Table of Contents

Introduction

This project is known as AionScript, which has been created in order to get an easy-to-use interface with Aion, allowing users to create scripts to automate game play or assist in playing. This application requires *Microsoft .NET 4.0* and *Visual C++ 2010 Redistributable X86*. Click here to view the latest patch notes!

Game Classes Reference

These classes include everything that is required to interact with game. Some of these depend on additional classes, which can be found in the Miscellaneous Classes Reference. There are some additional notes about the game classes and the way they have been listed in the following sections. All the methods are listed in either black or red (where you should avoid using those in red, if you can), but public variables are shown in green! Methods in blue are deprecated and cannot be used anymore. Parameters shown in purple are optional and are not required.

Class Overview

Ability	Contains information about a character ability, including its reuse and cooldown.
AbilityList	Contains the available abilities for your character.
Dialog	Reads the dialog, update the related information and Retrieve the child dialogs.
DialogList	Reads the available initial dialogs and children of each of them.
Entity	Contains information about an entity, such as its state, health, name and hostility.
EntityList	Contains the available entities that have been loaded in the surrounding area.
Force	Contains details about a force member, including its health and name.
ForceList	Contains the available members that are in a force with your character.
Inventory	Contains information about an inventory item, including its reuse and cooldown.
InventoryList	Reads the available inventory items for the currently logged in character.
Player	Contains information about the player, including entity values.
Skill	Contains information about a skill, including its name and attributes.
SkillList	Contains the skills that have been loaded from the skills file.
State	Contains information about a skill state, which is a skill as in your current state.
StateList	Contains the state of an entity, which means all of his buffs and debuffs.
Travel	Contains information about a travel name, including its name.
TravelList	Contains the list of travel nodes and provides methods to interact with them.

Duplicate Methods

They aren't duplicates, but have something different in the provided parameters. For example, you could access the method *GetEntity()* in the EntityList class using either a name or identifier. Both will work, but will have to be handled differently in the code. On the end-user perspective, it makes no difference which one is used, but there will be distinct advantages and disadvantages (speed, accuracy, etc).

Enumerators

Some methods will return enumerators, these can be used directly in an extension but you cannot handle them in Lua. If you want to use enumerators in Lua, convert the resulting enumerator to a string using the *ToString()* method and handle the resulting string accordingly.

Extended Classes

Extended classes add additional functionality to the base class. The prime example would based on an Entity, which has been extended as Player. This means that all the methods listed in the Entity class are all available when dealing with Player. It improves a few methods and adds new ones that can only be applied on your own character.

Protected Methods

Some methods have been marked in red and are therefore protected. You can access them just like any regular method but they can cause a lot of problems. They might modify the character in such a way that it will be detected as a cheat and get your character banned. Some other functions aren't needed, except when you're adding something new that AionInterface did not expose yet and you know **exactly** what you are doing!

Global Functions

Close	@desc	Closes the scripting engine and updates the visual feedback.
	@return	Void
Include	@desc	Includes a script by loading it into memory. Returns an object.
	@param [string]	Contains the name of the file to include.
	@return	object
Time	@desc	Return the time in milliseconds, based on the UNIX timestamp.
	@return	uint
Register	@desc	Registers a hotkey on the provided function.
	@param [string]	Contains the name of the function to execute.
	@param [string]	Contains the key to bind (click for a list).
	@param [string]	Contains the optional modifier (Alt/Control/Shift/Super).
	@return	bool
Travel	@desc	Loads the provided file as the travel node.
	@param [string]	Contains the file to load.
	@return	bool
Unregister	@desc	Unregisters the hotkey from the provided key
	@param [string]	Contains the key to unbind (click for a list).
	@param [string]	Contains the optional modifier (Alt/Control/Shift/Super).
	@return	bool
Write	@desc	Writes a message to the console interface.
	@param [string]	Contains the message to write to the console.
	@return	void

Ability

GetActivated	@desc	Retrieve the status whether or not the ability is active.
	@return	bool
GetAddress	@desc	Retrieve the ability address, this is extended information.
	@return	uint
GetCooldown	@desc	Retrieve the remaining cooldown for this ability.
	@return	uint
GetID	@desc	Retrieve the ability identifier.
	@return	uint
GetName	@desc	Retrieve the ability name.
	@return	uint
GetReuse	@desc	Retrieve the ability reuse time.
	@return	uint

AbilityList

GetActivated	@desc	Retrieve the status whether or not the ability is active.
	@return	bool
GetAbility	@desc	Retrieve the ability with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	Ability
GetAbility	@desc	Retrieve the ability with the provided name.
	@param [string]	Contains the name to look for (without roman allowed).
	@return	Ability
GetAbilityIndex	@desc	Retrieve the ability at the provided index.
	@param [uint]	Contains the index to get.
	@return	Ability
GetAbilitySize	@desc	Retrieve the total amount of abilities.
	@return	uint
GetList	@desc	Retrieve the list of the abilities based on the identifier.
	@return	Dictionary<uint, Ability>

