|  |
| --- |
| API |
| From Valve Developer Community |
| < [Dota 2 Workshop Tools:zh-cn](file:///C:\wiki\Dota_2_Workshop_Tools:zh-cn)‎ | [Scripting:zh-cn](file:///C:\wiki\Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn) |
| Jump to: [navigation](#mw-navigation), [search](#p-search) |
|  |
| **Note:** This page is automatically generated. Any changes may be overwritten |
|  |
| Accessing the DOTA 2 Scripting API from Lua |
| While Lua is [dynamically typed](http://en.wikipedia.org/wiki/Dynamically_typed), the DOTA 2 engine is written primarily in C++, which is [statically typed](http://en.wikipedia.org/wiki/Type_system#Static_type-checking). Thus, you'll need to be conscious of your data types when calling the API. (If you try to pass the wrong type to an API function, you'll get an error message in Vconsole telling you what you passed and what it was expecting.) |
| Contents |
| * [1 **Accessing the DOTA 2 Scripting API from Lua**](#Accessing_the_DOTA_2_Scripting_API_from) |
| * [**2 Global**](#Global) |
| * [**3 CBaseEntity**](#CBaseEntity) |
| * [**4 CEntities**](#CEntities) |
| * [**5 CEntityInstance**](#CEntityInstance) |
| * [**6 CDOTABaseAbility**](#CDOTABaseAbility) |
| * [**7 CDOTA\_Ability\_Animation\_Attack**](#CDOTA_Ability_Animation_Attack) |
| * [**8 CDOTA\_Ability\_Animation\_TailSpin**](#CDOTA_Ability_Animation_TailSpin) |
| * [**9 CDOTA\_Ability\_Nian\_Leap**](#CDOTA_Ability_Nian_Leap) |
| * [**10 CDOTA\_Ability\_Nian\_Dive**](#CDOTA_Ability_Nian_Dive) |
| * [**11 CDOTA\_Ability\_Nian\_Roar**](#CDOTA_Ability_Nian_Roar) |
| * [**12 CDOTA\_Item**](#CDOTA_Item) |
| * [**13 CDOTA\_Item\_Physical**](#CDOTA_Item_Physical) |
| * [**14 CDOTA\_Item\_DataDriven**](#CDOTA_Item_DataDriven) |
| * [**15 CDOTA\_Unit\_Nian**](#CDOTA_Unit_Nian) |
| * [**16 CBasePlayer**](#CBasePlayer) |
| * [**17 CDOTAPlayer**](#CDOTAPlayer) |
| * [**18 CDOTA\_PlayerResource**](#CDOTA_PlayerResource) |
| * [**19 CDOTA\_BaseNPC**](#CDOTA_BaseNPC) |
| * [**20 CDOTA\_BaseNPC\_Hero**](#CDOTA_BaseNPC_Hero) |
| * [**21 CDOTA\_BaseNPC\_Creature**](#CDOTA_BaseNPC_Creature) |
| * [**22 CDOTA\_BaseNPC\_Building**](#CDOTA_BaseNPC_Building) |
| * [**23 CRPG\_Unit**](#CRPG_Unit) |
| * [**24 CDOTABaseGameMode**](#CDOTABaseGameMode) |
| * [**25 CDotaQuest**](#CDotaQuest) |
| * [**26 CDotaSubquestBase**](#CDotaSubquestBase) |
| * [**27 CPhysicsComponent**](#CPhysicsComponent) |
| * [**28 CPointTemplate**](#CPointTemplate) |
| * [**29 CBodyComponent**](#CBodyComponent) |
| * [**30 CBaseAnimating**](#CBaseAnimating) |
| * [**31 CBaseCombatCharacter**](#CBaseCombatCharacter) |
| * [**32 ProjectileManager**](#ProjectileManager) |
| * [**33 CBaseTrigger**](#CBaseTrigger) |
| * [**34 CEnvEntityMaker**](#CEnvEntityMaker) |
| * [**35 CDOTAVoteSystem**](#CDOTAVoteSystem) |
| * [**36 CMarkupVolumeTagged**](#CMarkupVolumeTagged) |
| * [**37 CScriptPrecacheContext**](#CScriptPrecacheContext) |
| * [**38 CScriptKeyValues**](#CScriptKeyValues) |
| * [**39 CScriptParticleManager**](#CScriptParticleManager) |
| * [**40 CScriptHeroList**](#CScriptHeroList) |
| * [**41 CNativeOutputs**](#CNativeOutputs) |
| * [**42 CEnvProjectedTexture**](#CEnvProjectedTexture) |
| * [**43 CInfoData**](#CInfoData) |
| * [**44 CPhysicsProp**](#CPhysicsProp) |
| * [**45 CDOTAGamerules**](#CDOTAGamerules) |
| * [**46 CToneMapControllerComponent**](#CToneMapControllerComponent) |
| * [**47 CDebugOverlayScriptHelper**](#CDebugOverlayScriptHelper) |
| * [**48 CBaseFlex**](#CBaseFlex) |
| * [**49 CSceneEntity**](#CSceneEntity) |
| * [**50 GridNav**](#GridNav) |
| * [**51 Convars**](#Convars) |
| * [**52 Vector**](#Vector) |
| * [**53 CDOTA\_MapTree**](#CDOTA_MapTree) |
| * [**54 CDOTA\_SimpleObstruction**](#CDOTA_SimpleObstruction) |
| Global |
| ***Global functions. These can be called without any class*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AngleDiff](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.AngleDiff) | float AngleDiff(float ang1, float ang2) | 返回两个偏航角度之差 | | [AppendToLogFile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.AppendToLogFile) | void AppendToLogFile(string a, string b) | 在服务器日志文件上增加一个string | | [ApplyDamage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ApplyDamage) | float ApplyDamage(handle DamageTable) | 对单位造成伤害，Table输入：victim, attacker, damage, damage\_type, damage\_flags, ability | | [AxisAngleToQuaternion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.AxisAngleToQuaternion) | Quaternion AxisAngleToQuaternion(Vector a, float b) | (vector,float) 构造一个四元数，表现出围绕给定的vector 轴线的旋转角 | | [CancelEntityIOEvents](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CancelEntityIOEvents) | void CancelEntityIOEvents(ehandle a) | 为特定的实体创建所有输入/输出事件 | | [CreateEffect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateEffect) | bool CreateEffect(handle a) | 传递table - 输入：实体，效果 | | [CreateHeroForPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateHeroForPlayer) | handle CreateHeroForPlayer(string unitName, handle player) | 用DOTA英雄的dota\_npc\_units.txt中的名字创建它并把它设为给定玩家的控制英雄 | | [CreateItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateItem) | handle CreateItem(string item\_name, handle owner, handle owner) | 用类名称item\_name创建一个owner可以使用的物品 | | [CreateItemOnPositionSync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateItemOnPositionSync) | handle CreateItemOnPositionSync(Vector a, handle b) | 在给定位置创建一个可见的物品 | | [CreateTrigger](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateTrigger) | handle CreateTrigger(Vector a, Vector b, Vector c) | CreateTrigger( vecMin, vecMax ) : 创建并返回一个AABB触发器 | | [CreateTriggerRadiusApproximate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateTriggerRadiusApproximate) | handle CreateTriggerRadiusApproximate(Vector a, float b) | CreateTriggerRadiusApproximate( vecOrigin, flRadius ) : 创建并返回一个比给定radius大的AABB触发器 | | [CreateUnitByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateUnitByName) | handle CreateUnitByName(string a, Vector b, bool c, handle d, handle e, int f) | 用DOTA单位的dota\_npc\_units.txt中的名字创建它( szUnitName, vLocation, bFindClearSpace, hNPCOwner, hUnitOwner, iTeamNumber ) | | [CreateUnitByNameAsync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateUnitByNameAsync) | int CreateUnitByNameAsync(string a, Vector b, bool c, handle d, handle e, int f, handle g) | 用DOTA单位的dota\_npc\_units.txt中的名字创建它( szUnitName, vLocation, bFindClearSpace, hNPCOwner, hUnitOwner, iTeamNumber, hCallback ) | | [CreateUnitFromTable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.CreateUnitFromTable) | hscript CreateUnitFromTable(table unitProperties) | Creates a new data-driven entity with the given table | | [cvar\_getf](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.cvar_getf) | float cvar\_getf(string a) | 获取给定cvar的值，返回float. | | [cvar\_setf](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.cvar_setf) | bool cvar\_setf(string a, float b) | 设置给定cvar的值, 传递float. | | [DebugBreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugBreak) | void DebugBreak() | 在debugger（调试模式）中停止运行 | | [DebugDrawBox](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawBox) | void DebugDrawBox(Vector origin, Vector min, Vector max, int r, int g, int b, int a, float duration) | 显示一个debug overlay窗口(origin, mins, maxs, r, g, b, a, duration ) | | [DebugDrawBoxDirection](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawBoxDirection) | void DebugDrawBoxDirection(Vector a, Vector b, Vector c, Vector d, Vector e, float f, float g) | 显示一个debug forward窗口(cent, min, max, forward, vRgb, a, duration) | | [DebugDrawCircle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawCircle) | void DebugDrawCircle(Vector a, Vector b, float c, float d, bool e, float f) | 显示一个debug圆(center, vRgb, a, rad, ztest, duration) | | [DebugDrawClear](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawClear) | void DebugDrawClear() | 尝试清理所有debug overlay信息 | | [DebugDrawLine](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawLine) | void DebugDrawLine(Vector a, Vector b, int c, int d, int e, bool f, float g) | 显示一个debug overlay线段origin, target, r, g, b, ztest, duration) | | [DebugDrawLine\_vCol](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawLine_vCol) | void DebugDrawLine\_vCol(Vector a, Vector b, Vector c, bool d, float e) | 使用vec颜色显示一个debug线段(start, end, vRgb, a, ztest, duration) | | [DebugDrawScreenTextLine](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawScreenTextLine) | void DebugDrawScreenTextLine(float a, float b, int c, string d, int e, int f, int g, int h, float i) | 显示给定行间距（lineoffset）的文字(x, y, lineOffset, text, r, g, b, a, duration) | | [DebugDrawSphere](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawSphere) | void DebugDrawSphere(Vector a, Vector b, float c, float d, bool e, float f) | 显示一个debug球(center, vRgb, a, rad, ztest, duration) | | [DebugDrawText](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugDrawText) | void DebugDrawText(Vector a, string b, bool c, float d) | 显示3d文字(origin, text, bViewCheck, duration) | | [DebugScreenTextPretty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DebugScreenTextPretty) | void DebugScreenTextPretty(float a, float b, int c, string d, int e, int f, int g, int h, float i, string j, int k, bool l) | 显示漂亮的debug文字(x, y, lineOffset, text, r, g, b, a, duration, font, size, bBold) | | [DoEntFire](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DoEntFire) | void DoEntFire(string a, string b, string c, float d, handle e, handle f) | EntFire: 构造一个实体的输入/输出事件( szTarget, szAction, szValue, flDelay, hActivator, hCaller ) | | [DoEntFireByInstanceHandle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DoEntFireByInstanceHandle) | void DoEntFireByInstanceHandle(handle a, string b, string c, float d, handle e, handle f) | EntFireByHandle:构造一个实体的输入/输出事件 | | [DoIncludeScript](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DoIncludeScript) | bool DoIncludeScript(string a, handle b) | 运行一个内部脚本 | | [DoScriptAssert](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DoScriptAssert) | void DoScriptAssert(bool a, string b) | ScriptAssert:断言传入数据是数值，输出信息并显示断言对话框（参考编程中的Assert功能，主要用于调试，译者注） | | [DoUniqueString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.DoUniqueString) | string DoUniqueString(string a) | UniqueString:使用任意的根字符串构造一个保证在VM脚本运行周期内唯一的字符串。当给表增加数据但不确定已用关键字时有用 | | [EmitGlobalSound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.EmitGlobalSound) | void EmitGlobalSound(string a) | 为所有玩家播放给定名称的音效 | | [EmitSoundOn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.EmitSoundOn) | void EmitSoundOn(string a, handle b) | 在一个实体上播放给定名称的音效 | | [EmitSoundOnClient](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.EmitSoundOnClient) | void EmitSoundOnClient(string a, handle b) | 为指定玩家播放给定名称的音效 | | [EntIndexToHScript](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.EntIndexToHScript) | handle EntIndexToHScript(int a) | 把一个实体的整数索引转化为表达该实体脚本实例的HScript | | [ExecuteOrderFromTable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ExecuteOrderFromTable) | void ExecuteOrderFromTable(handle a) | 从一个Script表发布命令 | | [ExponentialDecay](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ExponentialDecay) | float ExponentialDecay(float a, float b, float c) | 平滑的曲线，接近0时减少较慢 | | [FileToString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FileToString) | string FileToString(string a) | 从文件读取字符串给脚本，目录遍历似乎无效，字符'.','/'和'\\'被忽略，文件从C:\Program Files (x86)\Steam\steamapps\common\dota 2 beta\dota\_ugc\game\dota\ems读取，函数的用途与StringToFile相反 | | [FindClearSpaceForUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FindClearSpaceForUnit) | void FindClearSpaceForUnit(handle a, Vector b, bool c) | 在未被占用的地方创建单位 | | [FindUnitsInRadius](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FindUnitsInRadius) | table FindUnitsInRadius(int teamNumber, Vector position, handle cacheUnit, float radius, int teamFilter, int typeFilter, int flagFilter, int order, bool canGrowCache) | 在给定范围内用给定flags搜索单位( iTeamNumber, vPosition, hCacheUnit, flRadius, iTeamFilter, iTypeFilter, iFlagFilter, iOrder, bCanGrowCache ) | | [FireEntityIOInputNameOnly](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FireEntityIOInputNameOnly) | void FireEntityIOInputNameOnly(ehandle a, string b) | 触发实体的动作输入w/no数据 | | [FireEntityIOInputString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FireEntityIOInputString) | void FireEntityIOInputString(ehandle a, string b, string c) | 用给定字符串（由你分配内存）触发实体的动作输入 | | [FireEntityIOInputVec](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FireEntityIOInputVec) | void FireEntityIOInputVec(ehandle a, string b, Vector c) | 用给定的Vector( hEntity, szActionName, vector )触发实体的动作输入 | | [FireGameEvent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FireGameEvent) | void FireGameEvent(string eventName, handle parameterTable) | 触发一个预定义事件，可以在custom\_events.txt或dota的资源/\*.res中找到 | | [FireGameEventLocal](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FireGameEventLocal) | void FireGameEventLocal(string a, handle b) | 触发一个游戏事件但不广播给客户端 | | [FrameTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.FrameTime) | float FrameTime() | 获取上一帧在服务器上花费的时间 | | [GetFrameCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetFrameCount) | int GetFrameCount() | 返回引擎当然的帧数 | | [GetFrostyBoostAmount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetFrostyBoostAmount) | float GetFrostyBoostAmount(int a, int b) | No Description Set | | [GetFrostyPointsForRound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetFrostyPointsForRound) | int GetFrostyPointsForRound(int a, int b, int c) | No Description Set | | [GetGoldFrostyBoostAmount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetGoldFrostyBoostAmount) | float GetGoldFrostyBoostAmount(int a, int b) | No Description Set | | [GetGoldFrostyPointsForRound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetGoldFrostyPointsForRound) | int GetGoldFrostyPointsForRound(int a, int b, int c) | No Description Set | | [GetGroundPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetGroundPosition) | Vector GetGroundPosition(Vector a, handle b) | 返回移动到地面的供给位置。