e. the id this record should inherit its inventory contents from) of the temporary record stored on the server for the currently specified record type.

Parameters

inventoryBaseId – The inventoryBaseId of the record.

Returns

void

static void **SetRecordSubtype** (unsigned int subtype) noexcept

Set the subtype of the temporary record stored on the server for the currently specified record type.

Parameters

subtype – The spell type.

Returns

void

static void SetRecordName (const char *name) noexcept

Set the name of the temporary record stored on the server for the currently specified record type.

Parameters

name – The name of the record.

Returns

void

static void **SetRecordModel** (const char *model) noexcept

Set the model of the temporary record stored on the server for the currently specified record type.

Parameters

model – The model of the record.

Returns

void

static void **SetRecordIcon**(const char *icon) noexcept

Set the icon of the temporary record stored on the server for the currently specified record type.

Parameters

icon – The icon of the record.

Returns

void

static void **SetRecordScript** (const char *script) noexcept

Set the script of the temporary record stored on the server for the currently specified record type.

Parameters

script – The script of the record.

Returns

void

static void **SetRecordEnchantmentId**(const char *enchantmentId) noexcept

Set the enchantment id of the temporary record stored on the server for the currently specified record type.

Parameters

enchantmentId – The enchantment id of the record.

void

static void SetRecordEnchantmentCharge(int enchantmentCharge) noexcept

Set the enchantment charge of the temporary record stored on the server for the currently specified record type.

Parameters

enchantmentCharge – The enchantmentCharge of the record.

Returns

void

static void SetRecordAutoCalc(int autoCalc) noexcept

Set the auto-calculation flag value of the temporary record stored on the server for the currently specified record type.

Parameters

autoCalc – The auto-calculation flag value of the record.

Returns

void

static void **SetRecordCharge**(int charge) noexcept

Set the charge of the temporary record stored on the server for the currently specified record type.

Parameters

charge – The charge of the record.

Returns

void

static void **SetRecordCost**(int cost) noexcept

Set the cost of the temporary record stored on the server for the currently specified record type.

Parameters

cost – The cost of the record.

Returns

void

static void **SetRecordFlags**(int flags) noexcept

Set the flags of the temporary record stored on the server for the currently specified record type.

Parameters

flags – The flags of the record.

Returns

void

static void **SetRecordValue**(int value) noexcept

Set the value of the temporary record stored on the server for the currently specified record type.

Parameters

value – The value of the record.

Returns

voic

static void **SetRecordWeight** (double weight) noexcept

Set the weight of the temporary record stored on the server for the currently specified record type.

Parameters

weight – The weight of the record.

Returns

void

static void **SetRecordQuality**(double quality) noexcept

Set the item quality of the temporary record stored on the server for the currently specified record type.

Parameters

quality – The quality of the record.

Returns

void

static void **SetRecordUses**(int uses) noexcept

Set the number of uses of the temporary record stored on the server for the currently specified record type.

Parameters

uses – The number of uses of the record.

Returns

void

static void **SetRecordTime**(int time) noexcept

Set the time of the temporary record stored on the server for the currently specified record type.

Parameters

time – The time of the record.

Returns

void

static void ${f SetRecordRadius}$ (int radius) no except

Set the radius of the temporary record stored on the server for the currently specified record type.

Parameters

radius – The radius of the record.

Returns

void

static void SetRecordColor (unsigned int red, unsigned int green, unsigned int blue) noexcept

Set the color of the temporary record stored on the server for the currently specified record type.

Parameters

- **red** The red value of the record.
- **green** The green value of the record.
- blue The blue value of the record.

Returns

void

static void **SetRecordArmorRating**(int armorRating) noexcept

Set the armor rating of the temporary record stored on the server for the currently specified record type.

Parameters

armorRating – The armor rating of the record.

Returns

static void **SetRecordHealth**(int health) noexcept

Set the health of the temporary record stored on the server for the currently specified record type.

Parameters

health – The health of the record.

Returns

void

static void **SetRecordDamageChop** (unsigned int minDamage, unsigned int maxDamage) noexcept

Set the chop damage of the temporary record stored on the server for the currently specified record type.

Parameters

- minDamage The minimum damage of the record.
- **maxDamage** The maximum damage of the record.

Returns

void

static void **SetRecordDamageSlash**(unsigned int minDamage, unsigned int maxDamage) noexcept

Set the slash damage of the temporary record stored on the server for the currently specified record type.

Parameters

- minDamage The minimum damage of the record.
- maxDamage The maximum damage of the record.

Returns

void

static void **SetRecordDamageThrust** (unsigned int minDamage, unsigned int maxDamage) noexcept

Set the thrust damage of the temporary record stored on the server for the currently specified record type.

Parameters

- minDamage The minimum damage of the record.
- maxDamage The maximum damage of the record.

Returns

void

static void SetRecordReach (double reach) noexcept

Set the reach of the temporary record stored on the server for the currently specified record type.

Parameters

reach – The reach of the record.

Returns

void

static void **SetRecordSpeed**(double speed) noexcept

Set the speed of the temporary record stored on the server for the currently specified record type.

Parameters

speed – The speed of the record.

Returns

static void **SetRecordKeyState**(bool keyState) noexcept

Set whether the temporary record stored on the server for the currently specified record type is a key.

Note: This is only applicable to Miscellaneous records.

Parameters

keyState – Whether the record is a key.

Returns

void

static void **SetRecordScrollState**(bool scrollState) noexcept

Set whether the temporary record stored on the server for the currently specified record type is a scroll.

Note: This is only applicable to Book records.

Parameters

scrollState – Whether the record is a scroll.

Returns

void

static void SetRecordSkillId(int skillId) noexcept

Set the skill ID of the temporary record stored on the server for the currently specified record type.

Parameters

skillId – The skill ID of the record.

Returns

void

static void **SetRecordText**(const char *text) noexcept

Set the text of the temporary record stored on the server for the currently specified record type.

Parameters

text – The text of the record.

Returns

void

static void **SetRecordHair**(const char *hair) noexcept

Set the hair of the temporary record stored on the server for the currently specified record type.

Parameters

hair – The hair of the record.

Returns

void

static void **SetRecordHead**(const char *head) noexcept

Set the head of the temporary record stored on the server for the currently specified record type.

Parameters

head – The head of the record.

Returns

void

static void SetRecordGender (unsigned int gender) noexcept

Set the gender of the temporary record stored on the server for the currently specified record type (0 for female, 1 for male).

Parameters

gender – The gender of the record.

Returns

void

static void **SetRecordRace**(const char *race) noexcept

Set the race of the temporary record stored on the server for the currently specified record type.

Parameters

race – The race of the record.

Returns

void

static void **SetRecordClass**(const char *charClass) noexcept

Set the character class of the temporary record stored on the server for the currently specified record type.

Parameters

charClass – The character class of the record.

Returns

void

static void **SetRecordFaction**(const char *faction) noexcept

Set the faction of the temporary record stored on the server for the currently specified record type.

Parameters

faction – The faction of the record.

Returns

void

static void SetRecordScale (double scale) noexcept

Set the scale of the temporary record stored on the server for the currently specified record type.

Parameters

scale – The scale of the record.

Returns

void

static void **SetRecordBloodType**(int bloodType) noexcept

Set the blood type of the temporary record stored on the server for the currently specified record type.

Parameters

bloodType – The blood type of the record.

Returns

void

$static\ void\ \textbf{SetRecordVampireState} (bool\ vampireState)\ no except$

Set the vampire state of the temporary record stored on the server for the currently specified record type.

Parameters

vampireState – The vampire state of the record.

Returns

static void **SetRecordLevel** (int level) noexcept

Set the level of the temporary record stored on the server for the currently specified record type.

Parameters

level – The level of the record.

Returns

void

static void **SetRecordMagicka**(int magicka) noexcept

Set the magicka of the temporary record stored on the server for the currently specified record type.

Parameters

magicka - The magicka of the record.

Returns

void

static void SetRecordFatigue(int fatigue) noexcept

Set the fatigue of the temporary record stored on the server for the currently specified record type.

Parameters

fatigue – The fatigue of the record.

Returns

void

static void SetRecordSoulValue(int soulValue) noexcept

Set the soul value of the temporary record stored on the server for the currently specified record type.

Parameters

soulValue – The soul value of the record.

Returns

void

static void **SetRecordAIFight** (int aiFight) noexcept

Set the AI fight value of the temporary record stored on the server for the currently specified record type.

Parameters

aiFight – The AI fight value of the record.

Returns

void

static void **SetRecordAIFlee**(int aiFlee) noexcept

Set the AI flee value of the temporary record stored on the server for the currently specified record type.

Parameters

aiFlee – The AI flee value of the record.

Returns

void

static void SetRecordAIAlarm(int aiAlarm) noexcept

Set the AI alarm value of the temporary record stored on the server for the currently specified record type.

Parameters

aiAlarm - The AI alarm value of the record.

Returns

static void **SetRecordAIServices** (int aiServices) noexcept

Set the AI services value of the temporary record stored on the server for the currently specified record type.

Parameters

aiServices – The AI services value of the record.

Returns

void

static void **SetRecordSound**(const char *sound) noexcept

Set the sound of the temporary record stored on the server for the currently specified record type.

Parameters

sound – The sound of the record.

Returns

void

static void SetRecordVolume (double volume) noexcept

Set the volume of the temporary record stored on the server for the currently specified record type.

Parameters

volume – The volume of the record.

Returns

void

$static\ void\ \textbf{SetRecordMinRange} (double\ minRange)\ no except$

Set the minimum range of the temporary record stored on the server for the currently specified record type.

Parameters

minRange – The minimum range of the record.

Returns

void

static void SetRecordMaxRange (double maxRange) noexcept

Set the maximum range of the temporary record stored on the server for the currently specified record type.

Parameters

maxRange – The maximum range of the record.

Returns

void

static void SetRecordOpenSound(const char *sound) noexcept

Set the opening sound of the temporary record stored on the server for the currently specified record type.

Parameters

sound – The opening sound of the record.

Returns

void

static void SetRecordCloseSound(const char *sound) noexcept

Set the closing sound of the temporary record stored on the server for the currently specified record type.

Parameters

sound – The closing sound of the record.

void

static void **SetRecordScriptText**(const char *scriptText) noexcept

Set the script text of the temporary record stored on the server for the currently specified record type.

Parameters

scriptText – The script text of the record.

Returns

void

static void **SetRecordIntegerVariable**(int intVar) noexcept

Set the integer variable of the temporary record stored on the server for the currently specified record type.

Parameters

intVar – The integer variable of the record.

Returns

void

static void **SetRecordFloatVariable**(double floatVar) noexcept

Set the float variable of the temporary record stored on the server for the currently specified record type.

Parameters

floatVar – The float variable of the record.

Returns

void

static void **SetRecordStringVariable**(const char *stringVar) noexcept

Set the string variable of the temporary record stored on the server for the currently specified record type.

Parameters

stringVar – The string variable of the record.

Returns

void

static void SetRecordIdByIndex (unsigned int index, const char *id) noexcept

Set the id of the record at a certain index in the records stored on the server.

When resending a received RecordsDynamic packet, this allows you to set the server-generated id of a record without having to clear and recreate the packet.

Parameters

- **index** The index of the record.
- id The id of the record.

Returns

void

static void **SetRecordEnchantmentIdByIndex**(unsigned int index, const char *enchantmentId) noexcept

Set the enchantment id of the record at a certain index in the records stored on the server.

When resending a received RecordsDynamic packet, this allows you to set the server-generated enchantment id of a record without having to clear and recreate the packet.

Parameters

• index – The index of the record.

• **enchantmentId** – The enchantment id of the record.

Returns

void

static void SetRecordEffectId(unsigned int effectId) noexcept

Set the ID of the temporary effect stored on the server.

Parameters

effectId – The ID of the effect.

Returns

void

static void **SetRecordEffectAttribute**(int attributeId) noexcept

Set the ID of the attribute modified by the temporary effect stored on the server.

Parameters

attributeId – The ID of the attribute.

Returns

void

static void SetRecordEffectSkill(int skillId) noexcept

Set the ID of the skill modified by the temporary effect stored on the server.

Parameters

skillId – The ID of the skill.

Returns

void

static void SetRecordEffectRangeType (unsigned int rangeType) noexcept

Set the range type of the temporary effect stored on the server (0 for self, 1 for touch, 2 for target).

Parameters

rangeType – The range type of the effect.

Returns

void

static void **SetRecordEffectArea**(int area) noexcept

Set the area of the temporary effect stored on the server.

Parameters

area – The area of the effect.

Returns

void

static void **SetRecordEffectDuration**(int duration) noexcept

Set the duration of the temporary effect stored on the server.

Parameters

duration – The duration of the effect.

Returns

void

static void **SetRecordEffectMagnitudeMax**(int magnitudeMax) noexcept

Set the maximum magnitude of the temporary effect stored on the server.

Parameters

magnitudeMax – The maximum magnitude of the effect.

Returns

void

static void **SetRecordEffectMagnitudeMin**(int magnitudeMin) noexcept

Set the minimum magnitude of the temporary effect stored on the server.

Parameters

magnitudeMin – The minimum magnitude of the effect.

Returns

void

static void **SetRecordBodyPartType**(unsigned int partType) noexcept

Set the body part type of the temporary body part stored on the server (which then needs to be added to ARMOR or CLOTHING records) or set the body part type of the current record if it's a BODYPART.

Parameters

partType – The type of the body part.

Returns

void

static void SetRecordBodyPartIdForMale(const char *partId) noexcept

Set the id of the male version of the temporary body part stored on the server.

Parameters

partId – The id of the body part.

Returns

void

$static\ void\ \textbf{SetRecordBodyPartIdForFemale} (const\ char\ *partId)\ no except$

Set the id of the female version of the temporary body part stored on the server.

Parameters

partId – The id of the body part.

Returns

void

$static\ void\ \textbf{SetRecordInventoryItemId} (const\ char\ *itemId)\ no except$

Set the id of the of the temporary inventory item stored on the server.

Parameters

itemId – The id of the inventory item.