Dialog : DialogList

Click	@desc	Simulates a click on the center of the dialog/control.
	@return	bool
GetAddress	@desc	Retrieve the ability address, this is extended information.
	@return	uint
GetHTML	@desc	Retrieve the HTML text identifier (HTML-controls only).
	@return	uint
GetIndex	@desc	Retrieve the index this dialog has in the parent dialog.
	@return	uint
GetName	@desc	Retrieve the dialog/control name.
	@return	string
GetPadding	@desc	Retrieve the padding of the dialog/control.
	@return	Vector3D
GetParent	@desc	Retrieve the parent dialog/control.
	@return	Dialog
GetPosition	@desc	Retrieve the absolute position of the dialog/control.
	@return	Vector3D
GetSize	@desc	Retrieve the size of the dialog/control.
	@return	Vector3D
GetText	@desc	Retrieve the text of the dialog/control.
	@return	string
IsEnabled	@desc	Indicates whether or not this dialog/control is enabled.
	@return	bool
IsVisible	@desc	Indicates whether or not this dialog/control is visible.
	@return	bool
SetEnabled	@desc	Sets the enabled status of this dialog/control.
	@param [bool]	Indicates whether or not to enable this dialog/control.
	@return	void
SetVisible	@desc	Sets the visibility status of this dialog/control.
	@param [bool]	Indicates whether or not to show this dialog/control.
	@return	void

DialogList

GetDialog	@desc	Retrieve the dialog with the provided name.
	@param [string]	Contains the name to look for.
	@return	Dialog
GetDialogIndex	@desc	Retrieve the dialog at the provided index.
	@param [int]	Contains the index to get.
	@return	Dialog
GetDialogSize	@desc	Retrieve the total amount of dialogs.
	@return	uint
GetList	@desc	Retrieve the list of dialogs.
	@return	List<Dialog>

Entity

GetAddress	@desc	Retrieve the entity address, this is extended information.
	@param [int]	Indicates which node (0 = Entity, 1 = Node, 2 = Extended).
	@return	uint
GetAttitude	@desc	Retrieve the entity attitude, such as passive or friendly.
	@return	eAttitude
GetAttackSpeed	@desc	Retrieve the attack speed the entity has.
	@return	uint
GetClass	@desc	Retrieve the entity class.
	@return	eClass
GetDP	@desc	Retrieve the amount divine power this entities has.
	@return	uint
GetHealth	@desc	Retrieve the health in percentages.
	@return	Byte
GetHealthCurrent	@desc	Retrieve the remaining amount of health (NPC and self).
	@return	uint
GetHealthMaximum	@desc	Retrieve the maximum amount of health (NPC and self).
	@return	uint
GetID	@desc	Retrieve the entity identifier.
	@return	uint
GetLegion	@desc	Retrieve the legion name of the entity.
	@return	string
GetLevel	@desc	Retrieve the level of the entity (NPC and players).
	@return	byte
GetPosition	@desc	Retrieve the position of the entity.
	@return	Vector3D
GetName	@desc	Retrieve the name of the entity.
	@return	string
GetOwnerID	@desc	Retrieve the identifier of the owner.
	@return	uint
GetOwnerName	@desc	Retrieve the name of the owner.
	@return	string
GetRank	@desc	Retrieve the abyss rank (0-17 where 0 is Soldier, Rank 9)
	@return	uint
GetRotation	@desc	Retrieve the rotation of the entity.
	@return	float
GetSkillID	@desc	Retrieve the skill identifier that is currently being executed.
	@return	uint
GetSkillTime	@desc	Retrieve the remaining time of currently executing skill.
	@return	uint
GetSpeed	@desc	Retrieve the moving speed of the entity.
	@return	float
GetStance	@desc	Retrieve the entity stance (such as flying and combat).
	@return	eStance
GetState	@desc	Retrieve the entity state.
	@return	EntityState
GetTargetID	@desc	Retrieve the identifier of the selected target.
	@return	uint
GetTypeID	@desc	Retrieve the type identifier (objects, monsters and gatherables).
	@return	uint
IsBusy	@desc	Returns whether or not this entity is currently busy.
	@return	bool
IsDead	@desc	Returns whether or not the entity is dead.
	@return	bool
IsGatherable	@desc	Returns whether or not the entity is gatherable.
	@return	bool
IsGliding	@desc	Returns whether or not the entity is currently gliding.
	@return	bool
IsKisk	@desc	Returns whether or not this entity is a kisk.
	@return	bool
IsFlying	@desc	Returns whether or not the entity is currently flying.
	@return	bool

IsFriendly	@desc	Returns whether or not the entity is friendly.
	@return	bool
IsHidden	@desc	Returns whether or not the entity is hidden.
	@return	bool
IsHostile	@desc	Returns whether or not the entity is hostile.
	@return	bool
IsMonster	@desc	Returns whether or not this entity is a monster or NPC.
	@return	bool
IsObject	@desc	Returns whether or not this entity is an object.
	@return	bool
IsPet	@desc	Returns whether or not this entity is a pet.
	@return	bool
IsPlayer	@desc	Returns whether or not this entity is a player.
	@return	bool
IsResting	@desc	Returns whether or not this entity is currently resting.
	@return	bool
SetAttackSpeed	@desc	Set the attack speed entity has.
	@param [uint]	Contains the attack delay in milliseconds.
	@return	void
SetLegion	@desc	Set the legion name of the entity.
	@param [string]	Contains the name to set.
	@return	Void
SetName	@desc	Set the name of the entity.
	@param [string]	Contains the name to set.
	@return	void
SetPosition	@desc	Set the location for the entity.
	@param [Vector3D]	Contains the new position for the entity.
	@return	void
SetPosition	@desc	Set the location for the entity.
	@param [float]	Contains the X-axis of the position to set.
	@param [float]	Contains the Y-axis of the position to set.
	@param [float]	Contains the Z-axis of the position to set.
	@return	void
SetSpeed	@desc	Set the moving speed of the entity.
	@param [float]	Contains the new speed to set.
	@return	void