第二个参数是一个NPC，用于测量碰撞体积 | | [GetListenServerHost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetListenServerHost) | handle GetListenServerHost() | 在广播服务器获取本地玩家 | | [GetMapName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetMapName) | string GetMapName() | 获取地图的名字 | | [GetMaxOutputDelay](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetMaxOutputDelay) | float GetMaxOutputDelay(ehandle a, string b) | 获取输出事件的最大延迟 | | [GetPhysAngularVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetPhysAngularVelocity) | Vector GetPhysAngularVelocity(handle a) | 获取VPHYS或普通物体的角速度 | | [GetPhysVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetPhysVelocity) | Vector GetPhysVelocity(handle a) | 获取VPHYS或普通物体的速度 | | [GetSystemDate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetSystemDate) | string GetSystemDate() | 获取真实世界的日期 | | [GetSystemTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetSystemTime) | string GetSystemTime() | 获取真实世界的时间 | | [GetTeamName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetTeamName) | string GetTeamName(int teamNumber) | Returns the team name | | [GetWorldMaxX](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetWorldMaxX) | float GetWorldMaxX() | 获取地图X坐标最大值 | | [GetWorldMaxY](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetWorldMaxY) | float GetWorldMaxY() | 获取地图Y坐标最大值 | | [GetWorldMinX](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetWorldMinX) | float GetWorldMinX() | 获取地图X坐标最小值 | | [GetWorldMinY](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.GetWorldMinY) | float GetWorldMinY() | 获取地图Y坐标最小值 | | [InitLogFile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.InitLogFile) | void InitLogFile(string a, string b) | 如果给定文件不存在，用给定内容创建它；如果存在则不做任何动作 | | [IsDedicatedServer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.IsDedicatedServer) | bool IsDedicatedServer() | 如果服务器开放返回true | | [IsMarkedForDeletion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.IsMarkedForDeletion) | bool IsMarkedForDeletion(handle a) | 如果实体有效并被标记为删除返回true | | [IsValidEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.IsValidEntity) | bool IsValidEntity(handle a) | 检查给定hScript是否为有效实体 | | [ListenToGameEvent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ListenToGameEvent) | int ListenToGameEvent(string EventName, handle functionNameToCall, handle context) | 从脚本给游戏事件注册一个listner，Tip:另外，为listenstandard engine events,你也可以创建自己的事件，把它们放置在/scripts/custom\_events.txt. | | [LoadKeyValues](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.LoadKeyValues) | table LoadKeyValues(string a) | 用给定的关键字文件创建table | | [LoadKeyValuesFromString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.LoadKeyValuesFromString) | table LoadKeyValuesFromString(string a) | 用给定的关键字string创建table | | [MakeStringToken](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.MakeStringToken) | int MakeStringToken(string a) | 检查给定的hScript是否是有效实体（？） | | [Msg](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.Msg) | void Msg(string a) | 输出信息 | | [PauseGame](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PauseGame) | void PauseGame(bool a) | 暂停/开始游戏 | | [PlayerInstanceFromIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PlayerInstanceFromIndex) | handle PlayerInstanceFromIndex(int a) | 通过索引获取玩家的脚本实例 | | [PrecacheEntityFromTable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheEntityFromTable) | void PrecacheEntityFromTable(string a, handle b, handle c) | 用关键字从table中预缓存一个实体 | | [PrecacheEntityListFromTable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheEntityListFromTable) | void PrecacheEntityListFromTable(handle a, handle b) | 预缓存一个实体关键字列表 | | [PrecacheItemByNameAsync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheItemByNameAsync) | void PrecacheItemByNameAsync(string a, handle b) | 用DOTA物品的dota\_npc\_items.txt名称异步预缓存它,结束时提供回调 | | [PrecacheItemByNameSync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheItemByNameSync) | void PrecacheItemByNameSync(string a, handle b) | 用DOTA物品的dota\_npc\_items.txt名称预缓存它 | | [PrecacheModel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheModel) | void PrecacheModel(string a, handle b) | ( modelName, context ) - 手动预缓存一个单一模型 | | [PrecacheResource](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheResource) | void PrecacheResource(string precacheMode, string resource, handle context) | 手动预缓存一个单一资源 | | [PrecacheUnitByNameAsync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheUnitByNameAsync) | void PrecacheUnitByNameAsync(string a, handle b) | 用DOTA单位的dota\_npc\_units.txt名称异步预缓存它,结束时提供回调 | | [PrecacheUnitByNameSync](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrecacheUnitByNameSync) | void PrecacheUnitByNameSync(string a, handle b) | 用DOTA单位的dota\_npc\_units.txt名称预缓存它 | | [PrintLinkedConsoleMessage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.PrintLinkedConsoleMessage) | void PrintLinkedConsoleMessage(string a, string b) | Print a console message with a linked console command | | [RandomFloat](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RandomFloat) | float RandomFloat(float a, float b) | 获取范围内随机float | | [RandomInt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RandomInt) | int RandomInt(int a, int b) | 获取范围内随机int | | [RandomVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RandomVector) | Vector RandomVector(float maxLength) | 获取随机二维矢量 | | [RegisterSpawnGroupFilterProxy](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RegisterSpawnGroupFilterProxy) | void RegisterSpawnGroupFilterProxy(string a) | 为一个脚本基础的生成单位过滤器创建一个C代理 | | [ReloadMOTD](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ReloadMOTD) | void ReloadMOTD() | 重载MotD文件 | | [RemoveSpawnGroupFilterProxy](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RemoveSpawnGroupFilterProxy) | void RemoveSpawnGroupFilterProxy(string a) | 为一个脚本基础的生成单位过滤器删除一个C代理 | | [RollPercentage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RollPercentage) | bool RollPercentage(int a) | 随机生成1-100内的数，小于等于给定数则返回true | | [RotateOrientation](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RotateOrientation) | QAngle RotateOrientation(QAngle a, QAngle b) | 旋转一个QAngle，用另一个QAngle | | [RotatePosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RotatePosition) | Vector RotatePosition(Vector a, QAngle b, Vector c) | 围绕点旋转一个Vector | | [RotateQuaternionByAxisAngle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RotateQuaternionByAxisAngle) | Quaternion RotateQuaternionByAxisAngle(Quaternion a, Vector b, float c) | 围绕给定vector轴线旋转一个四元数 | | [RotationDelta](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.RotationDelta) | QAngle RotationDelta(QAngle a, QAngle b) | 两个QAngle间的delta值 | | [rr\_AddDecisionRule](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.rr_AddDecisionRule) | bool rr\_AddDecisionRule(handle a) | 给决定数据库增加一个规则 | | [rr\_CommitAIResponse](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.rr_CommitAIResponse) | bool rr\_CommitAIResponse(handle a, handle b) | 把QueryBestResponse的结果交付给定的实体播放，用参数调用(entity, airesponse) | | [rr\_GetResponseTargets](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.rr_GetResponseTargets) | handle rr\_GetResponseTargets() | 获取所有提供的表达者目标的table，格式为{ name : handle, name: handle }. | | [rr\_QueryBestResponse](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.rr_QueryBestResponse) | bool rr\_QueryBestResponse(handle a, handle b, handle c) | 参数: ( hEnt, hQuery, hResult ) // 静态 : 对实体的回复系统测试 'query'并返回最佳回复(无回复返回nil) | | [Say](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.Say) | void Say(handle entity, string message, bool teamOnly) | 让实体说string,是否仅限队友可见 | | [ScreenShake](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ScreenShake) | void ScreenShake(Vector a, float b, float c, float d, float e, int f, bool g) | 用下列参数开始镜头摇晃：vecCenter, flAmplitude, flFrequency, flDuration, flRadius, eCommand( SHAKE\_START = 0, SHAKE\_STOP = 1 ), bAirShake | | [SendFrostivusTimeElapsedToGC](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SendFrostivusTimeElapsedToGC) | void SendFrostivusTimeElapsedToGC() | No Description Set | | [SendFrostyPointsMessageToGC](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SendFrostyPointsMessageToGC) | void SendFrostyPointsMessageToGC(handle a) | No Description Set | | [SendToConsole](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SendToConsole) | void SendToConsole(string a) | 给控制台发送string作为客户端命令 | | [SendToServerConsole](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SendToServerConsole) | void SendToServerConsole(string a) | 给控制台发送string作为服务器命令 | | [SetOpvarFloatAll](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SetOpvarFloatAll) | void SetOpvarFloatAll(string a, string b, string c, float d) | 为所有玩家设置opvar值 | | [SetOpvarFloatPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SetOpvarFloatPlayer) | void SetOpvarFloatPlayer(string a, string b, string c, float d, handle e) | 为单一玩家设置opvar值( szStackName, szOperatorName, szOpvarName, flOpvarValue, hEnt ) | | [SetQuestName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SetQuestName) | void SetQuestName(string a) | 设置当前任务名 | | [SetQuestPhase](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SetQuestPhase) | void SetQuestPhase(int a) | 设置当前任务阶段 | | [SetRenderingEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SetRenderingEnabled) | void SetRenderingEnabled(ehandle a, bool b) | 为ehandle设置rendering（渲染）开/关 | | [ShowCustomHeaderMessage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ShowCustomHeaderMessage) | void ShowCustomHeaderMessage(string message, int player, int value, float time) | Displays a header message from player | | [ShowGenericPopup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ShowGenericPopup) | void ShowGenericPopup(string title, string content, string unknown, string unknown, int containerType) | 对所有玩家显示通用的弹出窗口 | | [ShowGenericPopupToPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ShowGenericPopupToPlayer) | void ShowGenericPopupToPlayer(handle a, string b, string c, string d, string e, int f) | 对给定玩家显示通用的弹出窗口 | | [ShowMessage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.ShowMessage) | void ShowMessage(string a) | 在所有客户端显示hud信息 | | [SpawnEntityFromTableSynchronous](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SpawnEntityFromTableSynchronous) | handle SpawnEntityFromTableSynchronous(string a, handle b) | 从table同步生成一个实体 | | [SpawnEntityGroupFromTable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SpawnEntityGroupFromTable) | bool SpawnEntityGroupFromTable(handle groupSpawnTables, bool bAsync, handle hCallback) | 从一系列生成表中分级生成一个实体组 | | [SpawnEntityListFromTableAsynchronous](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SpawnEntityListFromTableAsynchronous) | int SpawnEntityListFromTableAsynchronous(handle a, handle b) | 从一个生成表列表异步生成一个实体组，生成完成后会触发一次回调 | | [SpawnEntityListFromTableSynchronous](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SpawnEntityListFromTableSynchronous) | handle SpawnEntityListFromTableSynchronous(handle a) | 从一个生成表列表同步生成一个实体组. | | [SplineQuaternions](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SplineQuaternions) | Quaternion SplineQuaternions(Quaternion a, Quaternion b, float c) | (quaternion,quaternion,float)非常基础的v0到v1差值，t属于[0,1]为参数 | | [SplineVectors](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.SplineVectors) | Vector SplineVectors(Vector a, Vector b, float c) | (vector,vector,float)非常基础的v0到v1差值，t属于[0,1]为参数 | | [StartSoundEvent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StartSoundEvent) | void StartSoundEvent(string a, handle b) | 开始音效事件 | | [StopEffect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StopEffect) | void StopEffect(handle a, string b) | (hEntity, szEffectName) | | [StopListeningToAllGameEvents](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StopListeningToAllGameEvents) | void StopListeningToAllGameEvents(handle a) | 停止listen在给定内容内的所有游戏事件 | | [StopListeningToGameEvent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StopListeningToGameEvent) | bool StopListeningToGameEvent(int a) | 停止listen特定的游戏事件 | | [StopSoundEvent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StopSoundEvent) | void StopSoundEvent(string a, handle b) | 停止音效事件 | | [StopSoundOn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StopSoundOn) | void StopSoundOn(string soundName, handle playingEntity) | 停止实体上的命名音效 | | [StringToFile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.StringToFile) | bool StringToFile(string a, string b) | \* 目录遍历似乎无效，字符'.','/'和'\\'被忽略，文件名不能为空，也不能包含忽略字符，否则函数无效并返回false，一次只能保存一个字符串，每次调用文件都被覆盖，文件以一个NUL字符结尾，文件被保存在C:\Program Files (x86)\Steam\steamapps\common\dota 2 beta\dota\_ugc\game\dota\ems，函数用途与FileToString相反 | | [Time](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.Time) | float Time() | 获取当前服务器时间 | | [TraceCollideable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.TraceCollideable) | bool TraceCollideable(handle a) | 传递table - 输入: start, end, ent, (可选 mins, maxs) -- 输出: pos, fraction, hit, startsolid, normal | | [TraceHull](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.TraceHull) | bool TraceHull(handle a) | 传递table - 输入: start, end, min, max, mask, ignore -- 输出: pos, fraction, hit, enthit, startsolid | | [TraceLine](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.TraceLine) | bool TraceLine(handle a) | Pass table - 输入: startpos, endpos, mask, ignore -- 输出: pos, fraction, hit, enthit, startsolid | | [UnloadSpawnGroup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UnloadSpawnGroup) | void UnloadSpawnGroup(string a) | 通过名称去除生成组 | | [UnloadSpawnGroupByHandle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UnloadSpawnGroupByHandle) | void UnloadSpawnGroupByHandle(int a) | 通过handle去除生成组 | | [UpdateEventPoints](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UpdateEventPoints) | void UpdateEventPoints(handle a) | No Description Set | | [UTIL\_MessageText](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_MessageText) | void UTIL\_MessageText(int playerId, string message, int r, int g, int b, int a) | Displays a message for a specific player | | [UTIL\_MessageText\_WithContext](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_MessageText_WithContext) | void UTIL\_MessageText\_WithContext(int playerId, string message, int r, int g, int b, int a, table context) | Sends a message to a specific player in the message box with a context table | | [UTIL\_MessageTextAll](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_MessageTextAll) | void UTIL\_MessageTextAll(string message, int r, int g, int b, int a) | Sends a message to everyone in the message box | | [UTIL\_MessageTextAll\_WithContext](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_MessageTextAll_WithContext) | void UTIL\_MessageTextAll\_WithContext(string message, int r, int g, int b, int a, table context) | Sends a message to everyone in the message box with a context table | | [UTIL\_Remove](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_Remove) | void UTIL\_Remove(handle a) | 删除给定实体 | | [UTIL\_RemoveImmediate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_RemoveImmediate) | void UTIL\_RemoveImmediate(handle a) | 立即删除给定实体 | | [UTIL\_ResetMessageText](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_ResetMessageText) | void UTIL\_ResetMessageText(int playerId) | Resets the message text for the player | | [UTIL\_ResetMessageTextAll](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.UTIL_ResetMessageTextAll) | void UTIL\_ResetMessageTextAll() | Resets the message text for all players | | [VectorToAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.VectorToAngles) | QAngle VectorToAngles(Vector a) | 为Vector获取Qangle（没有滚动） | | [Warning](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Global.Warning) | void Warning(string a) | 输出警告 | |
|  |
| CBaseEntity |