Returns

void

$static\ void\ \textbf{SetRecordInventoryItemCount} (unsigned\ int\ count)\ no except$

Set the count of the of the temporary inventory item stored on the server.

Parameters

count – The count of the inventory item.

Returns

static void **AddRecord()** noexcept

Add a copy of the server's temporary record of the current specified type to the stored records.

In the process, the server's temporary record will automatically be cleared so a new one can be set up.

Returns

void

static void AddRecordEffect() noexcept

Add a copy of the server's temporary effect to the temporary record of the current specified type.

In the process, the server's temporary effect will automatically be cleared so a new one can be set up.

Returns

void

static void AddRecordBodyPart() noexcept

Add a copy of the server's temporary body part to the temporary record of the current specified type.

In the process, the server's temporary body part will automatically be cleared so a new one can be set up.

Returns

void

static void AddRecordInventoryItem() noexcept

Add a copy of the server's temporary inventory item to the temporary record of the current specified type.

In the process, the server's temporary inventory item will automatically be cleared so a new one can be set up.

Note: Any items added this way will be ignored if the record already has a valid inventoryBaseId.

Returns

void

static void **SendRecordDynamic**(unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a RecordDynamic packet with the current specified record type.

Parameters

- **pid** The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

1.16 Server functions

class ServerFunctions

Public Static Functions

static void LogMessage (unsigned short level, const char *message) noexcept

Write a log message with its own timestamp.

It will have "[Script]:" prepended to it so as to mark it as a script-generated log message.

Parameters

- **level** The logging level used (0 for LOG_VERBOSE, 1 for LOG_INFO, 2 for LOG_WARN, 3 for LOG_ERROR, 4 for LOG_FATAL).
- message The message logged.

Returns

void

static void LogAppend (unsigned short level, const char *message) noexcept

Write a log message without its own timestamp.

It will have "[Script]:" prepended to it so as to mark it as a script-generated log message.

Parameters

- **level** The logging level used (0 for LOG_VERBOSE, 1 for LOG_INFO, 2 for LOG_WARN, 3 for LOG_ERROR, 4 for LOG_FATAL).
- message The message logged.

Returns

void

static void StopServer (int code) noexcept

Shut down the server.

Parameters

code – The shutdown code.

Returns

void

static void Kick(unsigned short pid) noexcept

Kick a certain player from the server.

Parameters

```
pid – The player ID.
```

Returns

void

static void **BanAddress**(const char *ipAddress) noexcept

Ban a certain IP address from the server.

Parameters

ipAddress – The IP address.

1.16. Server functions 93

void

static void **UnbanAddress** (const char *ipAddress) noexcept

Unban a certain IP address from the server.

Parameters

ipAddress – The IP address.

Returns

void

static bool DoesFilePathExist(const char *filePath) noexcept

Check whether a certain file path exists.

This will be a case sensitive check on case sensitive filesystems.

Whenever you want to enforce case insensitivity, use GetCaseInsensitiveFilename() instead.

Returns

Whether the file exists or not.

static const char *GetCaseInsensitiveFilename(const char *folderPath, const char *filename) noexcept

Get the first filename in a folder that has a case insensitive match with the filename argument.

This is used to retain case insensitivity when opening data files on Linux.

Returns

The filename that matches.

static const char *GetDataPath() noexcept

Get the path of the server's data folder.

Returns

The data path.

static unsigned int GetMillisecondsSinceServerStart() noexcept

Get the milliseconds elapsed since the server was started.

Returns

The time since the server's startup in milliseconds.

static const char *GetOperatingSystemType() noexcept

Get the type of the operating system used by the server.

Note: Currently, the type can be "Windows", "Linux", "OS X" or "Unknown OS".

Returns

The type of the operating system.

static const char *GetArchitectureType() noexcept

Get the architecture type used by the server.

Note: Currently, the type can be "64-bit", "32-bit", "ARMv#" or "Unknown architecture".

Returns

The architecture type.

 $static\ const\ char\ *\textbf{GetServerVersion()}\ no except$

Get the TES3MP version of the server.

Returns

The server version.

static const char *GetProtocolVersion() noexcept

Get the protocol version of the server.

Returns

The protocol version.

static int **GetAvgPing** (unsigned short pid) noexcept

Get the average ping of a certain player.

Parameters

pid – The player ID.

Returns

The average ping.

static const char ***GetIP**(unsigned short pid) noexcept

Get the IP address of a certain player.

Parameters

pid – The player ID.

Returns

The IP address.

static unsigned short **GetPort()** noexcept

Get the port used by the server.

Returns

The port.

static unsigned int **GetMaxPlayers**() noexcept

Get the maximum number of players.

Returns

Max players

static bool HasPassword() noexcept

Checking if the server requires a password to connect.

Returns

Whether the server requires a password

$static\ bool\ \textbf{GetDataFileEnforcementState}()\ no except$

Get the data file enforcement state of the server.

If true, clients are required to use the same data files as set for the server.

Returns

The enforcement state.

$static\ bool\ \textbf{GetScriptErrorIgnoringState()}\ no except$

Get the script error ignoring state of the server.

If true, script errors will not crash the server.

Returns

The script error ignoring state.

static void **SetGameMode**(const char *gameMode) noexcept

Set the game mode of the server, as displayed in the server browser.

Parameters

gameMode – The new game mode.

1.16. Server functions 95

void

static void **SetHostname**(const char *name) noexcept

Set the name of the server, as displayed in the server browser.

Parameters

name – The new name.

Returns

void

static void **SetServerPassword**(const char *password) noexcept

Set the password required to join the server.

Parameters

password – The password.

Returns

void

static void **SetDataFileEnforcementState**(bool state) noexcept

Set the data file enforcement state of the server.

If true, clients are required to use the same data files as set for the server.

Parameters

state – The new enforcement state.

Returns

void

static void **SetScriptErrorIgnoringState**(bool state) noexcept

Set whether script errors should be ignored or not.

If true, script errors will not crash the server, but could have any number of unforeseen consequences, which is why this is a highly experimental setting.

Parameters

state – The new script error ignoring state.

Returns

void

 $static\ void\ \textbf{SetRuleString} (const\ char\ *key,\ const\ char\ *value)\ noexcept$

Set a rule string for the server details displayed in the server browser.

Parameters

- **key** The name of the rule.
- **value** The string value of the rule.

Returns

void

static void **SetRuleValue** (const char *key, double value) noexcept

Set a rule value for the server details displayed in the server browser.

Parameters

- **key** The name of the rule.
- **value** The numerical value of the rule.

void

static void **AddDataFileRequirement**(const char *dataFilename, const char *checksumString) noexcept

Add a data file and a corresponding CRC32 checksum to the data file loadout that connecting clients need to match.

It can be used multiple times to set multiple checksums for the same data file.

Note: If an empty string is provided for the checksum, a checksum will not be required for that data file.

Parameters

- dataFilename The filename of the data file.
- **checksumString** A string with the CRC32 checksum required.

1.17 Setting functions

class SettingFunctions

Public Static Functions

static void **SetDifficulty**(unsigned short pid, int difficulty)

Set the difficulty for a player.

This changes the difficulty for that player in the server memory, but does not by itself send a packet.

Parameters

- pid The player ID.
- **difficulty** The difficulty.

Returns

void

static void **SetEnforcedLogLevel** (unsigned short pid, int enforcedLogLevel)

Set the client log level enforced for a player.

This changes the enforced log level for that player in the server memory, but does not by itself send a packet.

Enforcing a certain log level is necessary to prevent players from learning information from their console window that they are otherwise unable to obtain, such as the locations of other players.

If you do not wish to enforce a log level, simply set enforcedLogLevel to -1

Parameters

- **pid** The player ID.
- **enforcedLogLevel** The enforced log level.

Returns

void

static void **SetPhysicsFramerate**(unsigned short pid, double physicsFramerate)

Set the physics framerate for a player.

This changes the physics framerate for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- physicsFramerate The physics framerate.

Returns

void

static void **SetConsoleAllowed**(unsigned short pid, bool state)

Set whether the console is allowed for a player.

This changes the console permission for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **state** The console permission state.

Returns

void

static void **SetBedRestAllowed** (unsigned short pid, bool state)

Set whether resting in beds is allowed for a player.

This changes the resting permission for that player in the server memory, but does not by itself send a packet.

Parameters

- pid The player ID.
- **state** The resting permission state.

Returns

void

static void SetWildernessRestAllowed(unsigned short pid, bool state)

Set whether resting in the wilderness is allowed for a player.

This changes the resting permission for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **state** The resting permission state.

Returns

void

static void **SetWaitAllowed** (unsigned short pid, bool state)

Set whether waiting is allowed for a player.

This changes the waiting permission for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **state** The waiting permission state.

void

static void **SetGameSettingValue**(unsigned short pid, const char *setting, const char *value)

Set value for a game setting.

This overrides the setting value set in OpenMW Launcher. Only applies to the Game category.

Parameters

- **pid** The player ID.
- **setting** Name of a setting in the Game category
- **value** Value of the setting (as a string)

Returns

void

static void ClearGameSettingValues (unsigned short pid)

Clear the Game setting values stored for a player.

Clear any changes done by SetGameSettingValue()

Parameters

```
pid – The player ID.
```

Returns

voic

static void SetVRSettingValue(unsigned short pid, const char *setting, const char *value)

Set value for a VR setting.

This overrides the setting value set in OpenMW Launcher. Only applies to the VR category.

Parameters

- pid The player ID.
- **setting** Name of a setting in the VR category
- value Value of the setting (as a string)

Returns

void

static void ClearVRSettingValues (unsigned short pid)

Clear the VR setting values stored for a player.

Clear any changes done by SetVRSettingValue()

Parameters

```
pid – The player ID.
```

Returns

void

static void **SendSettings** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a PlayerSettings packet to the player affected by it.

Parameters

```
pid – The player ID to send it to.
```

Returns

1.18 Shapeshift functions

class ShapeshiftFunctions

Public Static Functions

static double GetScale (unsigned short pid) noexcept

Get the scale of a player.

Parameters

pid – The player ID.

Returns

The scale.

static bool IsWerewolf (unsigned short pid) noexcept

Check whether a player is a werewolf.

This is based on the last PlayerShapeshift packet received or sent for that player.

Parameters

pid – The player ID.

Returns

The werewolf state.

static const char *GetCreatureRefId(unsigned short pid) noexcept

Get the refId of the creature the player is disguised as.

Parameters

pid – The player ID.

Returns

The creature refId.

static bool GetCreatureNameDisplayState(unsigned short pid) noexcept

Check whether a player's name is replaced by that of the creature they are disguised as when other players hover over them.

This is based on the last PlayerShapeshift packet received or sent for that player.

Parameters

pid – The player ID.

Returns

The creature name display state.

static void SetScale (unsigned short pid, double scale) noexcept

Set the scale of a player.

This changes the scale recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- **scale** The new scale.

Returns

static void **SetWerewolfState**(unsigned short pid, bool isWerewolf) noexcept

Set the werewolf state of a player.

This changes the werewolf state recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- pid The player ID.
- **isWerewolf** The new werewolf state.

Returns

void

static void **SetCreatureRefId**(unsigned short pid, const char *refId) noexcept

Set the refId of the creature a player is disguised as.

This changes the creature refId recorded for that player in the server memory, but does not by itself send a packet.

Parameters

- **pid** The player ID.
- refId The creature refId.

Returns

void

static void SetCreatureNameDisplayState(unsigned short pid, bool displayState) noexcept

Set whether a player's name is replaced by that of the creature they are disguised as when other players hover over them.

Parameters

- **pid** The player ID.
- **displayState** The creature name display state.

Returns

void

static void SendShapeshift (unsigned short pid)

Send a PlayerShapeshift packet about a player.

This sends the packet to all players connected to the server. It is currently used only to communicate werewolf states.

Parameters

```
pid – The player ID.
```

Returns

1.19 Spell functions

class SpellFunctions

Public Static Functions

static void ClearSpellbookChanges (unsigned short pid) noexcept

Clear the last recorded spellbook changes for a player.

This is used to initialize the sending of new PlayerSpellbook packets.

Parameters

pid – The player ID whose spellbook changes should be used.

Returns

void

static void ClearSpellsActiveChanges (unsigned short pid) noexcept

Clear the last recorded spells active changes for a player.

This is used to initialize the sending of new PlayerSpellsActive packets.

Parameters

pid – The player ID whose spells active changes should be used.

Returns

void

static void ClearCooldownChanges (unsigned short pid) noexcept

Clear the last recorded cooldown changes for a player.

This is used to initialize the sending of new PlayerCooldown packets.

Parameters

pid – The player ID whose cooldown changes should be used.

Returns

void

static unsigned int **GetSpellbookChangesSize**(unsigned short pid) noexcept

Get the number of indexes in a player's latest spellbook changes.

Parameters

pid – The player ID whose spellbook changes should be used.

Returns

The number of indexes.

static unsigned int **GetSpellbookChangesAction**(unsigned short pid) noexcept

Get the action type used in a player's latest spellbook changes.

Parameters

pid – The player ID whose spellbook changes should be used.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE).

static unsigned int GetSpellsActiveChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest spells active changes.

Parameters

pid – The player ID whose spells active changes should be used.

Returns

The number of indexes for spells active changes.

static unsigned int GetSpellsActiveChangesAction (unsigned short pid) noexcept

Get the action type used in a player's latest spells active changes.

Parameters

pid – The player ID whose spells active changes should be used.

Returns

The action type (0 for SET, 1 for ADD, 2 for REMOVE).

static unsigned int GetCooldownChangesSize(unsigned short pid) noexcept

Get the number of indexes in a player's latest cooldown changes.

Parameters

pid – The player ID whose cooldown changes should be used.

Returns

The number of indexes.

static void **SetSpellbookChangesAction**(unsigned short pid, unsigned char action) noexcept Set the action type in a player's spellbook changes.

Parameters

- pid The player ID whose spellbook changes should be used.
- **action** The action (0 for SET, 1 for ADD, 2 for REMOVE).