EntityList

GetEntity	@desc	Retrieve the entity with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	Entity
GetEntity	@desc	Retrieve the entity with the provided name.
	@param [string]	Contains the name to look for.
	@return	Entity
GetEntityIndex	@desc	Retrieve the entity at the provided index.
	@param [uint]	Contains the index to get.
	@return	Entity
GetEntitySize	@desc	Retrieve the total amount of entities.
	@return	uint
GetList	@desc	Retrieve the list of the entities.
	@return	Dictionary<uint, Entity>

Force

GetAddress	@desc	Retrieve the force address, this is extended information.
	@return	uint
GetClass	@desc	Retrieve the class of the force member.
	@return	eClass
GetID	@desc	Retrieve the entity identifier.
	@return	uint
GetHealth	@desc	Retrieve the health in percentages.
	@return	byte
GetHealthMaximum	@desc	Retrieve the maximum amount of health.
	@return	uint
GetHealthCurrent	@desc	Retrieve the remaining amount of health.
	@return	uint
GetMana	@desc	Retrieve the mana in percentages.
	@return	byte
GetManaMaximum	@desc	Retrieve the maximum amount of mana.
	@return	uint
GetManaCurrent	@desc	Retrieve the remaining amount of mana.
	@return	uint
GetFlightTime	@desc	Retrieve the flight time in percentages.
	@return	byte
GetFlightTimeMaximum	@desc	Retrieve the maximum amount of flight time (in milliseconds).
	@return	uint
GetFlightTimeCurrent	@desc	Retrieve the remaining amount of flight time (in milliseconds).
	@return	uint
GetLevel	@desc	Retrieve the level of the force member.
	@return	byte
GetName	@desc	Retrieve the name of the force member.
	@return	string
GetPosition	@desc	Retrieve the position of the force member.
	@return	Vector3D
GetTeam	@desc	Retrieve the team identifier of this force member.
	@return	uint
GetWorld	@desc	Retrieve the world identifier the player is currently in.
	@return	uint
IsLeader	@desc	Indicates whether or not this force member is the force leader.
	@return	bool

ForceList

GetForce	@desc	Retrieve the force member with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	Force
GetForce	@desc	Retrieve the force member with the provided name.
	@param [string]	Contains the name to look for.
	@return	Force
GetForceLeader	@desc	Retrieve the force leader.
	@return	bool
GetForceIndex	@desc	Retrieve the force member at the provided index.
	@param [uint]	Contains the index to get.
	@return	Force
GetForceSize	@desc	Retrieve the total amount of force members.
	@return	uint
GetList	@desc	Retrieve the list of the entities based on the identifier as key.
	@return	Dictionary<uint, Force>

Inventory

GetAddress	@desc	Retrieve the inventory address, this is extended information.
	@return	uint
GetAmount	@desc	Retrieve the amount of items that are on this item its stack.
	@return	uint
GetCooldown	@desc	Retrieve the remaining cooldown for this inventory item.
	@return	uint
GetID	@desc	Retrieve the inventory item identifier.
	@return	uint
GetName	@desc	Retrieve the inventory item name.
	@return	String
GetReuse	@desc	Retrieve the inventory item reuse time.
	@return	uint
GetType	@desc	Returns the inventory item type.
	@return	eInventoryType
GetSlot	@desc	Retrieve the slot identifier in which this item is equipped.
	@return	eInventorySlotType

InventoryList

GetInventory	@desc	Retrieve the inventory item with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	Inventory
GetInventory	@desc	Retrieve the inventory item with the provided name.
	@param [string]	Contains the name to look for.
	@return	Inventory
GetInventoryIndex	@desc	Retrieve the inventory item at the provided index.
	@param [uint]	Contains the index to get.
	@return	Inventory
GetInventorySize	@desc	Retrieve the total amount of inventory items.
	@return	uint
GetList	@desc	Retrieve the list of the inventory items based on the identifier.
	@return	List<Inventory>
GetSlotCurrent	@desc	Retrieve the number of items that are currently in your cube.
	@return	uint
GetSlotMaximum	@desc	Retrieve the maximum of items that can be stored.
	@return	uint