| ***The base class for stuff*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ApplyAbsVelocityImpulse](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.ApplyAbsVelocityImpulse) | void ApplyAbsVelocityImpulse(Vector a) | 添加一个速度冲击 | | [ApplyLocalAngularVelocityImpulse](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.ApplyLocalAngularVelocityImpulse) | void ApplyLocalAngularVelocityImpulse(Vector a) | 添加一个角速度的冲击 | | [EmitSound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.EmitSound) | void EmitSound(string soundName) |  | | [EmitSoundParams](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.EmitSoundParams) | void EmitSoundParams(string soundName, int pitch, float volume, float soundTime) | 播放/修改这个实体的声音.当音调,音量或播放时间大于零时改变声音 | | [EyeAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.EyeAngles) | QAngle EyeAngles() | 获得这个实体的视角qangles | | [EyePosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.EyePosition) | Vector EyePosition() | 获得指向视角位置的矢量,位置为绝对坐标 | | [FirstMoveChild](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.FirstMoveChild) | handle FirstMoveChild() | No Description Set | | [GatherCriteria](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GatherCriteria) | void GatherCriteria(handle a) | 返回一个表包含用在这个实体的判定的标准.这个表等同于传入脚本方法的调用. | | [GetAbsOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetAbsOrigin) | Vector GetAbsOrigin() | No Description Set | | [GetAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetAngles) | QAngle GetAngles() | No Description Set | | [GetAnglesAsVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetAnglesAsVector) | Vector GetAnglesAsVector() | 获得实体的颠簸,偏航,摇晃的矢量 | | [GetAngularVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetAngularVelocity) | Vector GetAngularVelocity() | 获得当前角速度,返回颠簸,偏航,摇晃的矢量 | | [GetBaseVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetBaseVelocity) | Vector GetBaseVelocity() | 获得基础速度 | | [GetBoundingMaxs](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetBoundingMaxs) | Vector GetBoundingMaxs() | 获得一个包含最大范围,以物体为中心的矢量 | | [GetBoundingMins](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetBoundingMins) | Vector GetBoundingMins() | 获得一个包含最小范围,以物体为中心的矢量 | | [GetBounds](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetBounds) | table GetBounds() | 获得一个以物体为中心,包含“Mins”和“Maxs”矢量的表 | | [GetCenter](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetCenter) | Vector GetCenter() | 获得指向物体中心的矢量,以绝对坐标的形式 | | [GetChildren](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetChildren) | handle GetChildren() | 获得继承这个实体的实体 | | [GetContext](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetContext) | table GetContext(string a) | GetContext( name ):检查一个对象的状态然后返回对象是否可用.可能返回string,float,或nil(如果对象没有找到) | | [GetForwardVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetForwardVector) | Vector GetForwardVector() | 获得这个实体的前进矢量 | | [GetHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetHealth) | int GetHealth() | No Description Set | | [GetLocalAngularVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetLocalAngularVelocity) | QAngle GetLocalAngularVelocity() | 可能为当前角速度 | | [GetLocalVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetLocalVelocity) | Vector GetLocalVelocity() | 获得实体相对速度 | | [GetMaxHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetMaxHealth) | int GetMaxHealth() | No Description Set | | [GetModelName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetModelName) | string GetModelName() | 获得模型的名字 | | [GetMoveParent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetMoveParent) | handle GetMoveParent() | 如果在有继承对象,恢复实体所继承的对象 | | [GetOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetOrigin) | Vector GetOrigin() | No Description Set | | [GetOwner](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetOwner) | handle GetOwner() | 获得实体的所有者 | | [GetOwnerEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetOwnerEntity) | handle GetOwnerEntity() | 获得拥有者的实体,如果有的话 | | [GetRightVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetRightVector) | Vector GetRightVector() | 获得实体的右方矢量 | | [GetRootMoveParent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetRootMoveParent) | handle GetRootMoveParent() | 如果在继承关系里,在继承关系中寻找所继承的根源对象 | | [GetSoundDuration](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetSoundDuration) | float GetSoundDuration(string soundName, string actormodelname) | 返回声音的播放时间.传入声音名字和可选的应用模型名字. | | [GetTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetTeam) | int GetTeam() | No Description Set | | [GetUpVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetUpVector) | Vector GetUpVector() | 获得实体的上方矢量 | | [GetVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.GetVelocity) | Vector GetVelocity() | No Description Set | | [IsAlive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.IsAlive) | bool IsAlive() | No Description Set. | | [IsPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.IsPlayer) | bool IsPlayer() | 是否为玩家的实体 | | [Kill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.Kill) | void Kill() | No Description Set | | [NextMovePeer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.NextMovePeer) | handle NextMovePeer() | No Description Set | | [OverrideFriction](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.OverrideFriction) | void OverrideFriction(float a, float b) | 暂时重设阻力的持续时间和值 | | [PrecacheScriptSound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.PrecacheScriptSound) | void PrecacheScriptSound(string a) | 预载声音文件 | | [SetAbsOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetAbsOrigin) | void SetAbsOrigin(Vector origin) | 设置绝对来源 | | [SetAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetAngles) | void SetAngles(float pitch, float yaw, float roll) | 设置实体的颠簸,偏航,摇晃 | | [SetAngularVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetAngularVelocity) | void SetAngularVelocity(float pitch, float yaw, float roll) | 设置当前角速度,使用实数颠簸,偏航,摇晃速度 | | [SetContext](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetContext) | void SetContext(string a, string b, float c) | SetContext( 名字 , 值, 持续时间 ):存储任何在实体对话内容里的配对值.值是一个字符串.持续一段时间.(0代表永久) | | [SetContextNum](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetContextNum) | void SetContextNum(string a, float b, float c) | SetContext( 名字 , 值, 持续时间 ):存储任何在实体对话内容里的配对值.值是一个数字(int或者float).持续一段时间.(0代表永久) | | [SetContextThink](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetContextThink) | void SetContextThink(string a, handle b, float c) | 在这个实体上设置一个计时器/DescriptionCN><HowToCall><Parameter>a:名字,b:实体? c:时间</Parameter><Return></Return><PS></PS></HowToCall> | | [SetForwardVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetForwardVector) | void SetForwardVector(Vector forwardVec) | 设置实体的方向为前进矢量 | | [SetFriction](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetFriction) | void SetFriction(float a) | 设置玩家阻力,无视物体 | | [SetGravity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetGravity) | void SetGravity(float a) | 设置玩家引力,无视物体 | | [SetHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetHealth) | void SetHealth(int hp) | No Description Set | | [SetMaxHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetMaxHealth) | void SetMaxHealth(int maxHP) | No Description Set | | [SetModel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetModel) | void SetModel(string modelName) | No Description Set | | [SetOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetOrigin) | void SetOrigin(Vector origin) | No Description Set | | [SetOwner](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetOwner) | void SetOwner(handle owningEntity) | 设定实体的所有者 | | [SetParent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetParent) | void SetParent(handle a, string b) | 设定这个实体的所继承的父实体 | | [SetRenderColor](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetRenderColor) | void SetRenderColor(int a, int b, int c) | SetRenderColor( r, g, b ):设定实体的渲染颜色 | | [SetSize](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetSize) | void SetSize(Vector a, Vector b) | No Description Set | | [SetTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetTeam) | void SetTeam(int team) | No Description Set | | [SetVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.SetVelocity) | void SetVelocity(Vector a) | No Description Set | | [StopSound](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.StopSound) | void StopSound(string soundName) | 停止播放这个实体的特定的声音 | | [Trigger](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseEntity.Trigger) | void Trigger() | 触发这个实体的触发器 | |
|  |
| CEntities |
| ***No Description Set*** |
|  |
| ***Global accessor variable:* Entities** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CreateByClassname](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.CreateByClassname) | handle CreateByClassname(string className) | 用类名来创建一个实体 | | [FindAllByClassname](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByClassname) | table FindAllByClassname(string a) | 通过类名来寻找所有实体.返回被找到实体的数组或表. | | [FindAllByClassnameWithin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByClassnameWithin) | table FindAllByClassnameWithin(string a, Vector b, float c) | 在一个半径范围内通过类名寻找实体. | | [FindAllByModel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByModel) | table FindAllByModel(string modelName) | 通过模型名字来找到对象. | | [FindAllByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByName) | table FindAllByName(string name) | 通过名字来寻找所有实体.返回被找到实体的数组或表. | | [FindAllByNameWithin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByNameWithin) | table FindAllByNameWithin(string name, Vector origin, float maxRadius) | 在半径范围内通过名字来寻找实体. | | [FindAllByTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllByTarget) | table FindAllByTarget(string targetName) | 通过目标名字来寻找实体 | | [FindAllInSphere](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindAllInSphere) | table FindAllInSphere(Vector origin, float maxRadius) | 在(球形)范围内寻找实体 | | [FindByClassname](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByClassname) | handle FindByClassname(handle startFrom, string className) | 通过类名寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByClassnameNearest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByClassnameNearest) | handle FindByClassnameNearest(string className, Vector origin, float maxRadius) | 在最近一个点通过类名寻找实体 | | [FindByClassnameWithin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByClassnameWithin) | handle FindByClassnameWithin(handle startFrom, string className, Vector origin, float maxRadius) | 在半径范围内通过类名寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByModel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByModel) | handle FindByModel(handle startFrom, string modelName) | 通过模型名字寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByModelWithin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByModelWithin) | handle FindByModelWithin(handle startFrom, string modelName, Vector origin, float maxRadius) | 在半径范围内通过模型名字寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByName) | handle FindByName(handle lastEnt, string searchString) | 通过名字寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByNameNearest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByNameNearest) | handle FindByNameNearest(string name, Vector origin, float maxRadius) | 通过名字寻找最近原点的实体 | | [FindByNameWithin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByNameWithin) | handle FindByNameWithin(handle startFrom, string name, Vector origin, float maxRadius) | 在半径范围内通过名字寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindByTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindByTarget) | handle FindByTarget(handle startFrom, string targetName) | 通过目标名字寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [FindInSphere](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.FindInSphere) | handle FindInSphere(handle startFrom, Vector origin, float maxRadius) | 在半径范围内寻找实体.传入nil就从头进行迭代寻找,传入之前找到的实体可以以之作为起点开始寻找 | | [First](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.First) | handle First() | 在实体数组或表中开始一个迭代程序 | | [Next](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntities.Next) | handle Next(handle startFrom) | 从指定列表中的实体开始迭代 | |
|  |
| CEntityInstance |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ConnectOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.ConnectOutput) | void ConnectOutput(string a, string b) | 添加一个I/O链接,这可以当特定输出触发时,在这个实体调用方法 | | [Destroy](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.Destroy) | void Destroy() | No Description Set | | [DisconnectOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.DisconnectOutput) | void DisconnectOutput(string a, string b) | 从这个实体的I/O事件移除已连接的脚本方法 | | [DisconnectRedirectedOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.DisconnectRedirectedOutput) | void DisconnectRedirectedOutput(string a, string b, handle c) | 从传入实体的I/O事件移除已连接的脚本方法 | | [entindex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.entindex) | int entindex() | No Description Set | | [FireOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.FireOutput) | void FireOutput(string outputName, handle activator, handle caller, table args, float delay) | 触发实体的输出 | | [GetClassname](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetClassname) | string GetClassname() | No Description Set | | [GetDebugName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetDebugName) | string GetDebugName() | 如果没有定义的话获得实体名字的帮助 | | [GetEntityHandle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetEntityHandle) | ehandle GetEntityHandle() | 获得这个实体的EHANDLE | | [GetEntityIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetEntityIndex) | int GetEntityIndex() | No Description Set | | [GetIntAttr](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetIntAttr) | int GetIntAttr(string a) | 获得整型属性 | | [GetName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetName) | string GetName() | No Description Set | | [GetOrCreatePrivateScriptScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetOrCreatePrivateScriptScope) | handle GetOrCreatePrivateScriptScope() | 恢复实体相关的每个实例私有的脚本数据,必要时会创建 | | [GetOrCreatePublicScriptScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetOrCreatePublicScriptScope) | handle GetOrCreatePublicScriptScope() | 恢复实体相关的公用脚本数据,必要时会创建 | | [GetPrivateScriptScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetPrivateScriptScope) | handle GetPrivateScriptScope() | 恢复实体相关的每个实例私有的脚本数据 | | [GetPublicScriptScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.GetPublicScriptScope) | handle GetPublicScriptScope() | 恢复实体相关的公用脚本数据 | | [RedirectOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.RedirectOutput) | void RedirectOutput(string a, string b, handle c) | 添加一个I/O链接,这可以当特定输出触发时,在这个实体调用方法 | | [RemoveSelf](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.RemoveSelf) | void RemoveSelf() | 删除这个实体 | | [SetIntAttr](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEntityInstance.SetIntAttr) | void SetIntAttr(string a, int b) | 设置整型属性 | |
|  |