Returns

void

static void **SetSpellsActiveChangesAction**(unsigned short pid, unsigned char action) noexcept Set the action type in a player's spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- action The action (0 for SET, 1 for ADD, 2 for REMOVE).

Returns

void

static void AddSpell (unsigned short pid, const char *spellId) noexcept

Add a new spell to the spellbook changes for a player.

Parameters

- **pid** The player ID whose spellbook changes should be used.
- **spellId** The spellId of the spell.

Returns

static void **AddSpellActive** (unsigned short pid, const char *spellId, const char *displayName, bool stackingState) noexcept

Add a new active spell to the spells active changes for a player, using the temporary effect values stored so far.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **spellId** The spellId of the spell.
- **displayName** The displayName of the spell.
- **stackingState** Whether the spell should stack with other instances of itself.

Returns

void

static void **AddSpellActiveEffect** (unsigned short pid, int effectId, double magnitude, double duration, double timeLeft, int arg) noexcept

Add a new effect to the next active spell that will be added to a player.

Parameters

- pid The player ID whose spells active changes should be used.
- **effectId** The id of the effect.
- **magnitude** The magnitude of the effect.
- duration The duration of the effect.
- timeLeft The timeLeft for the effect.
- **arg** The arg of the effect when applicable, e.g. the skill used for Fortify Skill or the attribute used for Fortify Attribute.

Returns

void

static void **AddCooldownSpell** (unsigned short pid, const char *spellId, unsigned int startDay, double startHour) noexcept

Add a new cooldown spell to the cooldown changes for a player.

Parameters

- **pid** The player ID whose cooldown changes should be used.
- **spellId** The spellId of the spell.
- **startDay** The day on which the cooldown starts.
- **startHour** The hour at which the cooldown starts.

Returns

void

static const char ***GetSpellId**(unsigned short pid, unsigned int index) noexcept Get the spell id at a certain index in a player's latest spellbook changes.

Parameters

- pid The player ID whose spellbook changes should be used.
- **index** The index of the spell.

The spell id.

static const char ***GetSpellsActiveId**(unsigned short pid, unsigned int index) noexcept

Get the spell id at a certain index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The spell id.

static const char *GetSpellsActiveDisplayName(unsigned short pid, unsigned int index) noexcept

Get the spell display name at a certain index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The spell display name.

static bool **GetSpellsActiveStackingState** (unsigned short pid, unsigned int index) noexcept Get the spell stacking state at a certain index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The spell stacking state.

static unsigned int **GetSpellsActiveEffectCount** (unsigned short pid, unsigned int index) noexcept Get the number of effects at an index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The number of effects.

static unsigned int **GetSpellsActiveEffectId**(unsigned short pid, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the id for an effect index at a spell index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The id of the effect.

static int **GetSpellsActiveEffectArg**(unsigned short pid, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the arg for an effect index at a spell index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The arg of the effect.

static double **GetSpellsActiveEffectMagnitude**(unsigned short pid, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the magnitude for an effect index at a spell index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The magnitude of the effect.

static double **GetSpellsActiveEffectDuration**(unsigned short pid, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the duration for an effect index at a spell index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The duration of the effect.

static double **GetSpellsActiveEffectTimeLeft**(unsigned short pid, unsigned int spellIndex, unsigned int effectIndex) noexcept

Get the time left for an effect index at a spell index in a player's latest spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **spellIndex** The index of the spell.
- **effectIndex** The index of the effect.

Returns

The time left for the effect.

static bool **DoesSpellsActiveHavePlayerCaster** (unsigned short pid, unsigned int index) noexcept

Check whether the spell at a certain index in a player's latest spells active changes has a player as its caster.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Whether a player is the caster of the spell.

 $static \ int \ \textbf{GetSpellsActiveCasterPid} (unsigned \ short \ pid, \ unsigned \ int \ index) \ no except$

Get the player ID of the caster of the spell at a certain index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The player ID of the caster.

static const char *GetSpellsActiveCasterRefId(unsigned short pid, unsigned int index) noexcept

Get the refId of the actor caster of the spell at a certain index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The refId of the caster.

static unsigned int **GetSpellsActiveCasterRefNum**(unsigned short pid, unsigned int index) noexcept

Get the refNum of the actor caster of the spell at a certain index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The refNum of the caster.

static unsigned int **GetSpellsActiveCasterMpNum**(unsigned short pid, unsigned int index) noexcept

Get the mpNum of the actor caster of the spell at a certain index in a player's latest spells active changes.

Parameters

- pid The player ID whose spells active changes should be used.
- **index** The index of the spell.

Returns

The mpNum of the caster.

static const char *GetCooldownSpellId(unsigned short pid, unsigned int index) noexcept Get the spell id at a certain index in a player's latest cooldown changes.

Parameters

- pid The player ID whose cooldown changes should be used.
- index The index of the cooldown spell.

Returns

The spell id.

static unsigned int GetCooldownStartDay(unsigned short pid, unsigned int index) noexcept

Get the starting day of the cooldown at a certain index in a player's latest cooldown changes.

Parameters

- pid The player ID whose cooldown changes should be used.
- **index** The index of the cooldown spell.

Returns

The starting day of the cooldown.

static double GetCooldownStartHour (unsigned short pid, unsigned int index) noexcept

Get the starting hour of the cooldown at a certain index in a player's latest cooldown changes.

Parameters

- **pid** The player ID whose cooldown changes should be used.
- index The index of the cooldown spell.

Returns

The starting hour of the cooldown.

static void **SendSpellbookChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerSpellbook packet with a player's recorded spellbook changes.

Parameters

- **pid** The player ID whose spellbook changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendSpellsActiveChanges** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a PlayerSpellsActive packet with a player's recorded spells active changes.

Parameters

- **pid** The player ID whose spells active changes should be used.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendCooldownChanges** (unsigned short pid) noexcept

Send a PlayerCooldowns packet with a player's recorded cooldown changes.

Parameters

pid – The player ID whose cooldown changes should be used.

void

1.20 Stats functions

class StatsFunctions

Public Static Functions

static int GetAttributeCount() noexcept

Get the number of attributes.

The number is 8 before any dehardcoding is done in OpenMW.

Returns

The number of attributes.

static int GetSkillCount() noexcept

Get the number of skills.

The number is 27 before any dehardcoding is done in OpenMW.

Returns

The number of skills.

static int GetAttributeId(const char *name) noexcept

Get the numerical ID of an attribute with a certain name.

If an invalid name is used, the ID returned is -1

Parameters

name – The name of the attribute.

Returns

The ID of the attribute.

static int GetSkillId(const char *name) noexcept

Get the numerical ID of a skill with a certain name.

If an invalid name is used, the ID returned is -1

Parameters

name – The name of the skill.

Returns

The ID of the skill.

static const char *GetAttributeName(unsigned short attributeId) noexcept

Get the name of the attribute with a certain numerical ID.

If an invalid ID is used, "invalid" is returned.

Parameters

attributeId – The ID of the attribute.

Returns

The name of the attribute.

1.20. Stats functions 109

static const char *GetSkillName(unsigned short skillId) noexcept

Get the name of the skill with a certain numerical ID.

If an invalid ID is used, "invalid" is returned.

Parameters

skillId – The ID of the skill.

Returns

The name of the skill.

static const char *GetName(unsigned short pid) noexcept

Get the name of a player.

Parameters

pid – The player ID.

Returns

The name of the player.

static const char *GetRace(unsigned short pid) noexcept

Get the race of a player.

Parameters

pid – The player ID.

Returns

The race of the player.

static const char *GetHead(unsigned short pid) noexcept

Get the head mesh used by a player.

Parameters

pid – The player ID.

Returns

The head mesh of the player.

static const char *GetHairstyle(unsigned short pid) noexcept

Get the hairstyle mesh used by a player.

Parameters

pid – The player ID.

Returns

The hairstyle mesh of the player.

static int GetIsMale(unsigned short pid) noexcept

Check whether a player is male or not.

Parameters

pid – The player ID.

Returns

Whether the player is male.

static const char *GetModel (unsigned short pid) noexcept

Get the model of a player.

Parameters

pid – The player ID.

The model of the player.

static const char *GetBirthsign(unsigned short pid) noexcept

Get the birthsign of a player.

Parameters

pid – The player ID.

Returns

The birthsign of the player.

static int GetLevel (unsigned short pid) noexcept

Get the character level of a player.

Parameters

pid – The player ID.

Returns

The level of the player.

static int GetLevelProgress (unsigned short pid) noexcept

Get the player's progress to their next character level.

Parameters

pid – The player ID.

Returns

The level progress.

static double GetHealthBase (unsigned short pid) noexcept

Get the base health of the player.

Parameters

pid – The player ID.

Returns

The base health.

static double GetHealthCurrent (unsigned short pid) noexcept

Get the current health of the player.

Parameters

pid – The player ID.

Returns

The current health.

static double GetMagickaBase(unsigned short pid) noexcept

Get the base magicka of the player.

Parameters

pid – The player ID.

Returns

The base magicka.

static double GetMagickaCurrent (unsigned short pid) noexcept

Get the current magicka of the player.

Parameters

pid - The player ID.

1.20. Stats functions 111

The current magicka.

static double GetFatigueBase (unsigned short pid) noexcept

Get the base fatigue of the player.

Parameters

pid – The player ID.

Returns

The base fatigue.

static double GetFatigueCurrent (unsigned short pid) noexcept

Get the current fatigue of the player.

Parameters

pid – The player ID.

Returns

The current fatigue.

static int **GetAttributeBase** (unsigned short pid, unsigned short attributeId) noexcept Get the base value of a player's attribute.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.

Returns

The base value of the attribute.

static int **GetAttributeModifier**(unsigned short pid, unsigned short attributeId) noexcept Get the modifier value of a player's attribute.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.

Returns

The modifier value of the attribute.

static double GetAttributeDamage (unsigned short pid, unsigned short attributeId) noexcept

Get the amount of damage (as caused through the Damage Attribute effect) to a player's attribute.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.

Returns

The amount of damage to the attribute.

static int GetSkillBase (unsigned short pid, unsigned short skillId) noexcept

Get the base value of a player's skill.

Parameters

- **pid** The player ID.
- **skillId** The skill ID.

The base value of the skill.

static int GetSkillModifier (unsigned short pid, unsigned short skillId) noexcept

Get the modifier value of a player's skill.

Parameters

- pid The player ID.
- **skillId** The skill ID.

Returns

The modifier value of the skill.

static double GetSkillDamage (unsigned short pid, unsigned short skillId) noexcept

Get the amount of damage (as caused through the Damage Skill effect) to a player's skill.

Parameters

- pid The player ID.
- **skillId** The skill ID.

Returns

The amount of damage to the skill.

static double GetSkillProgress (unsigned short pid, unsigned short skillId) noexcept

Get the progress the player has made towards increasing a certain skill by 1.

Parameters

- pid The player ID.
- **skillId** The skill ID.

Returns

The skill progress.

static int GetSkillIncrease(unsigned short pid, unsigned int attributeId) noexcept

Get the bonus applied to a certain attribute at the next level up as a result of associated skill increases.

Although confusing, the term "skill increase" for this is taken from OpenMW itself.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.

Returns

The increase in the attribute caused by skills.

static int GetBounty (unsigned short pid) noexcept

Get the bounty of the player.

Parameters

pid – The player ID.

Returns

The bounty.

1.20. Stats functions 113

static void **SetName** (unsigned short pid, const char *name) noexcept Set the name of a player.

Parameters

- **pid** The player ID.
- name The new name of the player.

Returns

void

static void **SetRace**(unsigned short pid, const char *race) noexcept Set the race of a player.

Parameters

- pid The player ID.
- race The new race of the player.

Returns

void

static void **SetHead**(unsigned short pid, const char *head) noexcept Set the head mesh used by a player.

Parameters

- **pid** The player ID.
- **head** The new head mesh of the player.

Returns

void

static void **SetHairstyle**(unsigned short pid, const char *hairstyle) noexcept Set the hairstyle mesh used by a player.

Parameters

- **pid** The player ID.
- hairstyle The new hairstyle mesh of the player.

Returns

void

static void ${\tt SetIsMale} (unsigned short pid, int state)$ no except

Set whether a player is male or not.

Parameters

- pid The player ID.
- $\bullet \ \ \textbf{state} Whether \ the \ player \ is \ male.$

Returns

void

static void **SetModel** (unsigned short pid, const char *model) noexcept Set the model of a player.

Parameters

• **pid** – The player ID.

• model – The new model of the player.

Returns

void

static void SetBirthsign (unsigned short pid, const char *name) noexcept

Set the birthsign of a player.

Parameters

- **pid** The player ID.
- name The new birthsign of the player.

Returns

void

static void SetResetStats (unsigned short pid, bool resetStats) noexcept

Set whether the player's stats should be reset based on their current race as the result of a PlayerBaseInfo packet.

This changes the resetState for that player in the server memory, but does not by itself send a packet.

Parameters

- pid The player ID.
- resetStats The stat reset state.

Returns

void

static void SetLevel (unsigned short pid, int value) noexcept

Set the character level of a player.

Parameters

- **pid** The player ID.
- **value** The new level of the player.

Returns

void

static void SetLevelProgress (unsigned short pid, int value) noexcept

Set the player's progress to their next character level.

Parameters

- **pid** The player ID.
- **value** The new level progress of the player.

Returns

void

 $static\ void\ \textbf{SetHealthBase} (unsigned\ short\ pid,\ double\ value)\ no except$

Set the base health of a player.

Parameters

- pid The player ID.
- value The new base health of the player.

1.20. Stats functions 115

void

static void **SetHealthCurrent** (unsigned short pid, double value) noexcept Set the current health of a player.

Parameters

- **pid** The player ID.
- value The new current health of the player.

Returns

void

static void **SetMagickaBase**(unsigned short pid, double value) noexcept Set the base magicka of a player.

Parameters

- **pid** The player ID.
- value The new base magicka of the player.

Returns

void

static void **SetMagickaCurrent** (unsigned short pid, double value) noexcept Set the current magicka of a player.

Parameters

- **pid** The player ID.
- value The new current magicka of the player.