Player : Entity

GetAttackRange	@desc	Retrieve the weapon range for the currently equipped weapon.
	@return	float
GetBrand	@desc	Retrieve the entity identifier of the provided brand number.
	@param [uint]	Contains the brand number to check.
	@return	uint
GetCamera	@desc	Retrieve the camera yaw and pitch for the player.
	@return	Vector3D
GetExperienceCurrent	@desc	Retrieve the current experience of the player.
	@return	uint
GetExperienceRecoverable	@desc	Retrieve the recoverable experience of the player.
	@return	uint
GetExperienceRequired	@desc	Retrieve the required experience to level.
	@return	uint
GetFlightCooldown	@desc	Retrieve the remaining time before flight becomes available.
	@return	uint
GetMarked	@desc	Retrieve the name of the entity that is marked on the radar.
	@return	string
GetPositionMove	@desc	Retrieve the position that is being used for click-to-move.
	@return	Vector3D
GetWorld	@desc	Retrieve the world identifier the player is currently in.
	@return	uint
IsMoving	@desc	Returns whether or not this entity is moving to a position.
	@return	bool
SetAction	@desc	Set a action for the player entity.
	@param [eAction]	Contains the action enumerator to set.
	@return	bool
SetAction	@desc	Set a action for the player entity.
	@param [string]	Contains the action name to set.
	@return	bool
SetAttackRange	@desc	Set the weapon range for the currently equiped weapon.
	@return	void
SetCamera	@desc	Set the camera angles to face the provided position.
	@param [Vector3D]	Contains the position to face.
	@param [bool]	Indicates whether or not to move both X- and Y-axis.
	@return	void
SetMove	@desc	Set a move action for the player entity.
	@param [Vector3D]	Contains the position to move to.
	@param [int]	Indicates whether or not air checks are to be enforced.
	@return	bool
SetMove	@desc	Set a move action for the player entity.
	@param [float]	Contains the X-axis of the position to move to.
	@param [float]	Contains the Y-axis of the position to move to.
	@param [float]	Contains the Z-axis of the position to move to.
	@param [int]	Indicates whether or not air checks are to be enforced.
	@return	bool
SetTarget	@desc	Set the target for the player entity.
	@param [Entity]	Contains the target to select.
	@return	void

PlayerInput

Ability	@desc	Execute the ability with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	bool
Ability	@desc	Execute the ability with the provided name.
	@param [string]	Contains the name to look for (without roman allowed).
	@return	bool
Click	@desc	Sends a click to the game window.
	@param [uint]	Contains the X-offset.
	@param [uint]	Contains the Y-offset.
	@return	bool
Console	@desc	Send a message to the console and executes it.
	@param [string]	Message which is to be send.
	@return	bool
Inventory	@desc	Execute the inventory item with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	bool
Inventory	@desc	Execute the inventory item with the provided name.
	@param [string]	Contains the name to look for.
	@return	bool
Register	@desc	Registers a function on the provided hotkey/modifier (not Lua).
	@param [mixed]	Contains the function to execute.
	@param [mixed]	Contains the key on which to register.
	@param [mixed]	Contains the modifier on which to register.
	@return	bool
Unregister	@desc	Remove the function on the provided hotkey/modifier (not Lua).
	@param [mixed]	Contains the key on which to unregister.
	@param [mixed]	Contains the modifier on which to unregister.
	@return	void

Skill

GetDispell	@desc	Retrieve the dispel required for the skill.
	@return	eDispell
GetID	@desc	Retrieve the skill identifier.
	@return	uint
GetName	@desc	Retrieve the name of the skill.
	@return	string
GetType	@desc	Retrieve the skill type (Magical or Physical).
	@return	eSkillType
GetTypeSecondary	@desc	Retrieve the secondary type (Passive, Attack, Buff, etc).
	@return	eSkillTypeSecondary
IsAttack	@desc	Indicates whether or not this skill is an attack.
	@return	bool
IsBuff	@desc	Indicates whether or not this skill is a buff.
	@return	bool
IsDebuff	@desc	Indicates whether or not this skill is a debuff.
	@return	bool
IsHeal	@desc	Indicates whether or not this skill is a heal (not healing attacks).
	@return	bool
IsMagical	@desc	Indicates whether or not this skill is magical.
	@return	bool
IsPassive	@desc	Indicates whether or not this skill is passive.
	@return	bool
IsPhysical	@desc	Indicates whether or not this skill is physical.
	@return	bool
IsStun	@desc	Indicates whether or not this skill puts you in a stun state.
	@return	bool
IsValid	@desc	Returns whether or not this skill is valid (updated properly).
	@return	bool

SkillList

GetSkill	@desc	Retrieve the skill with the provided identifier.
	@param [uint]	Contains the identifier to look for.
	@return	Skill
GetSkill	@desc	Retrieve the skill with the provided name.
	@param [string]	Contains the name to look for.
	@return	Skill
GetSkillIndex	@desc	Retrieve the skill at the provided index (not recommended).
	@param [uint]	Contains the index to get.
	@return	Skill
GetSkillSize	@desc	Retrieve the total amount of abilities.
	@return	uint
GetList	@desc	Retrieve the list of the skills.
	@return	Dictionary<uint, Skill>

State : Skill

Temporary place holder; no additional methods have been implemented for a State at this time.