| CDOTABaseAbility |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CastAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.CastAbility) | void CastAbility() | No Description Set | | [ContinueCasting](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ContinueCasting) | bool ContinueCasting() | No Description Set | | [CreateVisibilityNode](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.CreateVisibilityNode) | void CreateVisibilityNode(Vector a, float b, float c) | No Description Set | | [DecrementModifierRefCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.DecrementModifierRefCount) | void DecrementModifierRefCount() | No Description Set | | [EndChannel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.EndChannel) | void EndChannel(bool a) | 参数: bool bInterrupted | | [EndCooldown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.EndCooldown) | void EndCooldown() | 刷新这个技能 | | [GetAbilityDamage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityDamage) | int GetAbilityDamage() | No Description Set | | [GetAbilityDamageType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityDamageType) | int GetAbilityDamageType() | No Description Set | | [GetAbilityIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityIndex) | int GetAbilityIndex() | No Description Set | | [GetAbilityName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityName) | string GetAbilityName() | No Description Set | | [GetAbilityTargetFlags](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityTargetFlags) | int GetAbilityTargetFlags() | No Description Set | | [GetAbilityTargetTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityTargetTeam) | int GetAbilityTargetTeam() | No Description Set | | [GetAbilityTargetType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityTargetType) | int GetAbilityTargetType() | No Description Set | | [GetAbilityType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAbilityType) | int GetAbilityType() | No Description Set | | [GetAnimationIgnoresModelScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAnimationIgnoresModelScale) | bool GetAnimationIgnoresModelScale() | No Description Set | | [GetAssociatedPrimaryAbilities](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAssociatedPrimaryAbilities) | string GetAssociatedPrimaryAbilities() | No Description Set | | [GetAssociatedSecondaryAbilities](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAssociatedSecondaryAbilities) | string GetAssociatedSecondaryAbilities() | No Description Set | | [GetAutoCastState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetAutoCastState) | bool GetAutoCastState() | No Description Set | | [GetBackswingTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetBackswingTime) | float GetBackswingTime() | No Description Set | | [GetBehavior](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetBehavior) | int GetBehavior() | No Description Set | | [GetCaster](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCaster) | handle GetCaster() | No Description Set | | [GetCastPoint](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCastPoint) | float GetCastPoint() | Returns the cast point of this ability. | | [GetCastRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCastRange) | int GetCastRange() | Returns the cast range of this ability. | | [GetChannelledManaCostPerSecond](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetChannelledManaCostPerSecond) | int GetChannelledManaCostPerSecond(int a) | Returns the channeled mana cost per second of this ability if it's a channeling ability. | | [GetChannelStartTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetChannelStartTime) | float GetChannelStartTime() | No Description Set | | [GetChannelTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetChannelTime) | float GetChannelTime() | No Description Set | | [GetCloneSource](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCloneSource) | handle GetCloneSource() | No Description Set | | [GetConceptRecipientType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetConceptRecipientType) | int GetConceptRecipientType() | No Description Set | | [GetCooldown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCooldown) | float GetCooldown(int a) | 获得技能在给出的等级设定上的冷却时间,而不是目前的剩余的冷却时间 | | [GetCooldownTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCooldownTime) | float GetCooldownTime() | No Description Set | | [GetCooldownTimeRemaining](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCooldownTimeRemaining) | float GetCooldownTimeRemaining() | No Description Set | | [GetCursorPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCursorPosition) | Vector GetCursorPosition() | Returns the target point vector from the last cast of this point-targeting ability . | | [GetCursorTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCursorTarget) | handle GetCursorTarget() | Returns the target unit from the last cast of this unit-targeting ability. | | [GetCursorTargetingNothing](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetCursorTargetingNothing) | bool GetCursorTargetingNothing() | No Description Set | | [GetDuration](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetDuration) | float GetDuration() | No Description Set | | [GetGoldCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetGoldCost) | int GetGoldCost(int a) | No Description Set | | [GetGoldCostForUpgrade](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetGoldCostForUpgrade) | int GetGoldCostForUpgrade(int a) | No Description Set | | [GetHeroLevelRequiredToUpgrade](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetHeroLevelRequiredToUpgrade) | int GetHeroLevelRequiredToUpgrade() | No Description Set | | [GetIntrinsicModifierName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetIntrinsicModifierName) | string GetIntrinsicModifierName() | No Description Set | | [GetLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetLevel) | int GetLevel() | 获得技能目前的等级 | | [GetLevelSpecialValueFor](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetLevelSpecialValueFor) | table GetLevelSpecialValueFor(string a, int b) | No Description Set | | [GetManaCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetManaCost) | int GetManaCost(int a) | No Description Set | | [GetMaxLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetMaxLevel) | int GetMaxLevel() | No Description Set | | [GetModifierValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetModifierValue) | float GetModifierValue() | No Description Set | | [GetModifierValueBonus](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetModifierValueBonus) | float GetModifierValueBonus() | No Description Set | | [GetPlaybackRateOverride](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetPlaybackRateOverride) | float GetPlaybackRateOverride() | No Description Set | | [GetSharedCooldownName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetSharedCooldownName) | string GetSharedCooldownName() | No Description Set | | [GetSpecialValueFor](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetSpecialValueFor) | table GetSpecialValueFor(string a) | No Description Set | | [GetStolenActivityModifier](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetStolenActivityModifier) | string GetStolenActivityModifier() | No Description Set | | [GetToggleState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.GetToggleState) | bool GetToggleState() | No Description Set | | [HeroXPChange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.HeroXPChange) | bool HeroXPChange(float a) | No Description Set | | [IncrementModifierRefCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IncrementModifierRefCount) | void IncrementModifierRefCount() | No Description Set | | [IsActivated](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsActivated) | bool IsActivated() | No Description Set | | [IsAttributeBonus](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsAttributeBonus) | bool IsAttributeBonus() | No Description Set | | [IsChanneling](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsChanneling) | bool IsChanneling() | 返回目前技能是否在施法状态 | | [IsCooldownReady](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsCooldownReady) | bool IsCooldownReady() | No Description Set | | [IsCosmetic](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsCosmetic) | bool IsCosmetic() | No Description Set | | [IsFullyCastable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsFullyCastable) | bool IsFullyCastable() | 返回技能能否被释放 | | [IsHidden](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsHidden) | bool IsHidden() | No Description Set | | [IsHiddenWhenStolen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsHiddenWhenStolen) | bool IsHiddenWhenStolen() | No Description Set | | [IsInAbilityPhase](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsInAbilityPhase) | bool IsInAbilityPhase() | 返回技能目前是否在被释放 | | [IsItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsItem) | bool IsItem() | Returns whether this is an ability attached to an item. | | [IsOwnersGoldEnough](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsOwnersGoldEnough) | bool IsOwnersGoldEnough(int a) | No Description Set | | [IsOwnersGoldEnoughForUpgrade](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsOwnersGoldEnoughForUpgrade) | bool IsOwnersGoldEnoughForUpgrade() | No Description Set | | [IsOwnersManaEnough](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsOwnersManaEnough) | bool IsOwnersManaEnough() | No Description Set | | [IsPassive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsPassive) | bool IsPassive() | Returns whether this is a passive ability. | | [IsSharedWithTeammates](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsSharedWithTeammates) | bool IsSharedWithTeammates() | No Description Set | | [IsStealable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsStealable) | bool IsStealable() | Returns whether this ability can be stealed (by Rubick's ultimate). | | [IsStolen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsStolen) | bool IsStolen() | Returns whether this ability has been stolen. | | [IsToggle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsToggle) | bool IsToggle() | No Description Set | | [IsTrained](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.IsTrained) | bool IsTrained() | No Description Set | | [MarkAbilityButtonDirty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.MarkAbilityButtonDirty) | void MarkAbilityButtonDirty() | 标记这个技能图标表示需要刷新 | | [NumModifiersUsingAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.NumModifiersUsingAbility) | int NumModifiersUsingAbility() | No Description Set | | [OnAbilityPhaseInterrupted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnAbilityPhaseInterrupted) | void OnAbilityPhaseInterrupted() | No Description Set | | [OnAbilityPhaseStart](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnAbilityPhaseStart) | bool OnAbilityPhaseStart() | No Description Set | | [OnAbilityPinged](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnAbilityPinged) | void OnAbilityPinged() | No Description Set | | [OnChannelFinish](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnChannelFinish) | void OnChannelFinish(bool a) | No Description Set | | [OnChannelThink](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnChannelThink) | void OnChannelThink(float a) | No Description Set | | [OnHeroCalculateStatBonus](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnHeroCalculateStatBonus) | void OnHeroCalculateStatBonus() | No Description Set | | [OnHeroLevelUp](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnHeroLevelUp) | void OnHeroLevelUp() | No Description Set | | [OnInventoryContentsChanged](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnInventoryContentsChanged) | void OnInventoryContentsChanged() | No Description Set | | [OnOwnerDied](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnOwnerDied) | void OnOwnerDied() | No Description Set | | [OnOwnerSpawned](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnOwnerSpawned) | void OnOwnerSpawned() | No Description Set | | [OnSpellStart](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnSpellStart) | void OnSpellStart() | No Description Set | | [OnToggle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnToggle) | void OnToggle() | No Description Set | | [OnUpgrade](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.OnUpgrade) | void OnUpgrade() | No Description Set | | [PayGoldCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.PayGoldCost) | void PayGoldCost() | No Description Set | | [PayGoldCostForUpgrade](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.PayGoldCostForUpgrade) | void PayGoldCostForUpgrade() | No Description Set | | [PayManaCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.PayManaCost) | void PayManaCost() | No Description Set | | [PlaysDefaultAnimWhenStolen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.PlaysDefaultAnimWhenStolen) | bool PlaysDefaultAnimWhenStolen() | No Description Set | | [ProcsMagicStick](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ProcsMagicStick) | bool ProcsMagicStick() | Returns whether this ability procs magic stick. | | [RefCountsModifiers](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.RefCountsModifiers) | bool RefCountsModifiers() | No Description Set | | [RefundManaCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.RefundManaCost) | void RefundManaCost() | No Description Set | | [ResetToggleOnRespawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ResetToggleOnRespawn) | bool ResetToggleOnRespawn() | No Description Set | | [SetAbilityIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetAbilityIndex) | void SetAbilityIndex(int a) | No Description Set | | [SetActivated](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetActivated) | void SetActivated(bool a) | No Description Set | | [SetChanneling](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetChanneling) | void SetChanneling(bool a) | No Description Set | | [SetHidden](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetHidden) | void SetHidden(bool a) | No Description Set | | [SetInAbilityPhase](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetInAbilityPhase) | void SetInAbilityPhase(bool a) | No Description Set | | [SetLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetLevel) | void SetLevel(int a) | 设置这个技能的等级 | | [SetOverrideCastPoint](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetOverrideCastPoint) | void SetOverrideCastPoint(float a) | No Description Set | | [SetRefCountsModifiers](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetRefCountsModifiers) | void SetRefCountsModifiers(bool a) | No Description Set | | [SetStolen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SetStolen) | void SetStolen(bool a) | No Description Set | | [ShouldUseResources](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ShouldUseResources) | bool ShouldUseResources() | No Description Set | | [SpeakAbilityConcept](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SpeakAbilityConcept) | void SpeakAbilityConcept(int a) | No Description Set | | [SpeakTrigger](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.SpeakTrigger) | bool SpeakTrigger() | No Description Set | | [StartCooldown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.StartCooldown) | void StartCooldown(float a) | 参数:fl冷却时间? | | [ToggleAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ToggleAbility) | void ToggleAbility() | No Description Set | | [ToggleAutoCast](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.ToggleAutoCast) | void ToggleAutoCast() | No Description Set | | [UpgradeAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.UpgradeAbility) | void UpgradeAbility() | No Description Set | | [UseResources](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseAbility.UseResources) | void UseResources(bool a, bool b, bool c) | No Description Set | |
| CDOTA\_Ability\_Animation\_Attack |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SetPlaybackRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Ability_Animation_Attack.SetPlaybackRate) | void SetPlaybackRate(float a) | 重写播放速率 | |