Returns

void

static void **SetFatigueBase** (unsigned short pid, double value) noexcept Set the base fatigue of a player.

Parameters

- **pid** The player ID.
- value The new base fatigue of the player.

Returns

void

static void **SetFatigueCurrent** (unsigned short pid, double value) noexcept Set the current fatigue of a player.

Parameters

- **pid** The player ID.
- **value** The new current fatigue of the player.

Returns

static void **SetAttributeBase**(unsigned short pid, unsigned short attributeId, int value) noexcept Set the base value of a player's attribute.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.
- value The new base value of the player's attribute.

Returns

void

static void ClearAttributeModifier (unsigned short pid, unsigned short attributeId) noexcept

Clear the modifier value of a player's attribute.

There's no way to set a modifier to a specific value because it can come from multiple different sources, but clearing it is a straightforward process that dispels associated effects on a client and, if necessary, unequips associated items.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.

Returns

void

static void **SetAttributeDamage** (unsigned short pid, unsigned short attributeId, double value) noexcept Set the amount of damage (as caused through the Damage Attribute effect) to a player's attribute.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.
- value The amount of damage to the player's attribute.

Returns

void

static void **SetSkillBase**(unsigned short pid, unsigned short skillId, int value) noexcept Set the base value of a player's skill.

Parameters

- **pid** The player ID.
- **skillId** The skill ID.
- value The new base value of the player's skill.

Returns

void

static void ClearSkillModifier (unsigned short pid, unsigned short skillId) noexcept

Clear the modifier value of a player's skill.

There's no way to set a modifier to a specific value because it can come from multiple different sources, but clearing it is a straightforward process that dispels associated effects on a client and, if necessary, unequips associated items.

Parameters

1.20. Stats functions 117

- **pid** The player ID.
- skillId The skill ID.

void

static void **SetSkillDamage** (unsigned short pid, unsigned short skillId, double value) noexcept Set the amount of damage (as caused through the Damage Skill effect) to a player's skill.

Parameters

- **pid** The player ID.
- **skillId** The skill ID.
- value The amount of damage to the player's skill.

Returns

void

static void **SetSkillProgress** (unsigned short pid, unsigned short skillId, double value) noexcept Set the progress the player has made towards increasing a certain skill by 1.

Parameters

- **pid** The player ID.
- **skillId** The skill ID.
- **value** The progress value.

Returns

void

static void **SetSkillIncrease**(unsigned short pid, unsigned int attributeId, int value) noexcept

Set the bonus applied to a certain attribute at the next level up as a result of associated skill increases.

Although confusing, the term "skill increase" for this is taken from OpenMW itself.

Parameters

- **pid** The player ID.
- attributeId The attribute ID.
- value The increase in the attribute caused by skills.

Returns

void

static void ${\tt SetBounty} (unsigned \ short \ pid, \ int \ value) \ no except$

Set the bounty of a player.

Parameters

- **pid** The player ID.
- **value** The new bounty.

Returns

void

static void **SetCharGenStage** (unsigned short pid, int currentStage, int endStage) noexcept Set the current and ending stages of character generation for a player.

This is used to repeat part of character generation or to only go through part of it.

Parameters

- pid The player ID.
- **currentStage** The new current stage.
- endStage The new ending stage.

Returns

void

static void SendBaseInfo(unsigned short pid) noexcept

Send a PlayerBaseInfo packet with a player's name, race, head mesh, hairstyle mesh, birthsign and stat reset state.

It is always sent to all players.

Parameters

pid – The player ID.

Returns

void

static void SendStatsDynamic(unsigned short pid) noexcept

Send a PlayerStatsDynamic packet with a player's dynamic stats (health, magicka and fatigue).

It is always sent to all players.

Parameters

pid – The player ID.

Returns

void

static void SendAttributes (unsigned short pid) noexcept

Send a PlayerAttribute packet with a player's attributes and bonuses to those attributes at the next level up (the latter being called "skill increases" as in OpenMW).

It is always sent to all players.

Parameters

pid – The player ID.

Returns

void

static void SendSkills (unsigned short pid) noexcept

Send a PlayerSkill packet with a player's skills.

It is always sent to all players.

Parameters

pid – The player ID.

Returns

void

static void **SendLevel** (unsigned short pid) noexcept

Send a PlayerLevel packet with a player's character level and progress towards the next level up.

It is always sent to all players.

Parameters

pid - The player ID.

1.20. Stats functions 119

void

static void **SendBounty** (unsigned short pid) noexcept

Send a PlayerBounty packet with a player's bounty.

It is always sent to all players.

Parameters

pid – The player ID.

Returns

void

1.21 Worldstate functions

class WorldstateFunctions

Public Static Functions

static void ReadReceivedWorldstate() noexcept

Use the last worldstate received by the server as the one being read.

Returns

void

static void CopyReceivedWorldstateToStore() noexcept

Take the contents of the read-only worldstate last received by the server from a player and move its contents to the stored worldstate that can be sent by the server.

Returns

void

$static\ void\ \textbf{ClearKillChanges}()\ noexcept$

Clear the kill count changes for the write-only worldstate.

This is used to initialize the sending of new WorldKillCount packets.

Returns

void

static void **ClearMapChanges()** noexcept

Clear the map changes for the write-only worldstate.

This is used to initialize the sending of new WorldMap packets.

Returns

void

static void ClearClientGlobals() noexcept

Clear the client globals for the write-only worldstate.

This is used to initialize the sending of new ClientScriptGlobal packets.

Returns

static unsigned int GetKillChangesSize() noexcept

Get the number of indexes in the read worldstate's kill changes.

Returns

The number of indexes.

static unsigned int **GetMapChangesSize()** noexcept

Get the number of indexes in the read worldstate's map changes.

Returns

The number of indexes.

static unsigned int **GetClientGlobalsSize()** noexcept

Get the number of indexes in the read worldstate's client globals.

Returns

The number of indexes.

static const char *GetKillRefId(unsigned int index) noexcept

Get the refId at a certain index in the read worldstate's kill count changes.

Parameters

index – The index of the kill count.

Returns

The refId.

static int **GetKillNumber**(unsigned int index) noexcept

Get the number of kills at a certain index in the read worldstate's kill count changes.

Parameters

index - The index of the kill count.

Returns

The number of kills.

static const char *GetWeatherRegion() noexcept

Get the weather region in the read worldstate.

Returns

The weather region.

static int GetWeatherCurrent() noexcept

Get the current weather in the read worldstate.

Returns

The current weather.

static int **GetWeatherNext()** noexcept

Get the next weather in the read worldstate.

Returns

The next weather.

static int GetWeatherQueued() noexcept

Get the queued weather in the read worldstate.

Returns

The queued weather.

static double **GetWeatherTransitionFactor()** noexcept

Get the transition factor of the weather in the read worldstate.

Returns

The transition factor of the weather.

static int GetMapTileCellX(unsigned int index) noexcept

Get the X coordinate of the cell corresponding to the map tile at a certain index in the read worldstate's map tiles.

Parameters

index – The index of the map tile.

Returns

The X coordinate of the cell.

static int GetMapTileCellY(unsigned int index) noexcept

Get the Y coordinate of the cell corresponding to the map tile at a certain index in the read worldstate's map tiles.

Parameters

index – The index of the map tile.

Returns

The Y coordinate of the cell.

static const char *GetClientGlobalId(unsigned int index) noexcept

Get the id of the global variable at a certain index in the read worldstate's client globals.

Parameters

index – The index of the client global.

Returns

The id.

static unsigned short **GetClientGlobalVariableType**(unsigned int index) noexcept

Get the type of the global variable at a certain index in the read worldstate's client globals.

Parameters

index – The index of the client global.

Returns

The variable type (0 for INTEGER, 1 for LONG, 2 for FLOAT).

static int GetClientGlobalIntValue(unsigned int index) noexcept

Get the integer value of the global variable at a certain index in the read worldstate's client globals.

Parameters

index – The index of the client global.

Returns

The integer value.

static double **GetClientGlobalFloatValue**(unsigned int index) noexcept

Get the float value of the global variable at a certain index in the read worldstate's client globals.

Parameters

index – The index of the client global.

Returns

The float value.

static void **SetAuthorityRegion**(const char *authorityRegion) noexcept

Set the region affected by the next WorldRegionAuthority packet sent.

Parameters

authorityRegion – The region.

Returns

void

static void **SetWeatherRegion**(const char *region) noexcept

Set the weather region in the write-only worldstate stored on the server.

Parameters

region – The region.

Returns

void

static void **SetWeatherForceState**(bool forceState) noexcept

Set the weather forcing state in the write-only worldstate stored on the server.

Players who receive a packet with forced weather will switch to that weather immediately.

Parameters

forceState – The weather forcing state.

Returns

void

static void SetWeatherCurrent(int currentWeather) noexcept

Set the current weather in the write-only worldstate stored on the server.

Parameters

currentWeather – The current weather.

Returns

void

static void **SetWeatherNext**(int nextWeather) noexcept

Set the next weather in the write-only worldstate stored on the server.

Parameters

nextWeather – The next weather.

Returns

void

static void **SetWeatherQueued**(int queuedWeather) noexcept

Set the queued weather in the write-only worldstate stored on the server.

Parameters

queuedWeather – The queued weather.

Returns

void

static void **SetWeatherTransitionFactor** (double transitionFactor) noexcept

Set the transition factor for the weather in the write-only worldstate stored on the server.

Parameters

transitionFactor – The transition factor.

void

static void SetHour (double hour) noexcept

Set the world's hour in the write-only worldstate stored on the server.

Parameters

hour - The hour.

Returns

void

static void SetDay(int day) noexcept

Set the world's day in the write-only worldstate stored on the server.

Parameters

day - The day.

Returns

void

static void **SetMonth**(int month) noexcept

Set the world's month in the write-only worldstate stored on the server.

Parameters

month – The month.

Returns

void

static void SetYear (int year) noexcept

Set the world's year in the write-only worldstate stored on the server.

Parameters

year – The year.

Returns

void

static void SetDaysPassed(int daysPassed) noexcept

Set the world's days passed in the write-only worldstate stored on the server.

Parameters

daysPassed – The days passed.

Returns

void

static void SetTimeScale (double timeScale) noexcept

Set the world's time scale in the write-only worldstate stored on the server.

Parameters

timeScale – The time scale.

Returns

void

static void **SetPlayerCollisionState**(bool state) noexcept

Set the collision state for other players in the write-only worldstate stored on the server.

Parameters

state – The collision state.

void

static void **SetActorCollisionState**(bool state) noexcept

Set the collision state for actors in the write-only worldstate stored on the server.

Parameters

state - The collision state.

Returns

void

static void SetPlacedObjectCollisionState(bool state) noexcept

Set the collision state for placed objects in the write-only worldstate stored on the server.

Parameters

state – The collision state.

Returns

void

static void UseActorCollisionForPlacedObjects(bool useActorCollision) noexcept

Whether placed objects with collision turned on should use actor collision, i.e. whether they should be slippery and prevent players from standing on them.

Parameters

useActorCollision – Whether to use actor collision.

Returns

void

static void AddKill (const char *refId, int number) noexcept

Add a new kill count to the kill count changes.

Parameters

- **refId** The refId of the kill count.
- number The number of kills in the kill count.

Returns

void

static void **AddClientGlobalInteger**(const char *id, int intValue, unsigned int variableType = 0) noexcept Add a new client global integer to the client globals.

Parameters

- **id** The id of the client global.
- **variableType** The variable type (0 for SHORT, 1 for LONG).
- intValue The integer value of the client global.

Returns

void

static void AddClientGlobalFloat(const char *id, double floatValue) noexcept

Add a new client global float to the client globals.

Parameters

- id The id of the client global.
- **floatValue** The float value of the client global.

void

static void AddSynchronizedClientScriptId(const char *scriptId) noexcept

Add an ID to the list of script IDs whose variable changes should be sent to the the server by clients.

Parameters

scriptId - The ID.

Returns

void

$static\ void\ \textbf{AddSynchronizedClientGlobalId} (const\ char\ *globalId)\ no except$

Add an ID to the list of global IDs whose value changes should be sent to the server by clients.

Parameters

globalId – The ID.

Returns

void

static void AddEnforcedCollisionRefId(const char *refId) noexcept

Add a refld to the list of reflds for which collision should be enforced irrespective of other settings.

Parameters

refId - The refId.

Returns

void

static void AddCellToReset(const char *cellDescription) noexcept

Add a cell with given cellDescription to the list of cells that should be reset on the client.

Returns

void

static void **AddDestinationOverride**(const char *oldCellDescription, const char *newCellDescription) noexcept

Add a destination override containing the cell description for the old cell and the new cell.

Parameters

- **oldCellDescription** The old cell description.
- **newCellDescription** The new cell description.

Returns

void

static void ClearSynchronizedClientScriptIds() noexcept

Clear the list of script IDs whose variable changes should be sent to the the server by clients.

Returns

void

static void ClearSynchronizedClientGlobalIds() noexcept

Clear the list of global IDs whose value changes should be sent to the the server by clients.

Returns

static void ClearEnforcedCollisionRefIds() noexcept

Clear the list of refIds for which collision should be enforced irrespective of other settings.

Returns

void

static void ClearCellsToReset() noexcept

Clear the list of cells which should be reset on the client.

Returns

void

static void ClearDestinationOverrides() noexcept

Clear the list of destination overrides.

Returns

void

static void SaveMapTileImageFile (unsigned int index, const char *filePath) noexcept

Save the .png image data of the map tile at a certain index in the read worldstate's map changes.

Parameters

- **index** The index of the map tile.
- **filePath** The file path of the resulting file.

Returns

void

static void LoadMapTileImageFile(int cellX, int cellY, const char *filePath) noexcept

Load a .png file as the image data for a map tile and add it to the write-only worldstate stored on the server.

Parameters

- **cellX** The X coordinate of the cell corresponding to the map tile.
- **cellY** The Y coordinate of the cell corresponding to the map tile.
- **filePath** The file path of the loaded file.

Returns

void

static void **SendClientScriptGlobal** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a ClientScriptGlobal packet with the current client script globals in the write-only worldstate.