StateList

GetList	@desc	Retrieve the list of the state items based on the identifier.
	@return	Dictionary<uint, State>
GetState	@desc	Retrieve the skill with the provided identifier (if available).
	@param [uint]	Contains the identifier to look for.
	@return	State
GetState	@desc	Retrieve the skill with the provided name (if available).
	@param [string]	Contains the name to look for.
	@return	State
GetStateIndex	@desc	Retrieve the skill state at the provided index.
	@param [uint]	Contains the index to get.
	@return	State
GetStateSize	@desc	Retrieve the total amount of states on this entity.
	@return	uint

Travel

GetName	@desc	Retrieve the name of the node.
	@return	String
GetPosition	@desc	Retrieve the position of the node.
	@return	Vector3D
IsAction	@desc	Retrieve whether or not this node is an action node.
	@return	bool
IsFlying	@desc	Retrieve whether or not this is an air node.
	@return	bool
IsMove	@desc	Retrieve whether or not this is an movement node.
	@return	bool
IsRest	@desc	Retrieve whether or not this node is a rest node.
	@return	bool

TravelList

GetCurrent	@desc	Retrieve the current travel node.
	@return	Travel
GetList	@desc	Retrieve the list of nodes that have been loaded.
	@return	List<Travel>
GetNext	@desc	Retrieve the next travel node, this will advance the node.
	@return	Travel
GetPrevious	@desc	Retrieve the previous travel node.
	@return	Travel
IsReverse	@desc	Retrieve the type of node, true if reverse, otherwise circular.
	@return	bool
Modify	@desc	Modifies the current travel path with the provided parameters.
	@param [Travel]	Contains the travel object to add/insert.
	@param [int]	Contains the position to use.
	@return	bool
Move	@desc	Move to the provided travel node and handle flying/walk nodes.
	@return	bool

Game Enumerator Reference

These enumerators are used by the various game classes and describe additional information that is required to function correctly. Lua scripts cannot handle enumerators directly, please look here to get around that. Actions that have been marked in red cannot be used to set an action, only to find out what your character is doing!

Action (Player)

None	No action is being taken.
Attack	Attacking a target, using auto-attacks and auto-approach when available.
Cast	Casting a spell on a target, the spell must have a cast time to show up as this.
Gather	Gathering a node, which includes Aether Tapping.
Follow	Following a target around, duplicating the exact path the target is walking.
MoveForward	Moving forward in a passive stance (weapon not held).
MoveBackward	Moving backward in a passive stance (weapon not held).
MoveForwardCombat	Moving forward in a combat stance (weapon held).
MoveBackwardCombat	Moving backward in a combat stance (weapon held).
ToggleCombat	Toggle combat, which either makes the character go into passive or combat mode.
RestSit	Sitting down to rest, note that this only applies on the sitting down animation!
RestStand	Standing up after resting, ending the resting state.
Emote	Doing some kind of emote (but we cannot determine which).
FlightTakeOff	Taking off to go into flight mode.
FlightLand	Landing after flying around in flight mode.
Loot	Looting a target, using auto-approach when available.
FaceTarget	Face a target, making your character look in its direction.
Talking	Talking to a NPC. This causes the target to become corrupted.

Attitude (Entity)

Passive	Does not attack on sight, but will not assist in battle.
Hostile	Does attack on sight or is a player of the opposite faction.
Friendly	Does not attack and will help out when you are attacked.
Utility	Does not do anything, this is an utility entity.

Dispell (Skill)

None	Cannot be dispelled.
Mental	This requires a Mental dispel to remove the effect (Cure Mind).
Physical	This requires a Physical dispel to remove the effect (Dispel).
Stun	This requires a stun-removing ability to remove the effect (Remove Shock).

InventoryType (Inventory)

Material	This is a kind of material.
Weapon1H	This is a one-handed weapon.
Weapon2H	This is a two-handed weapon.
Bow	This is a bow.
Arrow	These are arrows.
Shield	This is a shield.
Armor	This is a piece of armor.
Accessory	This is an acessory.
Stigma	This is a stigma stone.
StigmaShard	This is a stigma shard.
Quest	This is a quest item.
Food	This is a kind of food.
Resurrect	This is a kind of resurrection item.
Potion	This is a kind of potion.
Scroll	This is a kind of scroll.
Currency	This is a kind of currency.
Tools	This is a kind of tools.
Enchant	This is an enchant stone.
Manastone	This is a mana stone.
Godstone	This is a godstone.
Shard	This is a shard.
Design	This is a design.
DyeRemover	This is a dye remover.
Dye	This is a dye.
Manual	This is a manual.
Key	This is a key.
Wings	This is a pair of wings.
Special	This is a special item.
Craft	This is a craft-related material.

InventorySlotType (Inventory)

None	This should be rather obvious.
MainHand_Equipped	This should be rather obvious.
OffHand_Equipped	This should be rather obvious.
Head	This should be rather obvious.
Jacket	This should be rather obvious.
Gloves	This should be rather obvious.
Shoes	This should be rather obvious.
Ear_R	This should be rather obvious.
Ear_L	This should be rather obvious.
Ring_R	This should be rather obvious.
Ring_L	This should be rather obvious.
Necklace	This should be rather obvious.
Pauldrons	This should be rather obvious.
Pants	This should be rather obvious.
Shard_R	This should be rather obvious.
Shard_L	This should be rather obvious.
Wing	This should be rather obvious.
Belt	This should be rather obvious.
MainHand_NotEquipped	This should be rather obvious.
OffHand_NotEquipped	This should be rather obvious.
Stigma#	Equipped in a stigma slot (1 to 6).
AdvancedStigma#	Equipped in an advanced stigma slot (1 to 5).