|  |
| CDOTA\_Ability\_Animation\_TailSpin |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SetPlaybackRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Ability_Animation_TailSpin.SetPlaybackRate) | void SetPlaybackRate(float a) | Override playbackrate | |
|  |
| CDOTA\_Ability\_Nian\_Leap |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SetPlaybackRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Ability_Nian_Leap.SetPlaybackRate) | void SetPlaybackRate(float a) | Override playbackrate | |
|  |
| CDOTA\_Ability\_Nian\_Dive |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SetPlaybackRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Ability_Nian_Dive.SetPlaybackRate) | void SetPlaybackRate(float a) | Override playbackrate | |
|  |
| CDOTA\_Ability\_Nian\_Roar |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetCastCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Ability_Nian_Roar.GetCastCount) | int GetCastCount() | 年兽使用了咆哮的次数 | |
|  |
| CDOTA\_Item |
| **extends** [**CDOTABaseAbility**](#CDOTABaseAbility) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetContainer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetContainer) | handle GetContainer() | 获得物品所有者 | | [GetCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetCost) | int GetCost() | No Description Set | | [GetCurrentCharges](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetCurrentCharges) | int GetCurrentCharges() | 获得物品的目前充能 | | [GetInitialCharges](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetInitialCharges) | int GetInitialCharges() | 获得物品的初始充能 | | [GetPurchaser](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetPurchaser) | handle GetPurchaser() | 获得物品的购买者 | | [GetPurchaseTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetPurchaseTime) | float GetPurchaseTime() | 获得物品的购买时间 | | [GetShareability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.GetShareability) | int GetShareability() | No Description Set | | [IsPermanent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.IsPermanent) | bool IsPermanent() | 判断物品是否非消耗品 | | [LaunchLoot](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.LaunchLoot) | void LaunchLoot(bool useOnContact, float height, float speed, Vector destination) | For items lying on the ground. Fires the item up into the air so that it lands at the specified position. | | [SetCurrentCharges](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.SetCurrentCharges) | void SetCurrentCharges(int a) | 设置物品的充能数 | | [SetPurchaser](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.SetPurchaser) | void SetPurchaser(handle a) | 设置物品的购买者 | | [SetPurchaseTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.SetPurchaseTime) | void SetPurchaseTime(float a) | 设置物品的购买时间 | | [SetStacksWithOtherOwners](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.SetStacksWithOtherOwners) | void SetStacksWithOtherOwners(bool a) | No Description Set | | [StacksWithOtherOwners](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.StacksWithOtherOwners) | bool StacksWithOtherOwners() | No Description Set | | [Think](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item.Think) | void Think() | 判断此物品 | |
|  |
| CDOTA\_Item\_Physical |
| **extends** [**CBaseAnimating**](#CBaseAnimating) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetContainedItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item_Physical.GetContainedItem) | handle GetContainedItem() | 返回禁用的物品? | | [GetCreationTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item_Physical.GetCreationTime) | float GetCreationTime() | 返回物品第一次出现在地图上的时间 | | [SetContainedItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item_Physical.SetContainedItem) | void SetContainedItem(handle a) | 设置禁用的物品? | |
|  |
| CDOTA\_Item\_DataDriven |
| **extends** [**CDOTA\_Item**](#CDOTA_Item) |
| **'** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ApplyDataDrivenModifier](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Item_DataDriven.ApplyDataDrivenModifier) | void ApplyDataDrivenModifier(handle source, handle target, string modifier\_name, handle modifierArgs) |  | |
|  |
| CDOTA\_Unit\_Nian |
| **extends** [**CDOTA\_BaseNPC\_Creature**](#CDOTA_BaseNPC_Creature) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetHorn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Unit_Nian.GetHorn) | handle GetHorn() | 获取年兽角 | | [GetTail](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Unit_Nian.GetTail) | handle GetTail() | 获取年兽尾巴 | | [IsHornAlive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Unit_Nian.IsHornAlive) | bool IsHornAlive() | 年兽的角巴是否损坏? | | [IsTailAlive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_Unit_Nian.IsTailAlive) | bool IsTailAlive() | 年兽的尾巴是否损坏? | |
|  |
| CBasePlayer |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [IsNoclipping](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBasePlayer.IsNoclipping) | bool IsNoclipping() | 玩家处于穿墙模式（noclip mode）下返回true | |
|  |
| CDOTAPlayer |
| **extends** [**CBaseAnimating**](#CBaseAnimating) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetAssignedHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.GetAssignedHero) | handle GetAssignedHero() | 获取该玩家的英雄 | | [GetControlledRPGUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.GetControlledRPGUnit) | handle GetControlledRPGUnit() | 获取该玩家控制的RPG单位 | | [GetPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.GetPlayerID) | int GetPlayerID() | 获取该玩家的官方玩家ID（PlayerID）玩家不在队伍中时返回-1 | | [MakeRandomHeroSelection](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.MakeRandomHeroSelection) | void MakeRandomHeroSelection() | 随机该玩家的英雄 | | [SetKillCamUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.SetKillCamUnit) | void SetKillCamUnit(handle a) | 设置英雄的击杀镜头单位（kill cam unit） | | [SetMusicStatus](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAPlayer.SetMusicStatus) | void SetMusicStatus(int nMusicStatus, float flIntensity) | 设置玩家的音乐状态，注意这只有在dota\_music\_battle\_enable为off时才会有效 | |
|  |
| CDOTA\_PlayerResource |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
|  |
| ***Global accessor variable:* PlayerResource** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddAegisPickup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.AddAegisPickup) | void AddAegisPickup(int a) | No Description Set | | [AddClaimedFarm](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.AddClaimedFarm) | void AddClaimedFarm(int a, float b) | No Description Set | | [AddGoldSpentOnSupport](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.AddGoldSpentOnSupport) | void AddGoldSpentOnSupport(int a, int b) | No Description Set | | [AddRunePickup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.AddRunePickup) | void AddRunePickup(int a) | No Description Set | | [AreUnitsSharedWithPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.AreUnitsSharedWithPlayerID) | bool AreUnitsSharedWithPlayerID(int a, int b) | No Description Set | | [ClearKillsMatrix](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ClearKillsMatrix) | void ClearKillsMatrix(int a) | No Description Set | | [ClearLastHitMultikill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ClearLastHitMultikill) | void ClearLastHitMultikill(int a) | No Description Set | | [ClearLastHitStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ClearLastHitStreak) | void ClearLastHitStreak(int a) | No Description Set | | [ClearRawPlayerDamageMatrix](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ClearRawPlayerDamageMatrix) | void ClearRawPlayerDamageMatrix(int a) | No Description Set | | [ClearStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ClearStreak) | void ClearStreak(int a) | No Description Set | | [GetAegisPickups](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetAegisPickups) | int GetAegisPickups(int a) | No Description Set | | [GetAssists](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetAssists) | int GetAssists(int a) | No Description Set | | [GetBroadcasterChannel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetBroadcasterChannel) | <> GetBroadcasterChannel(int a) | No Description Set | | [GetBroadcasterChannelSlot](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetBroadcasterChannelSlot) | <> GetBroadcasterChannelSlot(int a) | No Description Set | | [GetClaimedDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetClaimedDenies) | int GetClaimedDenies(int a) | No Description Set | | [GetClaimedFarm](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetClaimedFarm) | float GetClaimedFarm(int a) | No Description Set | | [GetClaimedMisses](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetClaimedMisses) | int GetClaimedMisses(int a) | No Description Set | | [GetConnectionState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetConnectionState) | <> GetConnectionState(int a) | No Description Set | | [GetCreepDamageTaken](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetCreepDamageTaken) | int GetCreepDamageTaken(int a) | No Description Set | | [GetCustomBuybackCooldown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetCustomBuybackCooldown) | float GetCustomBuybackCooldown(int a) | No Description Set | | [GetCustomBuybackCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetCustomBuybackCost) | int GetCustomBuybackCost(int a) | No Description Set | | [GetCustomTeamAssignment](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetCustomTeamAssignment) | int GetCustomTeamAssignment(int playerIndex) | 返回玩家要加入的队伍编号 | | [GetDamageDoneToHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetDamageDoneToHero) | int GetDamageDoneToHero(int a, int b) | No Description Set | | [GetDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetDeaths) | int GetDeaths(int a) | No Description Set | | [GetDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetDenies) | int GetDenies(int a) | No Description Set | | [GetEventPointsForPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetEventPointsForPlayerID) | int GetEventPointsForPlayerID(int a) | No Description Set | | [GetEventPremiumPointsGranted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetEventPremiumPointsGranted) | int GetEventPremiumPointsGranted(int a) | No Description Set | | [GetEventRankGranted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetEventRankGranted) | int GetEventRankGranted(int a) | No Description Set | | [GetGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGold) | int GetGold(int playerID) | 返回指定玩家现有金钱量 | | [GetGoldBagsCollected](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldBagsCollected) | int GetGoldBagsCollected(int a) | No Description Set | | [GetGoldLostToDeath](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldLostToDeath) | int GetGoldLostToDeath(int a) | No Description Set | | [GetGoldPerMin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldPerMin) | float GetGoldPerMin(int a) | No Description Set | | [GetGoldSpentOnBuybacks](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldSpentOnBuybacks) | int GetGoldSpentOnBuybacks(int a) | No Description Set | | [GetGoldSpentOnConsumables](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldSpentOnConsumables) | int GetGoldSpentOnConsumables(int a) | No Description Set | | [GetGoldSpentOnItems](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldSpentOnItems) | int GetGoldSpentOnItems(int a) | No Description Set | | [GetGoldSpentOnSupport](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetGoldSpentOnSupport) | int GetGoldSpentOnSupport(int a) | No Description Set | | [GetHealing](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetHealing) | float GetHealing(int a) | No Description Set | | [GetHeroDamageTaken](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetHeroDamageTaken) | int GetHeroDamageTaken(int a) | No Description Set | | [GetKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetKills) | int GetKills(int playerID) | 返回指定玩家击杀次数 | | [GetKillsDoneToHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetKillsDoneToHero) | int GetKillsDoneToHero(int attackerPlayerID, int victimPlayerID) | 返回攻击者(第一个参数)杀死指定受害者(第二个参数)的次数 | | [GetLastHitMultikill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetLastHitMultikill) | int GetLastHitMultikill(int a) | No Description Set | | [GetLastHits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetLastHits) | int GetLastHits(int playerID) | No Description Set | | [GetLastHitStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetLastHitStreak) | int GetLastHitStreak(int a) | No Description Set | | [GetLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetLevel) | int GetLevel(int playerID) | No Description Set | | [GetMisses](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetMisses) | int GetMisses(int a) | No Description Set | | [GetNearbyCreepDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNearbyCreepDeaths) | int GetNearbyCreepDeaths(int a) | No Description Set | | [GetNthCourierForTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNthCourierForTeam) | handle GetNthCourierForTeam(int a, int b) | No Description Set | | [GetNthPlayerIDOnTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNthPlayerIDOnTeam) | int GetNthPlayerIDOnTeam(int a, int b) | No Description Set | | [GetNumConsumablesPurchased](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNumConsumablesPurchased) | int GetNumConsumablesPurchased(int a) | No Description Set | | [GetNumCouriersForTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNumCouriersForTeam) | int GetNumCouriersForTeam(int a) | No Description Set | | [GetNumItemsPurchased](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetNumItemsPurchased) | int GetNumItemsPurchased(int a) | No Description Set | | [GetPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetPlayer) | handle GetPlayer(int playerID) | No Description Set | | [GetPlayerLoadedCompletely](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetPlayerLoadedCompletely) | bool GetPlayerLoadedCompletely(int a) | No Description Set | | [GetPlayerName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetPlayerName) | string GetPlayerName(int a) | No Description Set | | [GetPlayerReservedState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetPlayerReservedState) | bool GetPlayerReservedState(int a) | No Description Set | | [GetRawPlayerDamage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetRawPlayerDamage) | int GetRawPlayerDamage(int a) | No Description Set | | [GetReliableGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetReliableGold) | int GetReliableGold(int playerID) | 返回指定玩家现有可靠金钱量 | | [GetRespawnSeconds](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetRespawnSeconds) | int GetRespawnSeconds(int a) | No Description Set | | [GetRoshanKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetRoshanKills) | int GetRoshanKills(int a) | No Description Set | | [GetRunePickups](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetRunePickups) | int GetRunePickups(int a) | No Description Set | | [GetSelectedHeroEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetSelectedHeroEntity) | handle GetSelectedHeroEntity(int a) | No Description Set | | [GetSelectedHeroID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetSelectedHeroID) | int