Parameters

- **pid** The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

static void **SendClientScriptSettings** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a ClientScriptSettings packet with the current client script settings in the write-only worldstate.

Parameters

- pid The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendWorldKillCount** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a WorldKillCount packet with the current set of kill count changes in the write-only worldstate.

Parameters

- **pid** The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendWorldRegionAuthority**(unsigned short pid) noexcept

Send a WorldRegionAuthority packet establishing a certain player as the only one who should process certain region-specific events (such as weather changes).

It is always sent to all players.

Parameters

pid – The player ID attached to the packet.

Returns

void

static void **SendWorldMap** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a WorldMap packet with the current set of map changes in the write-only worldstate.

Parameters

- pid The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

static void **SendWorldTime** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept Send a WorldTime packet with the current time and time scale in the write-only worldstate.

Parameters

- **pid** The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendWorldWeather** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a WorldWeather packet with the current weather in the write-only worldstate.

Parameters

- **pid** The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void **SendWorldCollisionOverride**(unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a WorldCollisionOverride packet with the current collision overrides in the write-only worldstate.

Parameters

- pid The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

Returns

void

static void SendCellReset (unsigned short pid, bool sendToOtherPlayers) noexcept

Send a CellReset packet with a list of cells,.

Parameters

pid – The player ID attached to the packet.

Returns

void

static void **SendWorldDestinationOverride** (unsigned short pid, bool sendToOtherPlayers, bool skipAttachedPlayer) noexcept

Send a WorldDestinationOverride packet with the current destination overrides in the write-only worldstate.

Parameters

- pid The player ID attached to the packet.
- **sendToOtherPlayers** Whether this packet should be sent to players other than the player attached to the packet (false by default).
- **skipAttachedPlayer** Whether the packet should skip being sent to the player attached to the packet (false by default).

INDEX

A	function), 5
ActorFunctions ($C++$ class), 3	${\tt ActorFunctions::GetActorKillerMpNum} \qquad (C++$
ActorFunctions::AddActor (C++ function), 17	function), 8
ActorFunctions::AddActorSpellActive (C++ function), 17	ActorFunctions::GetActorKillerName ($C++$ function), 8
ActorFunctions::AddActorSpellActiveEffect (C++ function), 17	ActorFunctions::GetActorKillerPid (C++ function), 7
ActorFunctions::ClearActorList(C++function), 3 ActorFunctions::CopyReceivedActorListToStore	ActorFunctions::GetActorKillerRefId $(C++function)$, 8
(C++function), 3	ActorFunctions::GetActorKillerRefNum $(C++function)$, 8
ActorFunctions::DoesActorHavePlayerKiller (C++ function), 7	ActorFunctions::GetActorListAction (C++ func- tion), 4
ActorFunctions::DoesActorHavePosition $(C++$ function), 12	ActorFunctions::GetActorListSize (C++ func-
ActorFunctions::DoesActorHaveStatsDynamic (C++ function), 12	tion), 4 ActorFunctions::GetActorMagickaBase $(C++$
ActorFunctions::DoesActorSpellsActiveHavePlay $(C++function)$, 11	erCaster <i>function</i>), 6 ActorFunctions::GetActorMagickaCurrent (<i>C</i> ++
ActorFunctions::EquipActorItem (C++ function),	function), 6
16 ActorFunctions::GetActorCell(C++ function), 4	ActorFunctions::GetActorMagickaModified (C++ function), 6
	ActorFunctions::GetActorMpNum(C++ function), 4
ActorFunctions::GetActorDeathState (C++ func- tion), 8	ActorFunctions::GetActorPosX ($C++$ function), 4
ActorFunctions::GetActorEquipmentItemCharge (C++ function), 7	ActorFunctions::GetActorPosY (C++ function), 4 ActorFunctions::GetActorPosZ (C++ function), 5
ActorFunctions::GetActorEquipmentItemCount (C++ function), 7	ActorFunctions::GetActorRefId $(C++function)$, 4 ActorFunctions::GetActorRefNum $(C++function)$, 4
ActorFunctions::GetActorEquipmentItemEnchantm $(C++function)$, 7	ActorFunctions::GetActorRotX(C++ function), 5 ActorFunctions::GetActorRotY(C++ function), 5
ActorFunctions::GetActorEquipmentItemRefId (C++ function), 6	ActorFunctions::GetActorRotZ(C++ function), 5 ActorFunctions::GetActorSpellsActiveCasterMpNum
ActorFunctions::GetActorFatigueBase $(C++$	(C++function), 11
function), 6	ActorFunctions::GetActorSpellsActiveCasterPid (C++ function), 11
ActorFunctions::GetActorFatigueCurrent $(C++$	ActorFunctions::GetActorSpellsActiveCasterRefId
function), 6	(C++function), 11
ActorFunctions::GetActorFatigueModified	ActorFunctions::GetActorSpellsActiveCasterRefNum
(C++ function), 6	(C++function), 11
ActorFunctions::GetActorHealthBase (C++ func- tion), 5	${\tt ActorFunctions::GetActorSpellsActiveChangesAction}$
ActorFunctions::GetActorHealthCurrent $(C++$	(C++ function), 8
function), 5	ActorFunctions::GetActorSpellsActiveChangesSize
ActorFunctions::GetActorHealthModified (C++	(C++function), 8

ActorFunctions::GetActorSpellsActiveDisplayNa	
(C++function), 9	${\tt ActorFunctions::SetActorDeathInstant} \qquad (C++$
${\tt ActorFunctions::GetActorSpellsActiveEffectArg}$	
(C++function), 10	$\verb ActorFunctions::SetActorDeathState (C++ \textit{func-}$
ActorFunctions::GetActorSpellsActiveEffectCou	
(C++function), 9	${\tt ActorFunctions::SetActorFatigueBase} \qquad (C++$
ActorFunctions::GetActorSpellsActiveEffectDur	
(C++function), 10	${\tt ActorFunctions::SetActorFatigueCurrent}\ (C++$
ActorFunctions::GetActorSpellsActiveEffectId	function), 14
(C++function), 9	ActorFunctions::SetActorFatigueModified
ActorFunctions::GetActorSpellsActiveEffectMag	
(C++function), 10	ActorFunctions::SetActorHealthBase (C++ func-
${\tt ActorFunctions::GetActorSpellsActiveEffectTim}$	
(C++function), 10	${\tt ActorFunctions::SetActorHealthCurrent} (C++$
$\verb ActorFunctions::GetActorSpellsActiveId (C++$	function), 13
function), 9	${\tt ActorFunctions::SetActorHealthModified}\ (C++$
${\tt ActorFunctions::GetActorSpellsActiveStackingS}$	tate function), 14
(C++function), 9	$\verb ActorFunctions::SetActorListAction (C++ \textit{func-}$
ActorFunctions::ReadCellActorList (C++ func-	tion), 12
<i>tion</i>), 3	$\verb ActorFunctions::SetActorListCell (C++ \textit{ func-}$
${\tt ActorFunctions::ReadReceivedActorList} (C++$	tion), 12
function), 3	${\tt ActorFunctions::SetActorListPid}\ (C++\textit{function}),$
ActorFunctions::SendActorAI (C++ function), 19	3
ActorFunctions::SendActorAuthority (C++ func-	${\tt ActorFunctions::SetActorMagickaBase} \qquad (C++$
tion), 18	function), 14
${\tt ActorFunctions::SendActorCellChange} \qquad (C++$	${\tt ActorFunctions::SetActorMagickaCurrent} \ \ (C++$
function), 19	function), 14
ActorFunctions::SendActorDeath (C++ function), 19	ActorFunctions::SetActorMagickaModified $(C++function)$, 14
ActorFunctions::SendActorEquipment (C++ func-	ActorFunctions::SetActorMpNum($C++function$), 13
tion), 18	${\tt ActorFunctions::SetActorPosition} (C++ \textit{func-}$
ActorFunctions::SendActorList($C++function$), 17	<i>tion</i>), 13
ActorFunctions::SendActorPosition (C++ func-	ActorFunctions::SetActorRefId($C++function$), 12
tion), 18	ActorFunctions::SetActorRefNum $(C++ function)$,
<pre>ActorFunctions::SendActorSpeech (C++ function),</pre>	13
19	${\tt ActorFunctions::SetActorRotation} (C++ \textit{func-}$
ActorFunctions::SendActorSpellsActiveChanges	<i>tion</i>), 13
(C++ function), 18	ActorFunctions::SetActorSound($C++function$), 15
ActorFunctions::SendActorStatsDynamic $(C++function)$, 18	ActorFunctions::SetActorSpellsActiveAction $(C++function)$, 15
ActorFunctions::SetActorAIAction (C++ func-	ActorFunctions::UnequipActorItem (C++ func-
tion), 15	tion), 17
ActorFunctions::SetActorAICoordinates $(C++$	_
function), 16	В
ActorFunctions::SetActorAIDistance (C++ func-	BookFunctions ($C++$ class), 20
tion), 16	BookFunctions::AddBook (C++ function), 20
ActorFunctions::SetActorAIDuration (C++ func-	BookFunctions::ClearBookChanges (C++ function),
tion), 16	20
ActorFunctions::SetActorAIRepetition $(C++$	BookFunctions::GetBookChangesSize (C++ func-
function), 16	tion), 20
ActorFunctions::SetActorAITargetToObject	BookFunctions::GetBookId (C++ function), 20
(C++ function), 15	BookFunctions::SendBookChanges (C++ function),
ActorFunctions::SetActorAITargetToPlayer	20
(C++ function) 15	20

C	ChatFunctions::CleanChatForPid (C++ function),
CellFunctions ($C++$ $class$), 21	Charles and Card Manager (Charles and Charles and Char
CellFunctions::GetCell(C++ function), 21	ChatFunctions::SendMessage (C++ function), 26
CellFunctions::GetCellStateChangesSize $(C++function)$, 21	D
CellFunctions::GetCellStateDescription $(C++$	DialogueFunctions ($C++$ $class$), 27
function), 21	DialogueFunctions::AddTopic (C++ function), 27
CellFunctions::GetCellStateType (<i>C</i> ++ <i>function</i>),	<pre>DialogueFunctions::ClearTopicChanges (C++ function), 27</pre>
CellFunctions::GetExteriorX (C++ function), 22	DialogueFunctions::GetTopicChangesSize (C++
CellFunctions::GetExteriorY (C++ function), 22	function), 27
CellFunctions::GetRegion (C++ function), 22	DialogueFunctions::GetTopicId(C++function), 27
CellFunctions::IsChangingRegion $(C++function)$,	DialogueFunctions::PlayAnimation (C++ function), 28
CellFunctions::IsInExterior (C++ function), 22	DialogueFunctions::PlaySpeech(C++function), 28
CellFunctions::SendCell (C++ function), 23	DialogueFunctions::SendTopicChanges (C++
CellFunctions::SetCell (C++ function), 22	function), 28
CellFunctions::SetExteriorCell (C++ function),	
23	F
CharClassFunctions (C++ class), 23	FactionFunctions ($C++$ class), 29
CharClassFunctions::GetClassDesc (C++ func-	FactionFunctions::AddFaction (C++ function), 31
tion), 23	FactionFunctions::ClearFactionChanges (C++
CharClassFunctions::GetClassMajorAttribute	function), 29
(C++ function), 24	FactionFunctions::GetFactionChangesAction
CharClassFunctions::GetClassMajorSkill $(C++$	(C++ function), 29
function), 24	FactionFunctions::GetFactionChangesSize
CharClassFunctions::GetClassMinorSkill $(C++$	(C++function), 29
function), 24	FactionFunctions::GetFactionExpulsionState
CharClassFunctions::GetClassName (C++ func-	(C++ function), 29
tion), 23	FactionFunctions::GetFactionId (C++ function),
CharClassFunctions::GetClassSpecialization	29
(C++ function), 24	FactionFunctions::GetFactionRank (C++ func-
CharClassFunctions::GetDefaultClass (C++	tion), 29
function), 23	FactionFunctions::GetFactionReputation (C++
CharClassFunctions::IsClassDefault (C++ func-	function), 30
tion), 24	FactionFunctions::SendFactionChanges (C++
CharClassFunctions::SendClass(C++ function), 26	function), 31
CharClassFunctions::SetClassDesc (C++ func-	FactionFunctions::SetFactionChangesAction
tion), 25	(C++ function), 30
CharClassFunctions::SetClassMajorAttribute	FactionFunctions::SetFactionExpulsionState
(C++ function), 25	(C++ function), 30
CharClassFunctions::SetClassMajorSkill (C++	FactionFunctions::SetFactionId (C++ function),
function), 25	30
CharClassFunctions::SetClassMinorSkill (C++	FactionFunctions::SetFactionRank (C++ func-
function), 26	tion), 30
CharClassFunctions::SetClassName (C++ func-	FactionFunctions::SetFactionReputation $(C++$
tion), 25	function), 30
CharClassFunctions::SetClassSpecialization	•
(C++ function), 25	G
CharClassFunctions::SetDefaultClass (C++	GUIFunctions ($C++$ class), 31
function), 24	GUIFunctions::_MessageBox ($C++$ function), 31
ChatFunctions ($C++$ class), 26	GUIFunctions::AddQuickKey (C++ function), 33
ChatFunctions::CleanChat (C++ function), 27	GUIFunctions::ClearQuickKeyChanges (C++ func-
· •	tion), 32