Stance (Entity)

`Normal`	Passive stance, not doing anything.
`Combat`	Combat stance, weapon is being held.
`Resting`	Resting stance, is currently sitting down.
`Flying`	Flying stance, has wings out but isn’t in combat stance.
`FlyingCombat`	Flying stance, which includes having a weapon held.
`Dead`	Dead.

SkillType (Skill)

`Magical`	This is a magical type skill.
`Physical`	This is a physical type skill.

SkillTypeSecondary (Skill)

`Passive`	Passive skill which does not need activation.
`Attack`	Active attack-type skill which inflicts damage (draining blows included).
`Buff`	Supportive effect, such as Blessing of Health and Blessing of Rock.
`Debuff`	Negative effect, such as Erosion and Chain of Earth.
`Heal`	Healing type skill, does not inflict damage but only heals (draining blows excluded).

Miscellaneous Classes Reference

These classes are designed to be easy-to-use and can be applied in many projects. This includes the Vector3D class, which has functions to calculate distances, 2D positions and camera angles as well as containing the X, Y and Z coordinates and the memory class that can be used to open and read any applications memory space.

Memory

Allocate	@desc	Allocate memory in the open process.
	@param [uint]	Size of memory to allocate.
	@return	uint
Base	@desc	Find the base address of the provided module.
	@param [string]	Contains the module name to retrieve.
	@return	uint
Close	@desc	Close the process handle.
	@return	bool
Import	@desc	Find the address of a function contained in the import table.
	@param [uint]	Contains the module base address.
	@param [string]	Contains the name of the module.
	@param [string]	Contains the name of the function.
	@return	uint
Open	@desc	Open a process with the desired access level.
	@param [string]	Contains the process name.
	@param [enum]	Contains the desired access level.
	@return	uint
Read	@desc	Read the value at the provided memory location with the size.
	@param [uint]	Address in target process to read.
	@param [uint]	Number of bytes to read.
	@return	byte[]
Window	@desc	Retrieve the window handle for the opened process.
	@return	IntPtr
Write	@desc	Write the provided value at the memory location with the size.
	@param [uint]	Address in target process to write.
	@param [byte[]]	Value which is to be written.
	@return	uint

Vector3D

X	@var	Contains the position on the X-axis.
Y	@var	Contains the position on the Y-axis.
Z	@var	Contains the position on the Z-axis.
Yaw	@var	Synonym for the X-axis using Yaw (camera usage).
Pitch	@var	Synonym for the Y-axis using Pitch (camera usage).
Roll	@var	Synonym for the Z-axis using Roll (camera usage).
CalculateCamera	@desc	Calculate the camera angles to the target vector.
	@param [uint]	Contains the position of the target.
	@return	Vector3D
CalculatePosition2D	@desc	Calculate the position of this vector compared to the camera.
	@param [Vector3D]	Contains the position of the player.
	@param [Vector3D]	Contains the camera of the player.
	@return	Vector3D
DistanceToPosition	@param	Calculate the distance to the provided position.
	@param [Vector3D]	Contains the position to calculate distance to.
	@param [double]	Contains the clamping distance (if any).
	@return	double

Patch Notes (More @ Website)

28 February, 2011

1. Added native support for Infinite Aion (required special library before).
2. Fixed/improved GetIndex-methods for AbilityList, DialogList and StateList.
3. Removed Demo mode from the application; You need an active account now!

22 February, 2011

1. Added array-access to list classes (Entity = EntityList["Blastradius"])
2. Added GetTypeID to Entity, which is used to determine the type of the entity.
3. Changed internal entity routines to check for invalid memory and identifiers.
4. Changed internal routines to be faster and more efficient (main loop, lists, entities).
5. Changed GetList methods to return identifier-based lists, rather than incremental ones.
6. Changed methods in StateList; now returning State instead of Skill (same methods available).
7. Changed GetSlot in Inventory to return an enumerator to easily identify which slot it is in.
8. Fixed the movement system to prevent it from interrupting actions.
9. Fixed an accidental release of an unfinished update for the EntityList (Sorry!)
10. Fixed an issue with the inventory slot offset. For some reason, it was changed.
11. Removed IsValid from Entity, new internal routines do not require this function.
12. Renamed GetDialogIndex to GetDialog. GetDialog requires identifiers (used thus far).
13. Renamed GetDialogByIndex to GetDialog (in Dialog class).

20 January, 2011

1. Revamped Party (now Force) and PartyList (now ForceList). Supports both alliances and parties.
2. Revamped Player and PlayerEntity (decrepated). All methods of PlayerEntity are now in Player.
3. Revamped Travel and TravelList, introduced node types and parameters (Action-node system).
4. Added HTML number to the dialog inspector and added clear buttons to console/script/travel box.
5. Added IsAsmodian, IsElyos, IsGliding, GetRank and GetSkillTime to Entity. Fixed related functions.
6. Added IsLeader, GetEntity, GetTeam and GetWorld to Force. GetTeam is useful for alliance teams.
7. Added GetFlightCooldown to Player, which shows the remaining flight cooldown.
8. Added index- and size methods to AbilityList, EntityList, SkillList and TravelList.
9. Added global method Unregister. For some reason, it was not available, but was documentated.
10. Fixed an issue with the camera offset which prevented pitch movement (2.1.0.X).
11. Fixed an issue which caused the entities extension to count kisks as enemies/friendlies.
12. Fixed an issue with the action system, this includes a rather huge fix for movement.
13. Fixed an issue with internal timers causing scripts to run too fast (Thanks Bytecompile).
14. Improved performance of SkillList. Requested skills are now cached to speed up skill lookups.
15. Renamed the method GetDialogByIndex to GetDialogIndex in Dialog.
16. Renamed the EntityState to StateList. This makes more sense once State is implemented.
17. Updated SetMove in PlayerEntity. It now allows enforcing of flying and walking.
18. Updated radar images. Added unique images for up/down/death to the radar (Thanks Evasionfruit).

04 January, 2011

1. Added the initial implementation of the Dialog system. Examples are now available.
2. Added GetTargetID method in Player which is more reliable. PlayerEntity now uses it internally.
3. Added the initial performance improvement systems designed to ease expensive system calls.
4. Added a file name to error messages caused by an included file (Lua).
5. Improved the Click method in PlayerInput to be much more reliable.
6. Improved the SetMove method in PlayerEntity to be much more reliable.
7. Fixed an issue with window detection for hotkeys (Yes, I fixed it, *again*).
8. Fixed an issue with the console input method getting stuck.

15 December, 2010

1. Added a "Trial Mode" for AionScript, allowing free use up to level 30 (Experimental feature).
2. Improved the Console method in PlayerInput to take care of an issue that could leave chat open.
3. Improved the walking method to be more accurate, about double as much.
4. Fixed an issue with binding keys; they are never lost again between character switches.
5. Fixed an issue with the main window detection which was needed for key bindings.
6. Fixed an issue in the Lua scripting engine. Switching characters keeps a script functional now.
7. Fixed the offset for the move position used by click-to-move. It was off by four bytes.
8. Fixed an issue with the Documentation saying the 'Register' method could not be used in Lua.
9. Fixed an issue with selecting Entity with the identifier zero.
10. Fixed an issue with entity states being cleared when they really shouldn't be.

02 December, 2010

1. Added initial implementation of Dialog and DialogList. Undocumented and subject to change!
2. Added active skills; allows detection of available skills, such as stun/parry/chain skills.
3. Added Friendly/Hostile counter in the Entities extension.
4. Added object display on the radar, for any object.
5. Moved SetPosition from PlayerEntity to Entity, so it can used on any entity.
6. Fixed a crashing issue when scripts tried to write an empty string to the console.
7. Fixed an issue with PlayerEntity.SetMove(null) which did not clear the movement state.
8. Fixed an issue with the AbilityList not updating properly when switching characters.
9. Fixed a potential issue with PlayerEntity.SetMove() at your own position and getting stuck.
10. Fixed an issue with percentages retrieved by force- and player methods.
11. Return method of PlayerEntity.SetAction is now a Boolean.
12. Fixed offsets for the force members again (the fix wasn’t included before, mea culpa).

26 November, 2010

1. Fixed an issue with the AbilityList, allowing invalid abilities (Thanks Bytecompile).
2. Fixed offsets, had the current/maximum HP for force members reversed (Thanks Bytecompile).
3. Removed script validation from CS/VB scripts. Your code is your responsibility!

19 November, 2010

1. Added global Include method. Scripts can now include and call other scripts (even other languages).
2. Added global Travel method. Scripts can now load different travel lists (which opens sweet posibilities).
3. Added global Register method for CS/VB scripts. These scripts can now easily bind keys too!
4. Added support for 2.1.0.1 and a convenient version switcher (to support older private servers).
5. Changed the GetInventoryByIndex method to require an integer, instead of an unsigned integer.
6. Changed player input to pause until a click/key is handled by the game, which is more reliable.
7. Fixed an issue with character detection. Sometimes the overlay would appear too early!
8. Fixed an issue with EntityState crashing the application.
9. Fixed an issue with the GetWorld method. Instances do not change the identifier now.

10 November, 2010

1. Added travel list options (Reversed/Circular), they went MIA for some reason.
2. Fixed an issue with incorrect identifier detection.
3. Fixed a potential issue with movement detection.
4. Fixed a serious issue with different OS languages.
5. Fixed a serious issue with the travel list never being loaded (Thanks TheyRot).
6. Fixed an issue with the inventory list by index going wrong.

04 November, 2010

1. Added the Inventory method in PlayerInput. It uses an inventory item using name/id.
2. Added the GetName method in Player. It Retrieve your name without entities.
3. Added Pick Process. Detects multiple windows and allows you to select one (Dualbox).
4. Added functionality to register functions to hotkeys (including modifiers).