GUIFunctions::CustomMessageBox (C++ function),	<pre>ItemFunctions::GetUsedItemCount (C++ function), 38</pre>
GUIFunctions::GetQuickKeyChangesSize (C++ function), 33	<pre>ItemFunctions::GetUsedItemEnchantmentCharge</pre>
GUIFunctions::GetQuickKeyItemId(C++ function), 33	<pre>ItemFunctions::GetUsedItemRefId(C++ function), 38</pre>
GUIFunctions::GetQuickKeySlot(C++function), 33 GUIFunctions::GetQuickKeyType(C++function), 33	<pre>ItemFunctions::GetUsedItemSoul (C++ function), 39</pre>
GUIFunctions::InputDialog (C++ function), 32 GUIFunctions::ListBox (C++ function), 32	<pre>ItemFunctions::HasItemEquipped (C++ function), 36</pre>
GUIFunctions::PasswordDialog(C++ function), 32 GUIFunctions::SendQuickKeyChanges(C++ function), 34	<pre>ItemFunctions::SendEquipment (C++ function), 39 ItemFunctions::SendInventoryChanges (C++</pre>
GUIFunctions::SetMapVisibility (C++ function), 34	<pre>ItemFunctions::SendItemUse (C++ function), 39 ItemFunctions::SetInventoryChangesAction</pre>
GUIFunctions::SetMapVisibilityAll (C++ func-tion), 34	(C++ function), 35 ItemFunctions::UnequipItem(C++ function), 36
	M
ItemFunctions (C++ class), 34 ItemFunctions::AddItemChange (C++ function), 36	MechanicsFunctions (C++ class), 40
ItemFunctions::ClearInventoryChanges $(C++$	MechanicsFunctions::AddAlliedPlayerForPlayer (C++ function), 43
<pre>function), 35 ItemFunctions::EquipItem(C++function), 35</pre>	MechanicsFunctions::ClearAlliedPlayersForPlayer (C++ function), 40
<pre>ItemFunctions::GetEquipmentChangesSize (C++ function), 35</pre>	MechanicsFunctions::DoesPlayerHavePlayerKiller (C++ function), 41
<pre>ItemFunctions::GetEquipmentChangesSlot (C++ function), 36</pre>	MechanicsFunctions::GetDrawState (C++ function), 42
<pre>ItemFunctions::GetEquipmentItemCharge (C++ function), 37</pre>	MechanicsFunctions::GetMarkCell(C++ function), 40
<pre>ItemFunctions::GetEquipmentItemCount (C++ function), 37</pre>	<pre>MechanicsFunctions::GetMarkPosX(C++ function), 40</pre>
	arWechanicsFunctions::GetMarkPosY(C++ function),
	<pre>MechanicsFunctions::GetMarkPosZ(C++ function), 41</pre>
	<pre>MechanicsFunctions::GetMarkRotX(C++ function), 41</pre>
<pre>ItemFunctions::GetInventoryChangesAction</pre>	<pre>MechanicsFunctions::GetMarkRotZ(C++ function), 41</pre>
<pre>ItemFunctions::GetInventoryChangesSize (C++ function), 35</pre>	MechanicsFunctions::GetMiscellaneousChangeType (C++ function), 40
<pre>ItemFunctions::GetInventoryItemCharge (C++ function), 38</pre>	MechanicsFunctions::GetPlayerKillerMpNum (C++ function), 42
· ·	MechanicsFunctions::GetPlayerKillerName (C++ function), 42
	artyechanicsFunctions::GetPlayerKillerPid (C++ function), 41
$ \begin{tabular}{ll} ItemFunctions::GetInventoryItemRefId & (C++) \end{tabular} $	MechanicsFunctions::GetPlayerKillerRefId
<pre>function), 37 ItemFunctions::GetInventoryItemSoul (C++)</pre>	(C++ function), 41 MechanicsFunctions::GetPlayerKillerRefNum
<pre>function), 38 ItemFunctions::GetUsedItemCharge (C++ func- tion), 39</pre>	(C++ function), 42 MechanicsFunctions::GetSelectedSpellId (C++
uon), 59	function), 41

MechanicsFunctions::GetSneakState (C++ function), 42	<pre>ObjectFunctions::GetClientLocalInternalIndex</pre>
MechanicsFunctions::Jail (C++ function), 44	ObjectFunctions::GetClientLocalIntValue
MechanicsFunctions::Resurrect $(C++function)$, 44	(C++ function), 55
MechanicsFunctions::SendAlliedPlayers $(C++$	ObjectFunctions::GetClientLocalsSize (C++
function), 44	function), 55
MechanicsFunctions::SendMarkLocation $(C++)$	ObjectFunctions::GetClientLocalVariableType
function), 43	(C++ function), 55
function(0), 43 MechanicsFunctions::SendSelectedSpell ($C++$	ObjectFunctions::GetContainerChangesSize
function), 44	(C++ function), 56
<pre>MechanicsFunctions::SetMarkCell(C++ function), 42</pre>	ObjectFunctions::GetContainerItemActionCount (C++ function), 57
<pre>MechanicsFunctions::SetMarkPos (C++ function), 43</pre>	<pre>ObjectFunctions::GetContainerItemCharge</pre>
<pre>MechanicsFunctions::SetMarkRot (C++ function),</pre>	ObjectFunctions::GetContainerItemCount (C++
43	function), 56
<pre>MechanicsFunctions::SetSelectedSpellId (C++</pre>	ObjectFunctions::GetContainerItemEnchantmentCharge
function), 43	(C++ function), 56
MiscellaneousFunctions ($C++$ class), 45	ObjectFunctions::GetContainerItemRefId (C++
MiscellaneousFunctions::GenerateRandomString	function), 56
(C++ function), 45 MiscellaneousFunctions::GetCurrentMpNum	ObjectFunctions::GetContainerItemSoul (C++ function), 57
(C++function), 45	ObjectFunctions::GetObjectActivatingMpNum
MiscellaneousFunctions::GetLastPlayerId	(C++ function), 51
(C++ function), 45	ObjectFunctions::GetObjectActivatingName
MiscellaneousFunctions::GetSHA256Hash $(C++)$	
	(C++ function), 51
function), 45	ObjectFunctions::GetObjectActivatingPid
MiscellaneousFunctions::SetCurrentMpNum	(C++ function), 50
(C++ function), 45	ObjectFunctions::GetObjectActivatingRefId
0	(C++ function), 50
O	ObjectFunctions::GetObjectActivatingRefNum
ObjectFunctions ($C++$ class), 46	(C++ function), 50
ObjectFunctions::AddClientLocalFloat (C++ function), 65	ObjectFunctions::GetObjectCharge (C++ function), 48
ObjectFunctions::AddClientLocalInteger (C++	<pre>ObjectFunctions::GetObjectCount (C++ function),</pre>
function), 65	48
ObjectFunctions::AddContainerItem (C++ func- tion), 65	ObjectFunctions::GetObjectDialogueChoiceTopic (C++ function), 49
ObjectFunctions::AddObject (C++ function), 65	ObjectFunctions::GetObjectDialogueChoiceType
ObjectFunctions::ClearObjectList (C++ func-	(C++function), 49
	ObjectFunctions::GetObjectDoorState (C++
tion), 46	
ObjectFunctions::CopyReceivedObjectListToStor (C++ function), 46	ObjectFunctions::GetObjectEnchantmentCharge
	(C++ function), 48
ObjectFunctions::DoesObjectHaveContainer	ObjectFunctions::GetObjectGoldPool (C++ func-
(C++ function), 57	
ObjectFunctions::DoesObjectHavePlayerActivati	ng ### ### ng #########################
(C++ function), 50	function), 48
ObjectFunctions::DoesObjectHavePlayerHitting	ObjectFunctions::GetObjectHitBlock (C++ func-
(C++function), 51	
ObjectFunctions::DoesObjectHavePlayerSummoner	tion), 51
(C++ function), 53	ObjectFunctions::GetObjectHitDamage (C++
ObjectFunctions::GetClientLocalFloatValue	function), 51
(C++ function), 55	ObjectFunctions::GetObjectHitKnockdown (C++ function) 51
	HANCHONI)) I

ObjectFunctions::GetObjectHitSuccess	(<i>C</i> ++	tion), 49	
function), 51		ObjectFunctions::GetObjectState(C++ fun	ction),
ObjectFunctions::GetObjectHittingMpNum	(<i>C</i> ++	49	
function), 52		ObjectFunctions::GetObjectSummonDuratio	n
ObjectFunctions::GetObjectHittingName	(C++	(C++function), 53	
function), 52		ObjectFunctions::GetObjectSummonEffectI	d
ObjectFunctions::GetObjectHittingPid	(C++	(C++function), 52	
function), 52		ObjectFunctions::GetObjectSummonerMpNum	ı
ObjectFunctions::GetObjectHittingRefId	(<i>C</i> ++	(C++ function), 53	
function), 52		ObjectFunctions::GetObjectSummonerPid	(C++
ObjectFunctions::GetObjectHittingRefNum	1	function), 53	(
(C++ function), 52		ObjectFunctions::GetObjectSummonerRefId	I
ObjectFunctions::GetObjectLastGoldResto	ckDav	(C++ function), 53	=
(C++function), 50	cnbuy	ObjectFunctions::GetObjectSummonerRefNu	ım
ObjectFunctions::GetObjectLastGoldResto	ckHour		
(C++function), 50	CKIIOUI	ObjectFunctions::GetObjectSummonSpellId	ı
ObjectFunctions::GetObjectListAction	(<i>C</i> ++	(C++function), 53	L
function), 47	(СТТ	ObjectFunctions::GetObjectSummonState	$(C \cup I)$
	int		(С++
ObjectFunctions::GetObjectListClientScr	ıpı	function), 52	<i>C</i>
(C++function), 47	,	ObjectFunctions::GetVideoFilename ($C++$	јипс-
ObjectFunctions::GetObjectListConsoleCo	mmana	tion), 54	
(C++ function), 47		ObjectFunctions::IsObjectDroppedByPlaye	er
ObjectFunctions::GetObjectListContainer	SubAct	· · · · · · · · · · · · · · · · · · ·	
(<i>C</i> ++ <i>function</i>), 47		ObjectFunctions::IsObjectPlayer(C++ fun	ction),
ObjectFunctions::GetObjectListOrigin	(C++	47	
function), 46		ObjectFunctions::ReadReceivedObjectList	-
ObjectFunctions::GetObjectListSize $(C++$	- func-	(C++function), 46	
tion), 46		ObjectFunctions::SendClientScriptLocal	(C++
ObjectFunctions::GetObjectLockLevel	(C++	function), 69	
function), 49		ObjectFunctions::SendConsoleCommand	(C++
ObjectFunctions::GetObjectMpNum ($C++fun$	ction),	function), 70	
48		ObjectFunctions::SendContainer ($C++$ fun	ction),
ObjectFunctions::GetObjectPid(C++function	(200), 47	69	
ObjectFunctions::GetObjectPosX ($C++ fun$	ction),	ObjectFunctions::SendDoorDestination	(C++
54		function), 69	
ObjectFunctions::GetObjectPosY $(C++ function)$	ction),	ObjectFunctions::SendDoorState (C++ fun	ction),
54		68	
ObjectFunctions::GetObjectPosZ (C++ fun	ction),	ObjectFunctions::SendObjectActivate	(C++
54	, ,	function), 65	`
ObjectFunctions::GetObjectRefId(C++ fun	ction).	ObjectFunctions::SendObjectDelete (C++	func-
47	,,	tion), 66	juite
ObjectFunctions::GetObjectRefNum $(C++$	func-	ObjectFunctions::SendObjectDialogueChoi	CE
tion), 47	junc	(C++ function), 66	
ObjectFunctions::GetObjectRotX (C++ fun	ction)	ObjectFunctions::SendObjectLock $(C++fun$	ction)
54	ciion),	66	cuon),
ObjectFunctions::GetObjectRotY ($C++$ fun	ati an		
•	cuon),	ObjectFunctions::SendObjectMiscellaneou	ıs
54	\	(C++function), 67	\
ObjectFunctions::GetObjectRotZ ($C++ fun$	ction),	ObjectFunctions::SendObjectMove ($C++fun$	ction),
54		68	
ObjectFunctions::GetObjectScale ($C++fun$	ction),	ObjectFunctions::SendObjectPlace $(C++$	func-
49		tion), 66	
ObjectFunctions::GetObjectSoul ($C++ function$)	ction),	ObjectFunctions::SendObjectRestock (C++	- func-
48		tion), 67	
ObjectFunctions::GetObjectSoundId $(C++$	func-	${\tt ObjectFunctions::SendObjectRotate}\ (C++$	func-

tion), 68	function), 59
ObjectFunctions::SendObjectScale (C++ func-tion), 67	<pre>ObjectFunctions::SetObjectLastGoldRestockDay</pre>
ObjectFunctions::SendObjectSound (C++ func- tion), 68	ObjectFunctions::SetObjectLastGoldRestockHour (C++ function), 60
ObjectFunctions::SendObjectSpawn (C++ func- tion), 66	ObjectFunctions::SetObjectListAction (C++ function), 57
ObjectFunctions::SendObjectState (C++ func- tion), 68	<pre>ObjectFunctions::SetObjectListCell (C++ func- tion), 57</pre>
<pre>ObjectFunctions::SendObjectTrap (C++ function), 67</pre>	ObjectFunctions::SetObjectListConsoleCommand (C++ function), 58
ObjectFunctions::SendVideoPlay (C++ function), 69	ObjectFunctions::SetObjectListContainerSubAction (C++ function), 58
ObjectFunctions::SetContainerItemActionCountB (C++ function), 64	yOlmjkextFunctions::SetObjectListPid (C++ func- tion), 46
ObjectFunctions::SetContainerItemCharge (C++ function), 64	ObjectFunctions::SetObjectLockLevel (C++ function), 60
ObjectFunctions::SetContainerItemCount (C++ function), 64	ObjectFunctions::SetObjectMpNum(C++ function), 58
ObjectFunctions::SetContainerItemEnchantmentC (C++ function), 64	habrigectFunctions::SetObjectPosition (C++ func- tion), 61
ObjectFunctions::SetContainerItemRefId (C++ function), 64	58
ObjectFunctions::SetContainerItemSoul (C++ function), 64	ObjectFunctions::SetObjectRefNum (C++ function), 58
ObjectFunctions::SetObjectActivatingPid (C++ function), 62	ObjectFunctions::SetObjectRotation (<i>C++ func-tion</i>), 61
ObjectFunctions::SetObjectCharge (C++ function), 59	ObjectFunctions::SetObjectScale(C++ function), 59
ObjectFunctions::SetObjectCount (C++ function), 59	59
ObjectFunctions::SetObjectDialogueChoiceTopic (C++ function), 60	60
ObjectFunctions::SetObjectDialogueChoiceType (C++ function), 60	ObjectFunctions::SetObjectSummonDuration (C++function), 62
function), 61	ObjectFunctions::SetObjectSummonEffectId (C++ function), 61
ObjectFunctions::SetObjectDoorDestinationCell (C++ function), 63	ObjectFunctions::SetObjectSummonerMpNum (C++ function), 62
ObjectFunctions::SetObjectDoorDestinationPosi (C++ function), 63	tdlmjectFunctions::SetObjectSummonerPid (C++ function), 62
ObjectFunctions::SetObjectDoorDestinationRota (C++ function), 63	tClopiectFunctions::SetObjectSummonerRefNum (C++ function), 62
ObjectFunctions::SetObjectDoorState (C++ function), 63	ObjectFunctions::SetObjectSummonSpellId (C++ function), 62
ObjectFunctions::SetObjectDoorTeleportState (C++ function), 63	ObjectFunctions::SetObjectSummonState (C++ function), 61
ObjectFunctions::SetObjectDroppedByPlayerStat (C++ function), 61	eObjectFunctions::SetPlayerAsObject (C++ func- tion), 64
ObjectFunctions::SetObjectEnchantmentCharge (C++ function), 59	Р
ObjectFunctions::SetObjectGoldPool (C++ func- tion), 60	PositionFunctions (C++ class), 70 PositionFunctions::GetPosX (C++ function), 70
ObjectFunctions::SetObjectGoldValue (C++	PositionFunctions::GetPosV (C++ function) 70