5. Fixed a serious issue in the EntityList which could trigger serious exceptions.
6. Fixed a serious issue with process detection. Initialization/closing is now perfect.
7. Removed the GetEntity method from Force. Perform entity retrieval through EntityList.
8. Removed the Send method from PlayerInput. Use Ability, Console and Inventory instead.
9. Removed dependencies on AutoItHelper and AutoItX3. Native functionality is used instead.
10. Updated interface design. This is the initial stage, more updates will come as I go along.
11. Updated AionInterface to support IntelliSense for Visual Studio integration.
12. Updated some internal logic to have a better performance, nothing too fancy.

21 October, 2010

1. Added GetBrand to Player, which allows you to read branding states.

17 October, 2010

1. Added InventoryList and Inventory classes to interact with your inventory.
2. Added the Inventory method in PlayerInput, which will executes inventory items.
3. Changed the update checks to be less error-prone (Read: Human error prone).
4. Renamed eType and eTypeSecondary (Skill) to eSkillType and eSkillTypeSecondary respectively.
5. Updated the Offsets.xml to include new addresses for the Inventory system.
6. Updated the radar extension to have different images for kisks and gatherables.
7. Added reload buttons to the script and node list, this is useful when you add scripts via 3th force apps.

15 October, 2010

1. Added a flexible updater to the application; checks for new files and downloads/extracts them.
2. Fixed an issue with OnClose calling OnLoad before attempting to close it (When it didn't happen).
3. Fixed security issues with VB.NET and C# scripts. Now they can only access exposed methods.
4. Updated the editor to support CS/VB.NET syntax highlighting with correct autocomplete.
5. Updated the travel path editor to look a little better and be easier to read.

14 October, 2010

1. Added an authentication system to validate users before allowing them to use the application.
2. Added a fix for TravelList when dealing with aerial nodes, which toggles correctly now.
3. Added a method in the player class to see in which world a user currently is.
4. Added a teleportation list to the Cheating extension. You can add locations with a button!
5. Added a sanity check to monsters disappearing so they can't be seen as alive monsters.
6. Added check in SetMove to allow clearing of a movement state with null/nil.
7. Added the new GetWorld() function and added it to the Player class documentation.
8. Added possibility of writing and executing csharp scripts (Which is faster but is harder to write).
9. Fixed overwriting files when creating a new script/node.
10. Fixed console input to use a combined method of new- and old, bringing the best of both worlds!
11. Fixed SetAction to allow strings again and fixed the action setter method.
12. Fixed IsBusy(), IsDead() and IsFriendly() to support actions and work faster.
13. Updated the interface to reflect changes that are brought forward by csharp scripting support.

06 October, 2010

1. Changed the console mode to use messaging instead (Has both positive and negative points).
2. Changed the internal memory reading to use a more efficient method (Rewritten Walker class).
3. Renamed the method GetReverse to IsReverse in TravelList.
4. TravelList supports movement correctly now. Modified structure to allow nodes in the air.

26 September, 2010

1. Added travel system, which allows users to easily create a travel list containing positions that a script can use. This way you can create scripts that move around to kill or gather, depending on your preferences. For more information, take a look at the documentation for Travel and TravelList.
2. Fixed teleport-to-target in the cheating extension (This was a rather funny mistake).
3. Made initial preparation for a new console-interface system as 2.0.0.3 broke the current method.

18 September, 2010

1. Added an extensions tab showing information about loaded extensions (extensions show these changes).
2. Added player mouse input into *Offsets.xml*. These weren't included in the file before.
3. Added documentation for the global methods Close, Time and Write (Very useful information indeed).
4. Fixed an issue with the offset handler not allowing comments in the file (it does allow it now).
5. Fixed an issue that caused script editor to ask whether or not to save, even without file changes.
6. Fixed broken code with position conflicting between the extended entity and the node.
7. Fixed the radar extension by adding a distance check on each entity (for some private servers).
8. Removed SetGravity from AionInterface. It is only available to the cheating extension now.
9. Removed the gathering state from Entities, it can only be checked on the player using actions.
10. Updated the interface design of the entire application. More visual changes will be added later.
11. Updated the Offsets.xml file for version 2.0.0.3 (Some values are missing right now).

09 September, 2010

1. Added missing Class Overview information to the Game Classes Reference in the documentation.
2. Added missing Game Enumerators Reference to the documentation and added links to the game classes.
3. Added missing variable information in Vector3D class to the documentation.
4. Added missing Extension Reference to the documentation.
5. Added synonyms for X, Y and Z-axis in the Vector3D class. Now accessible using Yaw, Pitch and Roll.
6. Fixed an issue with the double click handler in the cheating extension. Double clicking will now update.
7. Fixed the settings and key handler in the entities extension. Pressing 'S' will show the settings.
8. Removed the Radar from AionScript and made it an extension instead. Pressing 'S' will show the settings.
9. Added loading and saving of interface positions, so windows will position back to the last seen position.
10. Fixed an issue with interface windows not going to the front when clicked.
11. Fixed an issue with the scripting engine pausing the scanning thread.
12. Added protection to teleportation on a target in the cheating interface. Falling damage will be warned for!
13. Started interface design change. New features have been placed and will be implemented later.

07 September, 2010

1. Initial draft of the document, which is not yet entirely completed.
2. Missing enumerator information (eAttitude, eClass, eDispell, eAction, eStance, eType, eTypeSecondary).
3. Missing information about writing extensions and linking AionInterface with an external project.