PositionFunctions::GetPosZ(C++ function), 70 PositionFunctions::GetPreviousCellPosX(C++	RecordsDynamicFunctions::ClearRecords (C++ function), 76
<pre>function), 70 PositionFunctions::GetPreviousCellPosY (C++</pre>	RecordsDynamicFunctions::GetRecordAutoCalc (C++ function), 77
function), 71	RecordsDynamicFunctions::GetRecordBaseId
PositionFunctions::GetPreviousCellPosZ (C++	(C++function), 76
function), 71	RecordsDynamicFunctions::GetRecordCharge
PositionFunctions::GetRotX(C++ function), 71	(C++function), 78
PositionFunctions::GetRotX (C++ function), 71	RecordsDynamicFunctions::GetRecordCost (C++
PositionFunctions::SendMomentum $(C++function)$,	function), 78
77	RecordsDynamicFunctions::GetRecordCount
PositionFunctions::SendPos(C++function),72	(C++ function), 76
PositionFunctions::SetMomentum $(C++ function)$,	RecordsDynamicFunctions::GetRecordEffectArea
72	
· —	(C++ function), 79
PositionFunctions::SetPos (C++ function), 71	RecordsDynamicFunctions::GetRecordEffectAttribute
PositionFunctions::SetRot(C++ function),71	(C++ function), 79
Q	RecordsDynamicFunctions::GetRecordEffectCount
	(C++function), 76
QuestFunctions ($C++$ class), 73	RecordsDynamicFunctions::GetRecordEffectDuration
QuestFunctions::AddJournalEntry(C++function),	(C++function), 79
73	RecordsDynamicFunctions::GetRecordEffectId
QuestFunctions::AddJournalEntryWithTimestamp	(C++function), 78
(C++function), 73	Records Dynamic Functions:: Get Record Effect Magnitude Max
QuestFunctions::AddJournalIndex(C++ function),	(C++function), 80
74	RecordsDynamicFunctions::GetRecordEffectMagnitudeMin
QuestFunctions::ClearJournalChanges (C++	(C++function), 80
function), 73	RecordsDynamicFunctions::GetRecordEffectRangeType
QuestFunctions::GetJournalChangesSize (C++	(C++function), 79
function), 73	RecordsDynamicFunctions::GetRecordEffectSkill
QuestFunctions::GetJournalItemActorRefId	(C++function), 79
(C++ function), 74	RecordsDynamicFunctions::GetRecordEnchantmentCharge
QuestFunctions::GetJournalItemIndex $(C++)$	(C++ function), 77
function), 74	RecordsDynamicFunctions::GetRecordEnchantmentId
QuestFunctions::GetJournalItemQuest $(C++$	(C++ function), 77
function), 74	RecordsDynamicFunctions::GetRecordFlags
QuestFunctions::GetJournalItemType ($C++$ func-	(C++ function), 78
tion), 74	RecordsDynamicFunctions::GetRecordIcon (C++
QuestFunctions::GetReputation $(C++function)$, 75	function), 77
QuestFunctions::SendJournalChanges ($C++$ func-	RecordsDynamicFunctions::GetRecordId (C++
tion), 75	function), 76
QuestFunctions::SendReputation $(C++ function)$,	RecordsDynamicFunctions::GetRecordModel
questrailectionssenakeputation $(C++)$ junction),	(C++ function), 77
QuestFunctions::SetReputation(C++function),74	RecordsDynamicFunctions::GetRecordName (C++
questrunctionssetkeputation(C++junction), 74	function), 77
R	RecordsDynamicFunctions::GetRecordQuantity
	(C++ function), 78
RecordsDynamicFunctions ($C++$ $class$), 76	RecordsDynamicFunctions::GetRecordScript
RecordsDynamicFunctions::AddRecord (C++ func-	(C++ function), 77
tion), 91	RecordsDynamicFunctions::GetRecordSubtype
RecordsDynamicFunctions::AddRecordBodyPart	(C++ function), 76
(C++ function), 92	RecordsDynamicFunctions::GetRecordType (C++
RecordsDynamicFunctions::AddRecordEffect	function), 76
(C++ function), 92	RecordsDynamicFunctions::CatPacardValue
RecordsDynamicFunctions::AddRecordInventoryIt	(C++ function), 78
(C++function), 92	(CTTJuncuon), 70

```
RecordsDynamicFunctions::GetRecordWeight
                                                                                                         RecordsDynamicFunctions::SetRecordEffectRangeType
                                                                                                                           (C++function), 90
                 (C++ function), 78
RecordsDynamicFunctions::SendRecordDynamic
                                                                                                         RecordsDynamicFunctions::SetRecordEffectSkill
                 (C++ function), 92
                                                                                                                          (C++function), 90
RecordsDynamicFunctions::SetRecordAIAlarm
                                                                                                         RecordsDynamicFunctions::SetRecordEnchantmentCharge
                 (C++ function), 87
                                                                                                                          (C++ function), 82
RecordsDynamicFunctions::SetRecordAIFight
                                                                                                         RecordsDynamicFunctions::SetRecordEnchantmentId
                 (C++ function), 87
                                                                                                                           (C++ function), 81
RecordsDynamicFunctions::SetRecordAIFlee
                                                                                                         RecordsDynamicFunctions::SetRecordEnchantmentIdByIndex
                 (C++ function), 87
                                                                                                                          (C++function), 89
RecordsDynamicFunctions::SetRecordAIServices RecordsDynamicFunctions::SetRecordFaction
                 (C++ function), 87
                                                                                                                          (C++ function), 86
RecordsDynamicFunctions::SetRecordArmorRating RecordsDynamicFunctions::SetRecordFatigue
                 (C++ function), 83
                                                                                                                           (C++ function), 87
RecordsDynamicFunctions::SetRecordAutoCalc
                                                                                                         RecordsDynamicFunctions::SetRecordFlags
                 (C++ function), 82
                                                                                                                           (C++function), 82
RecordsDynamicFunctions::SetRecordBaseId
                                                                                                         RecordsDynamicFunctions::SetRecordFloatVariable
                 (C++ function), 80
                                                                                                                          (C++ function), 89
RecordsDynamicFunctions::SetRecordBloodType
                                                                                                         RecordsDynamicFunctions::SetRecordGender
                 (C++ function), 86
                                                                                                                           (C++ function), 85
RecordsDynamicFunctions::SetRecordBodyPartIdFoReFeemadssDynamicFunctions::SetRecordHair (C++
                 (C++ function), 91
                                                                                                                          function), 85
Records Dynamic Functions:: Set Record Body Part Id Follows a discount of the conditions:: Set Record Head (C++) and the conditions are the conditions and the conditions are the conditions and the conditions are the cond
                 (C++ function), 91
                                                                                                                          function), 85
RecordsDynamicFunctions::SetRecordBodyPartTypeRecordsDynamicFunctions::SetRecordHealth
                 (C++ function), 91
                                                                                                                          (C++ function), 83
RecordsDynamicFunctions::SetRecordCharge
                                                                                                         {\tt RecordsDynamicFunctions::SetRecordIcon} \ \ (C++
                 (C++ function), 82
                                                                                                                          function), 81
RecordsDynamicFunctions::SetRecordClass
                                                                                                         RecordsDynamicFunctions::SetRecordId
                                                                                                                                                                                                   (C++
                 (C++ function), 86
                                                                                                                          function), 80
RecordsDynamicFunctions::SetRecordCloseSound RecordsDynamicFunctions::SetRecordIdByIndex
                 (C++ function), 88
                                                                                                                          (C++ function), 89
RecordsDynamicFunctions::SetRecordColor
                                                                                                         RecordsDynamicFunctions::SetRecordIntegerVariable
                 (C++ function), 83
                                                                                                                          (C++function), 89
RecordsDynamicFunctions::SetRecordCost (C++ RecordsDynamicFunctions::SetRecordInventoryBaseId
                 function), 82
                                                                                                                          (C++function), 80
RecordsDynamicFunctions::SetRecordDamageChop RecordsDynamicFunctions::SetRecordInventoryItemCount
                 (C++function), 84
                                                                                                                          (C++function), 91
RecordsDynamicFunctions::SetRecordDamageSlash RecordsDynamicFunctions::SetRecordInventoryItemId
                 (C++function), 84
                                                                                                                          (C++function), 91
RecordsDynamicFunctions::SetRecordDamageThrustRecordsDynamicFunctions::SetRecordKeyState
                 (C++ function), 84
                                                                                                                           (C++ function), 84
RecordsDynamicFunctions::SetRecordEffectArea RecordsDynamicFunctions::SetRecordLevel
                 (C++ function), 90
                                                                                                                          (C++ function), 86
RecordsDynamicFunctions::SetRecordEffectAttribRuteordsDynamicFunctions::SetRecordMagicka
                 (C++ function), 90
                                                                                                                           (C++ function), 87
RecordsDynamicFunctions::SetRecordEffectDuratiRecordsDynamicFunctions::SetRecordMaxRange
                 (C++ function), 90
                                                                                                                          (C++ function), 88
RecordsDynamicFunctions::SetRecordEffectId
                                                                                                         RecordsDynamicFunctions::SetRecordMinRange
                 (C++ function), 90
                                                                                                                           (C++ function), 88
RecordsDynamicFunctions::SetRecordEffectMagnitRuckeMadksDynamicFunctions::SetRecordModel
                 (C++ function), 90
                                                                                                                          (C++ function), 81
{\tt RecordsDynamicFunctions::SetRecordEffectMagnitRecordIndesDynamicFunctions::SetRecordName \ (\textit{C++} in the content of the 
                 (C++ function), 91
                                                                                                                          function), 81
```

RecordsDynamicFunctions::SetRecordOpenSound (C++ function), 88	ServerFunctions::GetArchitectureType (C++ function), 94
RecordsDynamicFunctions::SetRecordQuality (C++ function), 83	ServerFunctions::GetAvgPing (C++ function), 95 ServerFunctions::GetCaseInsensitiveFilename
RecordsDynamicFunctions::SetRecordRace $(C++$	(C++function), 94
function), 86	ServerFunctions::GetDataFileEnforcementState
RecordsDynamicFunctions::SetRecordRadius	(C++function), 95
(C++ function), 83	ServerFunctions::GetDataPath (C++ function), 94
RecordsDynamicFunctions::SetRecordReach	ServerFunctions::GetIP (C++ function), 95
(C++ function), 84	ServerFunctions::GetMaxPlayers (C++ function),
RecordsDynamicFunctions::SetRecordScale	95
(C++ function), 86	ServerFunctions::GetMillisecondsSinceServerStart
RecordsDynamicFunctions::SetRecordScript	(C++ function), 94
(C++ function), 81	ServerFunctions::GetOperatingSystemType
RecordsDynamicFunctions::SetRecordScriptText	(C++function), 94
(C++ function), 89	ServerFunctions::GetPort (C++ function), 95
RecordsDynamicFunctions::SetRecordScrollState	
-	
(C++ function), 85 RecordsDynamicFunctions::SetRecordSkillId	function), 94
(C++ function), 85	ServerFunctions::GetScriptErrorIgnoringState (C++ function), 95
RecordsDynamicFunctions::SetRecordSoulValue (C++ function), 87	ServerFunctions::GetServerVersion (C++ function), 94
RecordsDynamicFunctions::SetRecordSound	ServerFunctions::HasPassword(C++ function), 95
(C++function), 88	ServerFunctions::Kick (C++ function), 93
RecordsDynamicFunctions::SetRecordSpeed	ServerFunctions::LogAppend (C++ function), 93
(C++ function), 84	ServerFunctions::LogMessage (C++ function), 93
RecordsDynamicFunctions::SetRecordStringVaria (C++ function), 89	
RecordsDynamicFunctions::SetRecordSubtype	ServerFunctions::SetGameMode (C++ function), 95
(C++ function), 81	ServerFunctions::SetHostname(C++ function), 96
RecordsDynamicFunctions::SetRecordText $(C++)$	ServerFunctions::SetRuleString (C++ function),
function), 85	96
RecordsDynamicFunctions::SetRecordTime $(C++$	ServerFunctions::SetRuleValue(C++function),96
function), 83	ServerFunctions::SetScriptErrorIgnoringState
RecordsDynamicFunctions::SetRecordType $(C++$	(C++function), 96
function), 80	ServerFunctions::SetServerPassword (C++ func-
RecordsDynamicFunctions::SetRecordUses $(C++$	tion), 96
function), 83	ServerFunctions::StopServer(C++function), 93
RecordsDynamicFunctions::SetRecordValue	ServerFunctions::UnbanAddress(C++function),94
(C++function), 82	SettingFunctions ($C++$ class), 97
<pre>RecordsDynamicFunctions::SetRecordVampireStat</pre>	<pre>eSettingFunctions::ClearGameSettingValues</pre>
RecordsDynamicFunctions::SetRecordVolume	SettingFunctions::ClearVRSettingValues (C++
(C++ function), 88	function), 99
RecordsDynamicFunctions::SetRecordWeight	SettingFunctions::SendSettings (C++ function),
(C++ function), 82	99
S	SettingFunctions::SetBedRestAllowed $(C++function)$, 98
ServerFunctions ($C++$ class), 93	SettingFunctions::SetConsoleAllowed $(C++$
ServerFunctions::AddDataFileRequirement	function), 98
(C++ function), 97	SettingFunctions::SetDifficulty(C++ function),
ServerFunctions::BanAddress (C++ function), 93	97
ServerFunctions::DoesFilePathExist (C++ func-	SettingFunctions::SetEnforcedLogLevel (C++
tion). 94	function), 97

SettingFunctions::SetGameSettingValue	(C++	(C++function), 102
function), 99		SpellFunctions::GetSpellbookChangesSize
SettingFunctions::SetPhysicsFramerate	(C++	(C++function), 102
function), 97		SpellFunctions::GetSpellId (C++ function), 104
SettingFunctions::SetVRSettingValue	(C++	SpellFunctions::GetSpellsActiveCasterMpNum
function), 99		(C++function), 107
${\tt SettingFunctions::SetWaitAllowed} (C++$	func-	SpellFunctions::GetSpellsActiveCasterPid
tion), 98		(C++function), 107
SettingFunctions::SetWildernessRestAllo	wed	<pre>SpellFunctions::GetSpellsActiveCasterRefId</pre>
(C++ function), 98		(C++function), 107
ShapeshiftFunctions (C++ class), 100		SpellFunctions::GetSpellsActiveCasterRefNum
ShapeshiftFunctions::GetCreatureNameDis	playSt	ate $(C++function)$, 107
(C++ function), 100		<pre>SpellFunctions::GetSpellsActiveChangesAction</pre>
ShapeshiftFunctions::GetCreatureRefId	(C++	(C++function), 103
function), 100		SpellFunctions::GetSpellsActiveChangesSize
ShapeshiftFunctions::GetScale (C++ fun	ction),	(C++function), 102
100		SpellFunctions::GetSpellsActiveDisplayName
ShapeshiftFunctions::IsWerewolf $(C++fun$	ction),	(C++function), 105
100		<pre>SpellFunctions::GetSpellsActiveEffectArg</pre>
ShapeshiftFunctions::SendShapeshift	(C++	(C++ function), 105
function), 101		SpellFunctions::GetSpellsActiveEffectCount
ShapeshiftFunctions::SetCreatureNameDis	playSt	ate $(C++function)$, 105
(C++ function), 101		${\tt SpellFunctions::GetSpellsActiveEffectDuration}$
ShapeshiftFunctions::SetCreatureRefId	(C++	(C++ function), 106
function), 101		SpellFunctions::GetSpellsActiveEffectId
ShapeshiftFunctions::SetScale (C++ fun	ction),	(C++function), 105
100		<pre>SpellFunctions::GetSpellsActiveEffectMagnitud</pre>
ShapeshiftFunctions::SetWerewolfState	(C++	(C++function), 106
function), 100		<pre>SpellFunctions::GetSpellsActiveEffectTimeLeft</pre>
SpellFunctions ($C++$ class), 102		(C++ function), 106
${\tt SpellFunctions::AddCooldownSpell} (C++$	func-	SpellFunctions::GetSpellsActiveId ($C++$ func-
tion), 104		tion), 105
SpellFunctions::AddSpell(C++ function), 10		SpellFunctions::GetSpellsActiveStackingState
SpellFunctions::AddSpellActive ($C++$ fun	ction),	(C++function), 105
103	(0	SpellFunctions::SendCooldownChanges $(C++$
SpellFunctions::AddSpellActiveEffect	(<i>C</i> ++	function), 108
function), 104	(6)	SpellFunctions::SendSpellbookChanges $(C++$
SpellFunctions::ClearCooldownChanges	(C++	function), 108
function), 102	(C 1	SpellFunctions::SendSpellsActiveChanges
SpellFunctions::ClearSpellbookChanges <i>function</i>), 102	(C++	(C++ function), 108
SpellFunctions::ClearSpellsActiveChange		SpellFunctions::SetSpellbookChangesAction $(C++function)$, 103
(C++function), 102	:5	SpellFunctions::SetSpellsActiveChangesAction
SpellFunctions::DoesSpellsActiveHavePla	war(2c	
(C++function), 106	iyer cas	StatsFunctions ($C++$ class), 109
SpellFunctions::GetCooldownChangesSize	(C++	StatsFunctions::ClearAttributeModifier (C++
function), 103	(011	function), 117
SpellFunctions::GetCooldownSpellId (C++	- func-	StatsFunctions::ClearSkillModifier (C++ func-
tion), 107	june	tion), 117
SpellFunctions::GetCooldownStartDay	(C++	StatsFunctions::GetAttributeBase (C++ func-
function), 107		tion), 112
SpellFunctions::GetCooldownStartHour	(<i>C</i> ++	StatsFunctions::GetAttributeCount (C++ func-
function), 108		tion), 109
SpellFunctions::GetSpellbookChangesActi	.on	${\tt StatsFunctions::GetAttributeDamage}\ (\textit{C++ func-}$

tion), 112	tion), 117
StatsFunctions::GetAttributeId (C++ function), 109	StatsFunctions::SetBirthsign(C++function), 115 StatsFunctions::SetBounty(C++function), 118
StatsFunctions::GetAttributeModifier $(C++function)$, 112	StatsFunctions::SetCharGenStage(C++ function), 118
StatsFunctions::GetAttributeName (C++ function), 109	StatsFunctions::SetFatigueBase ($C++$ function), 116
StatsFunctions::GetBirthsign(C++function), 111 StatsFunctions::GetBounty(C++function), 113	StatsFunctions::SetFatigueCurrent (C++ function), 116
StatsFunctions::GetFatigueBase ($C++$ function), 112	StatsFunctions::SetHairstyle(C++function), 114 StatsFunctions::SetHead(C++function), 114
StatsFunctions::GetFatigueCurrent (C++ function), 112	StatsFunctions::SetHealthBase (C++ function), 115
StatsFunctions::GetHairstyle(C++function), 110 StatsFunctions::GetHead(C++function), 110	StatsFunctions::SetHealthCurrent (C++ function), 116
StatsFunctions::GetHealthBase $(C++ function)$, 111	StatsFunctions::SetIsMale (C++ function), 114 StatsFunctions::SetLevel (C++ function), 115
StatsFunctions::GetHealthCurrent (C++ func-tion), 111	StatsFunctions::SetLevelProgress (C++ function), 115
StatsFunctions::GetIsMale(C++ function), 110 StatsFunctions::GetLevel(C++ function), 111	StatsFunctions::SetMagickaBase (C++ function), 116
StatsFunctions::GetLevelProgress (C++ function), 111	StatsFunctions::SetMagickaCurrent (C++ function), 116
StatsFunctions::GetMagickaBase $(C++ function)$, 111	StatsFunctions::SetModel (C++ function), 114 StatsFunctions::SetName (C++ function), 113
StatsFunctions::GetMagickaCurrent (C++ function), 111	StatsFunctions::SetRace $(C++function)$, 114 StatsFunctions::SetResetStats $(C++function)$,
StatsFunctions::GetModel (C++ function), 110	115
StatsFunctions::GetName (C++ function), 110 StatsFunctions::GetRace (C++ function), 110 StatsFunctions::GetSkillBase (C++ function), 112	StatsFunctions::SetSkillBase(C++ function), 117 StatsFunctions::SetSkillDamage(C++ function), 118
StatsFunctions::GetSkillCount (C++ function), 109	StatsFunctions::SetSkillIncrease (C++ function), 118
StatsFunctions::GetSkillDamage $(C++ function)$, 113	StatsFunctions::SetSkillProgress (C++ function), 118
$\label{eq:StatsFunctions::GetSkillId} StatsFunctions::GetSkillIncrease (C++ functions).$	W
tion), 113	WorldstateFunctions (C++ class), 120
StatsFunctions::GetSkillModifier (C++ function), 113	WorldstateFunctions::AddCellToReset (C++ function), 126
$\label{eq:StatsFunctions::GetSkillName} StatsFunctions::GetSkillProgress \ (C++ \ func-$	WorldstateFunctions::AddClientGlobalFloat (C++function), 125
tion), 113	WorldstateFunctions::AddClientGlobalInteger
StatsFunctions::SendAttributes (C++ function),	(C++function), 125
119 StatsFunctions::SendBaseInfo(C++function), 119	WorldstateFunctions::AddDestinationOverride
StatsFunctions::SendBounty (C++ function), 120	(C++ function), 126
StatsFunctions::SendLevel (C++ function), 120	<pre>WorldstateFunctions::AddEnforcedCollisionRefId</pre>
StatsFunctions::SendSkills (C++ function), 119	WorldstateFunctions::AddKill(C++function), 125
StatsFunctions::SendStatsDynamic (C++ func- tion), 119	WorldstateFunctions::AddSynchronizedClientGlobalId (C++ function), 126
StatsFunctions::SetAttributeBase (C++ func- tion), 116	WorldstateFunctions::AddSynchronizedClientScriptId (C++ function), 126
StatsFunctions::SetAttributeDamage (C++ func-	ζ-·· y ······· <i>γ</i> // -

```
WorldstateFunctions::ClearCellsToReset (C++ WorldstateFunctions::SaveMapTileImageFile
                                                         (C++function), 127
       function), 127
WorldstateFunctions::ClearClientGlobals
                                                WorldstateFunctions::SendCellReset (C++ func-
        (C++ function), 120
                                                         tion), 129
WorldstateFunctions::ClearDestinationOverridesWorldstateFunctions::SendClientScriptGlobal
        (C++ function), 127
                                                         (C++ function), 127
WorldstateFunctions::ClearEnforcedCollisionRefWorldstateFunctions::SendClientScriptSettings
        (C++ function), 126
                                                         (C++ function), 127
WorldstateFunctions::ClearKillChanges (C++ WorldstateFunctions::SendWorldCollisionOverride
       function), 120
                                                         (C++ function), 129
WorldstateFunctions::ClearMapChanges
                                         (C++ WorldstateFunctions::SendWorldDestinationOverride
       function), 120
                                                         (C++ function), 129
WorldstateFunctions::ClearSynchronizedClientGlwdmalldsstsateFunctions::SendWorldKillCount
        (C++ function), 126
                                                         (C++ function), 128
WorldstateFunctions::ClearSynchronizedClientSchrödistateFunctions::SendWorldMap (C++ func-
        (C++ function), 126
                                                         tion), 128
WorldstateFunctions::CopyReceivedWorldstateToSMcomrledstateFunctions::SendWorldRegionAuthority
        (C++ function), 120
                                                         (C++ function), 128
WorldstateFunctions::GetClientGlobalFloatValueWorldstateFunctions::SendWorldTime\ (C++\ functions)
        (C++ function), 122
                                                         tion), 128
{\tt WorldstateFunctions::GetClientGlobalId} (C++ {\tt WorldstateFunctions::SendWorldWeather} (C++
       function), 122
                                                        function), 129
WorldstateFunctions::GetClientGlobalIntValue WorldstateFunctions::SetActorCollisionState
        (C++function), 122
                                                         (C++ function), 125
WorldstateFunctions::GetClientGlobalsSize
                                                WorldstateFunctions::SetAuthorityRegion
        (C++ function), 121
                                                         (C++ function), 122
WorldstateFunctions::GetClientGlobalVariableTyWoerldstateFunctions::SetDay (C++function), 124
        (C++ function), 122
                                                WorldstateFunctions::SetDaysPassed (C++ func-
WorldstateFunctions::GetKillChangesSize
                                                         tion), 124
                                                WorldstateFunctions::SetHour(C++function), 124
        (C++ function), 120
WorldstateFunctions::GetKillNumber (C++ func-
                                                WorldstateFunctions::SetMonth (C++ function),
        tion), 121
                                                         124
                                                WorldstateFunctions::SetPlacedObjectCollisionState
WorldstateFunctions::GetKillRefId (C++ func-
        tion), 121
                                                         (C++ function), 125
WorldstateFunctions::GetMapChangesSize (C++ WorldstateFunctions::SetPlayerCollisionState
       function), 121
                                                         (C++ function), 124
WorldstateFunctions::GetMapTileCellX
                                         (C++
                                                WorldstateFunctions::SetTimeScale (C++ func-
       function), 122
                                                         tion), 124
WorldstateFunctions::GetMapTileCellY
                                          (C++ WorldstateFunctions::SetWeatherCurrent (C++
       function), 122
                                                        function), 123
WorldstateFunctions::GetWeatherCurrent (C++ WorldstateFunctions::SetWeatherForceState
       function), 121
                                                         (C++ function), 123
WorldstateFunctions::GetWeatherNext
                                          (C++
                                                WorldstateFunctions::SetWeatherNext
                                                                                          (C++
       function), 121
                                                        function), 123
WorldstateFunctions::GetWeatherQueued
                                         (C++
                                                WorldstateFunctions::SetWeatherQueued
                                                                                          (C++
       function), 121
                                                        function), 123
WorldstateFunctions::GetWeatherRegion (C++ WorldstateFunctions::SetWeatherRegion (C++
       function), 121
                                                        function), 123
WorldstateFunctions::GetWeatherTransitionFactdWorldstateFunctions::SetWeatherTransitionFactor
        (C++ function), 121
                                                         (C++ function), 123
WorldstateFunctions::LoadMapTileImageFile
                                                WorldstateFunctions::SetYear(C++function), 124
        (C++ function), 127
                                                WorldstateFunctions::UseActorCollisionForPlacedObjects
WorldstateFunctions::ReadReceivedWorldstate
                                                         (C++ function), 125
        (C++ function), 120
```