GetSelectedHeroID(int a) | No Description Set | | [GetSelectedHeroName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetSelectedHeroName) | string GetSelectedHeroName(int a) | No Description Set | | [GetSteamAccountID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetSteamAccountID) | <> GetSteamAccountID(int a) | No Description Set | | [GetStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetStreak) | int GetStreak(int a) | No Description Set | | [GetStuns](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetStuns) | float GetStuns(int a) | No Description Set | | [GetTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTeam) | int GetTeam(int a) | No Description Set | | [GetTeamKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTeamKills) | int GetTeamKills(int a) | No Description Set | | [GetTimeOfLastConsumablePurchase](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTimeOfLastConsumablePurchase) | float GetTimeOfLastConsumablePurchase(int a) | No Description Set | | [GetTimeOfLastDeath](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTimeOfLastDeath) | float GetTimeOfLastDeath(int a) | No Description Set | | [GetTimeOfLastItemPurchase](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTimeOfLastItemPurchase) | float GetTimeOfLastItemPurchase(int a) | No Description Set | | [GetTotalEarnedGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTotalEarnedGold) | int GetTotalEarnedGold(int a) | No Description Set | | [GetTotalEarnedXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTotalEarnedXP) | int GetTotalEarnedXP(int a) | No Description Set | | [GetTotalGoldSpent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTotalGoldSpent) | int GetTotalGoldSpent(int a) | No Description Set | | [GetTowerDamageTaken](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTowerDamageTaken) | int GetTowerDamageTaken(int a) | No Description Set | | [GetTowerKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetTowerKills) | int GetTowerKills(int a) | No Description Set | | [GetUnitShareMaskForPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetUnitShareMaskForPlayer) | int GetUnitShareMaskForPlayer(int a, int b) | No Description Set | | [GetUnreliableGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetUnreliableGold) | int GetUnreliableGold(int playerID) | 返回指定玩家现有不可靠金钱量 | | [GetXPPerMin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.GetXPPerMin) | float GetXPPerMin(int a) | No Description Set | | [HasRandomed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.HasRandomed) | bool HasRandomed(int a) | No Description Set | | [HasRepicked](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.HasRepicked) | bool HasRepicked(int playerID) | No Description Set | | [HasSelectedHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.HasSelectedHero) | bool HasSelectedHero(int a) | No Description Set | | [HaveAllPlayersJoined](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.HaveAllPlayersJoined) | bool HaveAllPlayersJoined() | No Description Set | | [HeroLevelUp](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.HeroLevelUp) | void HeroLevelUp(int a) | No Description Set | | [IncrementAssists](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementAssists) | void IncrementAssists(int playerID) | No Description Set | | [IncrementClaimedDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementClaimedDenies) | void IncrementClaimedDenies(int a) | No Description Set | | [IncrementClaimedMisses](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementClaimedMisses) | void IncrementClaimedMisses(int a) | No Description Set | | [IncrementDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementDeaths) | void IncrementDeaths(int playerID) | No Description Set | | [IncrementDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementDenies) | void IncrementDenies(int a) | No Description Set | | [IncrementGoldBagsCollected](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementGoldBagsCollected) | void IncrementGoldBagsCollected(int a) | No Description Set | | [IncrementKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementKills) | void IncrementKills(int playerID, int kills) | No Description Set | | [IncrementLastHitMultikill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementLastHitMultikill) | void IncrementLastHitMultikill(int a) | No Description Set | | [IncrementLastHits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementLastHits) | void IncrementLastHits(int a) | No Description Set | | [IncrementLastHitStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementLastHitStreak) | void IncrementLastHitStreak(int a) | No Description Set | | [IncrementMisses](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementMisses) | void IncrementMisses(int a) | No Description Set | | [IncrementNearbyCreepDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementNearbyCreepDeaths) | void IncrementNearbyCreepDeaths(int a) | No Description Set | | [IncrementStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementStreak) | void IncrementStreak(int a) | No Description Set | | [IncrementTotalEarnedXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IncrementTotalEarnedXP) | void IncrementTotalEarnedXP(int a, int b) | No Description Set | | [IsBroadcaster](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsBroadcaster) | bool IsBroadcaster(int a) | No Description Set | | [IsDisableHelpSetForPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsDisableHelpSetForPlayerID) | bool IsDisableHelpSetForPlayerID(int a, int b) | No Description Set | | [IsFakeClient](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsFakeClient) | bool IsFakeClient(int a) | No Description Set | | [IsHeroSelected](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsHeroSelected) | bool IsHeroSelected(string a) | No Description Set | | [IsHeroSharedWithPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsHeroSharedWithPlayerID) | bool IsHeroSharedWithPlayerID(int a, int b) | No Description Set | | [IsValidPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsValidPlayer) | bool IsValidPlayer(int playerID) | No Description Set | | [IsValidPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsValidPlayerID) | bool IsValidPlayerID(int playerID) | No Description Set | | [IsValidTeamPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsValidTeamPlayer) | bool IsValidTeamPlayer(int playerID) | No Description Set | | [IsValidTeamPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.IsValidTeamPlayerID) | bool IsValidTeamPlayerID(int playerID) | No Description Set | | [ModifyGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ModifyGold) | int ModifyGold(int playerID, int goldAmmt, bool reliable, int d) | No Description Set | | [ReplaceHeroWith](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ReplaceHeroWith) | handle ReplaceHeroWith(int a, string b, int c, int d) | (玩家ID,英雄职业名,金钱,经验值) - 将玩家的英雄替换成指定的英雄、金钱和经验值 | | [ResetBuybackCostTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ResetBuybackCostTime) | void ResetBuybackCostTime(int a) | No Description Set | | [ResetTotalEarnedGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.ResetTotalEarnedGold) | void ResetTotalEarnedGold(int a) | No Description Set | | [SetBuybackCooldownTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetBuybackCooldownTime) | void SetBuybackCooldownTime(int a, float b) | No Description Set | | [SetBuybackGoldLimitTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetBuybackGoldLimitTime) | void SetBuybackGoldLimitTime(int a, float b) | No Description Set | | [SetCameraTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetCameraTarget) | void SetCameraTarget(int a, handle b) | (玩家ID,实体) - 强制玩家镜头跟随目标实体 | | [SetCustomBuybackCooldown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetCustomBuybackCooldown) | void SetCustomBuybackCooldown(int a, float b) | 设置玩家买活冷却时间 | | [SetCustomBuybackCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetCustomBuybackCost) | void SetCustomBuybackCost(int a, int b) | 设置玩家买活花费 | | [SetCustomTeamAssignment](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetCustomTeamAssignment) | void SetCustomTeamAssignment(int playerIndex, int teamToJoin) | 设置玩家要加入的队伍 | | [SetGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetGold) | void SetGold(int playerID, int amount, bool reliableGold) | 设置玩家的可靠/不可靠金钱数 | | [SetHasRandomed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetHasRandomed) | void SetHasRandomed(int playerID) | No Description Set | | [SetHasRepicked](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetHasRepicked) | void SetHasRepicked(int playerID) | No Description Set | | [SetLastBuybackTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetLastBuybackTime) | void SetLastBuybackTime(int a, int b) | No Description Set | | [SetPlayerReservedState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetPlayerReservedState) | void SetPlayerReservedState(int a, bool b) | No Description Set | | [SetUnitShareMaskForPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SetUnitShareMaskForPlayer) | void SetUnitShareMaskForPlayer(int a, int b, int c, bool d) | No Description Set | | [SpendGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.SpendGold) | void SpendGold(int playerID, int amount, int reason) | No Description Set | | [UpdateTeamSlot](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.UpdateTeamSlot) | void UpdateTeamSlot(int a, int b) | No Description Set | | [WhoSelectedHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_PlayerResource.WhoSelectedHero) | int WhoSelectedHero(string a) | No Description Set | |
|  |
| CDOTA\_BaseNPC |
| **extends** [**CBaseFlex**](#CBaseFlex) |
| **'** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AddAbility) | void AddAbility(string a) | 以技能名称为该单位新增技能。 | | [AddItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AddItem) | void AddItem(handle a) | 增加物品到该单位的库存 。 | | [AddNewModifier](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AddNewModifier) | void AddNewModifier(handle caster, handle optionalSourceAbility, string modifierName, handle modifierData) | 为该单位增加修改器。 | | [AddNoDraw](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AddNoDraw) | void AddNoDraw() | Adds the no draw flag. | | [AlertNearbyUnits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AlertNearbyUnits) | void AlertNearbyUnits(handle a, handle b) | No Description Set | | [AngerNearbyUnits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AngerNearbyUnits) | void AngerNearbyUnits() | No Description Set | | [AttackNoEarlierThan](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AttackNoEarlierThan) | void AttackNoEarlierThan(float a) | No Description Set | | [AttackReady](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.AttackReady) | bool AttackReady() | No Description Set | | [BoundingRadius2D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.BoundingRadius2D) | float BoundingRadius2D() | No Description Set | | [CanEntityBeSeenByMyTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CanEntityBeSeenByMyTeam) | bool CanEntityBeSeenByMyTeam(handle a) | Check FoW to see if an entity is visible. | | [CastAbilityImmediately](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CastAbilityImmediately) | void CastAbilityImmediately(handle ability, int playerIndex) | 技能立即施法。 | | [CastAbilityNoTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CastAbilityNoTarget) | void CastAbilityNoTarget(handle ability, int playerIndex) | 无目标的施放技能 ( hAbility, iPlayerIndex ) 。 | | [CastAbilityOnPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CastAbilityOnPosition) | void CastAbilityOnPosition(Vector a, handle b, int c) | 对一地点施放技能。 | | [CastAbilityOnTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CastAbilityOnTarget) | void CastAbilityOnTarget(handle target, handle ability, int playerIndex) | 对一实体施放技能。 | | [CastAbilityToggle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.CastAbilityToggle) | void CastAbilityToggle(handle a, int b) | 切换技能 ( hAbility, iPlayerIndex ) 。 | | [DisassembleItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.DisassembleItem) | void DisassembleItem(handle a) | 拆解该单位库存中过时的物品。 | | [DropItemAtPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.DropItemAtPosition) | void DropItemAtPosition(Vector a, handle b) | 在指定地点掉落物品。 | | [DropItemAtPositionImmediate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.DropItemAtPositionImmediate) | void DropItemAtPositionImmediate(handle a, Vector b) | 立即在指定地点掉落物品。 | | [EjectItemFromStash](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.EjectItemFromStash) | void EjectItemFromStash(handle a) | 自库存栏丢落指定的物品。 | | [FindAbilityByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.FindAbilityByName) | handle FindAbilityByName(string a) | 以名称寻找技能。 | | [ForceKill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.ForceKill) | void ForceKill(bool a) | 强制击杀该单位。 | | [GetAbilityByIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAbilityByIndex) | handle GetAbilityByIndex(int a) | 以下标值检索该单位的技能。 | | [GetAbilityCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAbilityCount) | int GetAbilityCount() | 获取拥有的技能数量。 | | [GetAcquisitionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAcquisitionRange) | float GetAcquisitionRange() | 获取单位的自动收集距离。 | | [GetAdditionalBattleMusicWeight](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAdditionalBattleMusicWeight) | float GetAdditionalBattleMusicWeight() | 获取当与该单位战斗时将会计算的战斗音乐权重。 | | [GetAttackAnimationPoint](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackAnimationPoint) | float GetAttackAnimationPoint() | No Description Set | | [GetAttackCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackCapability) | int GetAttackCapability() | Gets the Attack Capabilities of the unit | | [GetAttackDamage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackDamage) | int GetAttackDamage() | 获取该单位伤害最大值与最小值之间的随机整数。 | | [GetAttackRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackRange) | float GetAttackRange() | 获取单位当前的攻击距离。 | | [GetAttackRangeBuffer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackRangeBuffer) | float GetAttackRangeBuffer() | 获取状态增加的攻击距离。 | | [GetAttackSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackSpeed) | float GetAttackSpeed() | 获取该单位的攻速。 | | [GetAttacksPerSecond](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttacksPerSecond) | float GetAttacksPerSecond() | 获取APS值(每秒能够打几下)。 | | [GetAttackTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAttackTarget) | handle GetAttackTarget() | 返回该单位的攻击目标。 | | [GetAverageTrueAttackDamage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetAverageTrueAttackDamage) | int GetAverageTrueAttackDamage() | 获取该单位最大伤害值与最小伤害值的平均数。 | | [GetBaseAttackRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseAttackRange) | int GetBaseAttackRange() | 获取单位的基础攻击距离(修改器不列入计算)。 | | [GetBaseAttackTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseAttackTime) | float GetBaseAttackTime() | No Description Set | | [GetBaseDamageMax](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseDamageMax) | int GetBaseDamageMax() | 获取最大伤害值。 | | [GetBaseDamageMin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseDamageMin) | int GetBaseDamageMin() | 获取最小伤害值。 | | [GetBaseDayTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseDayTimeVisionRange) | int GetBaseDayTimeVisionRange() | 获取单位的基础日间视野距离(修改器不列入计算)。 | | [GetBaseHealthRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseHealthRegen) | float GetBaseHealthRegen() | 获取单位的基础血量恢复值。 | | [GetBaseMagicalResistanceValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseMagicalResistanceValue) | float GetBaseMagicalResistanceValue() | 获取单位的基础魔法抗性值。 | | [GetBaseMaxHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseMaxHealth) | float GetBaseMaxHealth() | 获取基础血量值。 | | [GetBaseMoveSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseMoveSpeed) | float GetBaseMoveSpeed() | 获取基础跑速。 | | [GetBaseNightTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetBaseNightTimeVisionRange) | int GetBaseNightTimeVisionRange() | 获取单位基础的夜间视野距离(修改器不列入计算)。 | | [GetCastPoint](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCastPoint) | float GetCastPoint(bool a) | Parameter: bAttack | | [GetCollisionPadding](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCollisionPadding) | float GetCollisionPadding() | 返回该单位的碰撞体积。 | | [GetConstantBasedManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetConstantBasedManaRegen) | float GetConstantBasedManaRegen() | 获取基础的魔力恢复值常数。 | | [GetCreationTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCreationTime) | float GetCreationTime() | No Description Set | | [GetCurrentActiveAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCurrentActiveAbility) | handle GetCurrentActiveAbility() | 获取该单位正在使用中的法术。 | | [GetCurrentVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCurrentVisionRange) | int GetCurrentVisionRange() | 获取当前的视野距离。 | | [GetCursorCastTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCursorCastTarget) | handle GetCursorCastTarget() | 获取鼠标指定的施法目标。 | | [GetCursorPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCursorPosition) | Vector GetCursorPosition() | 获取鼠标位置。 | | [GetCursorTargetingNothing](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetCursorTargetingNothing) | bool GetCursorTargetingNothing() | No Description Set | | [GetDayTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetDayTimeVisionRange) | int GetDayTimeVisionRange() | 获取该单位的日间视野(将修改器并入计算)。 | | [GetDeathXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetDeathXP) | int GetDeathXP() | 获取该单位的击杀经验值奖赏。 | | [GetForceAttackTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetForceAttackTarget) | handle GetForceAttackTarget() | No Description Set | | [GetGoldBounty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetGoldBounty) | int GetGoldBounty() | 获取该单位的击杀金钱奖赏。 | | [GetHasteFactor](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHasteFactor) | float GetHasteFactor() | No Description Set | | [GetHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHealth) | int GetHealth() | 获取该单位的血量。 | | [GetHealthDeficit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHealthDeficit) | int GetHealthDeficit() | 返回损失的血量 (最大血量 - 当前血量)。 | | [GetHealthPercent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHealthPercent) | int GetHealthPercent() | 获取该单位当前的血量百分比。 | | [GetHealthRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHealthRegen) | float GetHealthRegen() | 获取该单位当前的血量恢复值。 | | [GetHullRadius](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetHullRadius) | float GetHullRadius() | 获取该单位的碰撞体积半径。 | | [GetIdealSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetIdealSpeed) | float GetIdealSpeed() | 获取该单位的跑速(将修改器并入计算)。 | | [GetIncreasedAttackSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetIncreasedAttackSpeed) | float GetIncreasedAttackSpeed() | No Description Set | | [GetInitialGoalEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetInitialGoalEntity) | handle GetInitialGoalEntity() | 返回该单位初始的路径点。 | | [GetItemInSlot](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetItemInSlot) | handle GetItemInSlot(int a) | 以下标值获取物品栏中的物品。 | | [GetLastIdleChangeTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetLastIdleChangeTime) | float GetLastIdleChangeTime() | 获取该单位最后 进入/离开 闲置状态的时间。 | | [GetLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetLevel) | int GetLevel() | 返回该单位当前的等级。 | | [GetMagicalArmorValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMagicalArmorValue) | float GetMagicalArmorValue() | 返回该单位当前的魔法抗性值。 | | [GetMainControllingPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMainControllingPlayer) | int GetMainControllingPlayer() | 返回控制该单位的玩家ID。 | | [GetMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMana) | float GetMana() | 获取该单位的魔力值。 | | [GetManaPercent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetManaPercent) | int GetManaPercent() | 获取该单位的剩余魔力值百分比。 | | [GetManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetManaRegen) | float GetManaRegen() | 获取该单位的魔力恢复值。 | | [GetMaxHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMaxHealth) | int GetMaxHealth() | 获取该单位的最大血量。 | | [GetMaxMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMaxMana) | float GetMaxMana() | 获取该单位的最大魔力。 | | [GetModelRadius](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetModelRadius) | float GetModelRadius() | 获取模型大小半径。 | | [GetModifierCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetModifierCount) | int GetModifierCount() | 返回该单位所拥有的修改器数量。 | | [GetModifierNameByIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetModifierNameByIndex) | string GetModifierNameByIndex(int a) | 以下标值索引修改器，并返回修改器名称字符串。 | | [GetModifierStackCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetModifierStackCount) | void GetModifierStackCount(string modifierName, handle b) | Gets the stack count of a given modifier. | | [GetMoveSpeedModifier](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMoveSpeedModifier) | float GetMoveSpeedModifier(float a) | No Description Set | | [GetMustReachEachGoalEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetMustReachEachGoalEntity) | bool GetMustReachEachGoalEntity() | 该单位是否被要求走访每个路径点? | | [GetNightTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetNightTimeVisionRange) | int GetNightTimeVisionRange() | 返回夜间视野距离(将修改器并入计算)。 | | [GetOpposingTeamNumber](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetOpposingTeamNumber) | int GetOpposingTeamNumber() | No Description Set | | [GetPaddedCollisionRadius](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPaddedCollisionRadius) | float GetPaddedCollisionRadius() | 获取单位碰撞体积半径。 | | [GetPercentageBasedManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPercentageBasedManaRegen) | float GetPercentageBasedManaRegen() | 获取基础魔力恢复值百分比。 | | [GetPhysicalArmorBaseValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPhysicalArmorBaseValue) | float GetPhysicalArmorBaseValue() | 返回基础护甲值。 | | [GetPhysicalArmorValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPhysicalArmorValue) | float GetPhysicalArmorValue() | 返回当前的护甲值。 | | [GetPlayerOwner](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPlayerOwner) | handle GetPlayerOwner() | 获取拥有此单位的玩家实体。 | | [GetPlayerOwnerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetPlayerOwnerID) | int GetPlayerOwnerID() | 获取拥有此单位的玩家ID。 | | [GetProjectileSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetProjectileSpeed) | int GetProjectileSpeed() | No Description Set | | [GetRangeToUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetRangeToUnit) | float GetRangeToUnit(handle a) | No Description Set | | [GetSecondsPerAttack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetSecondsPerAttack) | float GetSecondsPerAttack() | No Description Set | | [GetStatsBasedManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetStatsBasedManaRegen) | float GetStatsBasedManaRegen() | 获取因智慧而增加的魔法恢复值。 | | [GetTeamNumber](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetTeamNumber) | int GetTeamNumber() | 获取该单位所属的队伍ID。 | | [GetTotalPurchasedUpgradeGoldCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetTotalPurchasedUpgradeGoldCost) | int GetTotalPurchasedUpgradeGoldCost() | 获取用在升级的金钱。 | | [GetUnitLabel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetUnitLabel) | string GetUnitLabel() | No Description Set | | [GetUnitName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GetUnitName) | string GetUnitName() | 获取单位名称。 | | [GiveMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.GiveMana) | void GiveMana(float a) | 给予此单位指定的魔法值。(Giff me mana!) | | [HasAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasAbility) | bool HasAbility(string a) | 以技能名称搜寻该单位是否拥有此技能。 | | [HasAttackCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasAttackCapability) | bool HasAttackCapability() | No Description Set | | [HasFlyingVision](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasFlyingVision) | bool HasFlyingVision() | No Description Set | | [HasFlyMovementCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasFlyMovementCapability) | bool HasFlyMovementCapability() | No Description Set | | [HasGroundMovementCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasGroundMovementCapability) | bool HasGroundMovementCapability() | No Description Set | | [HasInventory](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasInventory) | bool HasInventory() | No Description Set | | [HasItemInInventory](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasItemInInventory) | bool HasItemInInventory(string a) | 以物品名称搜寻该单位是否拥有此物品。 | | [HasModifier](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasModifier) | bool HasModifier(string a) | 以修改器名称搜寻该单位是否拥有此修改器。 | | [HasMovementCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasMovementCapability) | bool HasMovementCapability() | No Description Set | | [HasScepter](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.HasScepter) | bool HasScepter() | No Description Set | | [Heal](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.Heal) | void Heal(float amount, handle source) | 治疗该单位。 | | [Hold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.Hold) | void Hold() | 命令单位在原地待命。 | | [Interrupt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.Interrupt) | void Interrupt() | No Description Set | | [InterruptChannel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.InterruptChannel) | void InterruptChannel() | No Description Set | | [InterruptMotionControllers](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.InterruptMotionControllers) | void InterruptMotionControllers(bool a) | Parameter boolean determines finding clear space. | | [IsAlive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsAlive) | bool IsAlive() | 此单位是否还活着? | | [IsAncient](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsAncient) | bool IsAncient() | 该生物是否为远古野怪? | | [IsAttackImmune](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsAttackImmune) | bool IsAttackImmune() | No Description Set | | [IsAttacking](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsAttacking) | bool IsAttacking() | No Description Set | | [IsAttackingEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsAttackingEntity) | bool IsAttackingEntity(handle a) | No Description Set | | [IsBlind](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsBlind) | bool IsBlind() | No Description Set | | [IsBlockDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsBlockDisabled) | bool IsBlockDisabled() | No Description Set | | [IsCommandRestricted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsCommandRestricted) | bool IsCommandRestricted() | No Description Set | | [IsControllableByAnyPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsControllableByAnyPlayer) | bool IsControllableByAnyPlayer() | 该单位是否为非Bot玩家所控制? | | [IsCreature](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsCreature) | bool IsCreature() | 此单位是否为怪物类型单位? | | [IsDeniable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsDeniable) | bool IsDeniable() | No Description Set | | [IsDisarmed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsDisarmed) | bool IsDisarmed() | No Description Set | | [IsDominated](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsDominated) | bool IsDominated() | No Description Set | | [IsEvadeDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsEvadeDisabled) | bool IsEvadeDisabled() | No Description Set | | [IsFrozen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsFrozen) | bool IsFrozen() | No Description Set | | [IsHardDisarmed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsHardDisarmed) | bool IsHardDisarmed() | No Description Set | | [IsHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsHero) | bool IsHero() | 此单位是否为英雄或英雄幻象? | | [IsHexed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsHexed) | bool IsHexed() | No Description Set | | [IsIdle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsIdle) | bool IsIdle() | 该生物是否处于闲置状态? | | [IsIllusion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsIllusion) | bool IsIllusion() | 该单位是否为幻象单位? | | [IsInvisible](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsInvisible) | bool IsInvisible() | No Description Set | | [IsInvulnerable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsInvulnerable) | bool IsInvulnerable() | No Description Set | | [IsLowAttackPriority](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsLowAttackPriority) | bool IsLowAttackPriority() | No Description Set | | [IsMagicImmune](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsMagicImmune) | bool IsMagicImmune() | 该单位是否魔法免疫? | | [IsMechanical](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsMechanical) | bool IsMechanical() | 该单位是否为机械单位? | | [IsMovementImpaired](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsMovementImpaired) | bool IsMovementImpaired() | No Description Set | | [IsMuted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsMuted) | bool IsMuted() | No Description Set | | [IsNeutralUnitType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsNeutralUnitType) | bool IsNeutralUnitType() | 该单位是否为野怪? | | [IsNightmared](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsNightmared) | bool IsNightmared() | No Description Set | | [IsOpposingTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsOpposingTeam) | bool IsOpposingTeam(int a) | No Description Set | | [IsOutOfGame](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsOutOfGame) | bool IsOutOfGame() | No Description Set | | [IsOwnedByAnyPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsOwnedByAnyPlayer) | bool IsOwnedByAnyPlayer() | 该单位是否为非Bot之玩家所有? | | [IsPhantom](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsPhantom) | bool IsPhantom() | 该单位是否为幻象单位? | | [IsPhantomBlocker](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsPhantomBlocker) | bool IsPhantomBlocker() | No Description Set | | [IsPhased](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsPhased) | bool IsPhased() | 是否在相位状态? | | [IsPositionInRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsPositionInRange) | bool IsPositionInRange(Vector a, float b) | 是否在某点的范围内 | | [IsRangedAttacker](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsRangedAttacker) | bool IsRangedAttacker() | 该单位攻击类型是否为远程攻击? | | [IsRealHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsRealHero) | bool IsRealHero() | 返回该单位是否为真正的英雄单位，而非小兵或是幻象。 | | [IsRooted](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsRooted) | bool IsRooted() | No Description Set | | [IsSilenced](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsSilenced) | bool IsSilenced() | 该单位是否被沉默? | | [IsSoftDisarmed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsSoftDisarmed) | bool IsSoftDisarmed() | No Description Set | | [IsSpeciallyDeniable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsSpeciallyDeniable) | bool IsSpeciallyDeniable() | No Description Set | | [IsStunned](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsStunned) | bool IsStunned() | 该单位是否被晕眩了? | | [IsSummoned](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsSummoned) | bool IsSummoned() | 此单位是否为召唤物? | | [IsTower](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsTower) | bool IsTower() | 此单位是否为塔? | | [IsUnableToMiss](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsUnableToMiss) | bool IsUnableToMiss() | No Description Set | | [IsUnselectable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.IsUnselectable) | bool IsUnselectable() | No Description Set | | [Kill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.Kill) | void Kill(handle a, handle b) | 杀死一个单位，参数为使用的技能与攻击者。 | | [MakeIllusion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MakeIllusion) | void MakeIllusion() | No Description Set | | [MakePhantomBlocker](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MakePhantomBlocker) | void MakePhantomBlocker() | No Description Set | | [MakeVisibleDueToAttack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MakeVisibleDueToAttack) | void MakeVisibleDueToAttack(int a) | No Description Set | | [MakeVisibleToTeam](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MakeVisibleToTeam) | void MakeVisibleToTeam(int a, float b) | No Description Set | | [ModifyHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.ModifyHealth) | void ModifyHealth(int a, handle b, bool c, int d) | 设置血量为所指定的值 | | [MoveToNPC](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MoveToNPC) | void MoveToNPC(handle a) | 移动至指定的单位。 | | [MoveToNPCToGiveItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MoveToNPCToGiveItem) | void MoveToNPCToGiveItem(handle npc, handle item) | 将身上的物品丢给指定的单位。 | | [MoveToPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MoveToPosition) | void MoveToPosition(Vector a) | 移动到指定的位置。 | | [MoveToPositionAggressive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MoveToPositionAggressive) | void MoveToPositionAggressive(Vector a) | 移动到指定的位置。 | | [MoveToTargetToAttack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.MoveToTargetToAttack) | void MoveToTargetToAttack(handle a) | 移动至指定的单位，并且发动攻击。 | | [NoHealthBar](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NoHealthBar) | bool NoHealthBar() | No Description Set | | [NoTeamMoveTo](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NoTeamMoveTo) | bool NoTeamMoveTo() | No Description Set | | [NoTeamSelect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NoTeamSelect) | bool NoTeamSelect() | No Description Set | | [NotOnMinimap](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NotOnMinimap) | bool NotOnMinimap() | No Description Set | | [NotOnMinimapForEnemies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NotOnMinimapForEnemies) | bool NotOnMinimapForEnemies() | No Description Set | | [NoUnitCollision](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.NoUnitCollision) | bool NoUnitCollision() | No Description Set | | [PassivesDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.PassivesDisabled) | bool PassivesDisabled() | No Description Set | | [PerformAttack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.PerformAttack) | void PerformAttack(handle a, bool b, bool c, bool d, bool e) | 对一单位执行攻击。 | | [PickupDroppedItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.PickupDroppedItem) | void PickupDroppedItem(handle a) | 捡起掉落的物品。 | | [PickupRune](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.PickupRune) | void PickupRune(handle a) | 捡起神符。 | | [ProvidesVision](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.ProvidesVision) | bool ProvidesVision() | No Description Set | | [ReduceMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.ReduceMana) | void ReduceMana(float a) | 移除单位的魔力，不过是用在该单位非自愿失去魔力，而非用在使用技能消耗魔力的情况。 | | [RemoveAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RemoveAbility) | void RemoveAbility(string a) | 以技能名称移除该单位相应的技能。 | | [RemoveItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RemoveItem) | void RemoveItem(handle a) | 将该单位库存中的过期物品移除。 | | [RemoveModifierByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RemoveModifierByName) | void RemoveModifierByName(string a) | 以名称移除修改器。 | | [RemoveModifierByNameAndCaster](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RemoveModifierByNameAndCaster) | void RemoveModifierByNameAndCaster(string a, handle b) | 以施法者与修改器名称移除修改器。 | | [RemoveNoDraw](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RemoveNoDraw) | void RemoveNoDraw() | Remove the no draw flag. | | [RespawnUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.RespawnUnit) | void RespawnUnit() | 复活该单位。 | | [SellItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SellItem) | void SellItem(handle a) | 卖出该单位的物品。 | | [SetAcquisitionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetAcquisitionRange) | void SetAcquisitionRange(float a) | Sets acquisition range | | [SetAdditionalBattleMusicWeight](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetAdditionalBattleMusicWeight) | void SetAdditionalBattleMusicWeight(float a) | 获取当与该单位战斗时将会计算的战斗音乐权重。 | | [SetAttackCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetAttackCapability) | void SetAttackCapability(int a) | No Description Set | | [SetAttacking](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetAttacking) | void SetAttacking(handle a) | No Description Set | | [SetBaseAttackTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseAttackTime) | void SetBaseAttackTime(float a) | No Description Set | | [SetBaseDamageMax](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseDamageMax) | void SetBaseDamageMax(int a) | 设置基础最大攻击力。 | | [SetBaseDamageMin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseDamageMin) | void SetBaseDamageMin(int a) | 设置基础最小攻击力。 | | [SetBaseHealthRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseHealthRegen) | void SetBaseHealthRegen(float a) | No Description Set | | [SetBaseMagicalResistanceValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseMagicalResistanceValue) | void SetBaseMagicalResistanceValue(float a) | 设置基础魔法抗性值。 | | [SetBaseManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseManaRegen) | void SetBaseManaRegen(float a) | 设置基础魔力恢复值。 | | [SetBaseMaxHealth](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseMaxHealth) | void SetBaseMaxHealth(float a) | 设置基础最大血量。 | | [SetBaseMoveSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetBaseMoveSpeed) | void SetBaseMoveSpeed(int a) | 设置基础跑速。 | | [SetControllableByPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetControllableByPlayer) | void SetControllableByPlayer(int a, bool b) | 以玩家ID设置该单位的控制权。 | | [SetCursorCastTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetCursorCastTarget) | void SetCursorCastTarget(handle a) | No Description Set | | [SetCursorPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetCursorPosition) | void SetCursorPosition(Vector a) | No Description Set | | [SetCursorTargetingNothing](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetCursorTargetingNothing) | void SetCursorTargetingNothing(bool a) | No Description Set | | [SetDayTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetDayTimeVisionRange) | void SetDayTimeVisionRange(int a) | 设置基础的视间视野距离。 | | [SetDeathXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetDeathXP) | void SetDeathXP(int a) | 设置该单位的击杀经验值奖赏。 | | [SetForceAttackTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetForceAttackTarget) | void SetForceAttackTarget(handle a) | No Description Set | | [SetHasInventory](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetHasInventory) | void SetHasInventory(bool a) | No Description Set | | [SetHullRadius](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetHullRadius) | void SetHullRadius(float a) | 设置该单位的碰撞半径。 | | [SetIdleAcquire](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetIdleAcquire) | void SetIdleAcquire(bool a) | No Description Set | | [SetInitialGoalEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetInitialGoalEntity) | void SetInitialGoalEntity(handle a) | 设置该单位的初始路径点。 | | [SetMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetMana) | void SetMana(float a) | 设置该单位的魔力值。 | | [SetMaximumGoldBounty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetMaximumGoldBounty) | void SetMaximumGoldBounty(int a) | 设置该单位的最高赏金。 | | [SetMinimumGoldBounty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetMinimumGoldBounty) | void SetMinimumGoldBounty(int a) | 设置该单位的最低赏金。 | | [SetModifierStackCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetModifierStackCount) | void SetModifierStackCount(string modifierName, handle b, int modifierCount) | Sets the stack count of a given modifier. | | [SetMoveCapability](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetMoveCapability) | void SetMoveCapability(int a) | No Description Set | | [SetMustReachEachGoalEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetMustReachEachGoalEntity) | void SetMustReachEachGoalEntity(bool a) | 设置该单位是否要走访每个路径点。 | | [SetNeverMoveToClearSpace](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetNeverMoveToClearSpace) | void SetNeverMoveToClearSpace(bool a) | 如果设置为True，则不会将该单位移动到净空的地方。 | | [SetNightTimeVisionRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetNightTimeVisionRange) | void SetNightTimeVisionRange(int a) | 设置基础夜间视野距离。 | | [SetOriginalModel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetOriginalModel) | void SetOriginalModel(string originalModel) | 设置单位原模型。 | | [SetPhysicalArmorBaseValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetPhysicalArmorBaseValue) | void SetPhysicalArmorBaseValue(float a) | 设置基础护甲值。 | | [SetRangedProjectileName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetRangedProjectileName) | void SetRangedProjectileName(string a) | No Description Set | | [SetStolenScepter](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetStolenScepter) | void SetStolenScepter(bool a) | No Description Set | | [SetUnitName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SetUnitName) | void SetUnitName(string a) | No Description Set | | [ShouldIdleAcquire](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.ShouldIdleAcquire) | bool ShouldIdleAcquire() | No Description Set | | [SpendMana](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SpendMana) | void SpendMana(float a, handle b) | 消耗魔力，可以用在技能消耗或使用物品。 | | [Stop](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.Stop) | void Stop() | 使该单位暂停当前动作。 | | [SwapAbilities](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.SwapAbilities) | void SwapAbilities(string a, string b, bool c, bool d) | 交换两个技能的字段，并设置其是否生效。 | | [TimeUntilNextAttack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.TimeUntilNextAttack) | float TimeUntilNextAttack() | No Description Set | | [TriggerModifierDodge](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.TriggerModifierDodge) | bool TriggerModifierDodge() | No Description Set | | [TriggerSpellAbsorb](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.TriggerSpellAbsorb) | bool TriggerSpellAbsorb(handle a) | 判断传递的技能是否会触发该单位的技能吸收 | | [UnitCanRespawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC.UnitCanRespawn) | bool UnitCanRespawn() | No Description Set | |
|  |
| CDOTA\_BaseNPC\_Hero |
| **extends** [**CDOTA\_BaseNPC**](#CDOTA_BaseNPC) |
| **'** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddExperience](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.AddExperience) | bool AddExperience(float amount, bool applyBotDifficultyScaling) | 给单位增加经验值 | | [Buyback](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.Buyback) | void Buyback() | 花费金钱并买活英雄 | | [CalculateStatBonus](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.CalculateStatBonus) | void CalculateStatBonus() | 英雄返回属性后，重新计算全部属性 | | [CanEarnGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.CanEarnGold) | bool CanEarnGold() | 返回布尔值，买活返回金钱限制时间是否少于游戏时间（其实就是是否能够正常返回金钱.....） | | [ClearLastHitMultikill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ClearLastHitMultikill) | void ClearLastHitMultikill() | 数值存储在玩家资源(PlayerResource)中 | | [ClearLastHitStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ClearLastHitStreak) | void ClearLastHitStreak() | 数值存储在玩家资源(PlayerResource)中 | | [ClearStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ClearStreak) | void ClearStreak() | 数值存储在玩家资源(PlayerResource)中 | | [GetAbilityPoints](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetAbilityPoints) | int GetAbilityPoints() | 返回未分配的技能点数 | | [GetAgility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetAgility) | float GetAgility() | 返回敏捷值 | | [GetAgilityGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetAgilityGain) | float GetAgilityGain() | 返回敏捷成长 | | [GetAssists](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetAssists) | int GetAssists() | 返回助攻数，数值存储在玩家资源(PlayerResource)中 | | [GetAttacker](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetAttacker) | int GetAttacker(int a) | No Description Set | | [GetBaseAgility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBaseAgility) | float GetBaseAgility() | 返回基础敏捷 | | [GetBaseDamageMax](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBaseDamageMax) | int GetBaseDamageMax() | 返回基础攻击伤害最大值，英雄的基础攻击伤害也会受到属性影响 | | [GetBaseDamageMin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBaseDamageMin) | int GetBaseDamageMin() | 返回基础攻击伤害最小值，英雄的基础攻击伤害也会受到属性影响 | | [GetBaseIntellect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBaseIntellect) | float GetBaseIntellect() | 返回基础智力 | | [GetBaseStrength](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBaseStrength) | float GetBaseStrength() | 返回基础力量 | | [GetBonusDamageFromPrimaryStat](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBonusDamageFromPrimaryStat) | int GetBonusDamageFromPrimaryStat() | 返回来自主属性奖励的攻击伤害值 | | [GetBuybackCooldownTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBuybackCooldownTime) | float GetBuybackCooldownTime() | 返回该英雄买活CD的剩余时间 | | [GetBuybackCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBuybackCost) | int GetBuybackCost() | 返回买活所需金钱 | | [GetBuybackGoldLimitTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetBuybackGoldLimitTime) | float GetBuybackGoldLimitTime() | 返回买活所带来的金钱惩罚剩余时间 | | [GetCurrentXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetCurrentXP) | int GetCurrentXP() | 返回当前经验值 | | [GetDeathGoldCost](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetDeathGoldCost) | int GetDeathGoldCost() | 返回死亡损失金钱 | | [GetDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetDeaths) | int GetDeaths() | 返回死亡次数 | | [GetDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetDenies) | int GetDenies() | 返回反补次数 | | [GetGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetGold) | int GetGold() | 返回该英雄拥有者所有的金钱数量 | | [GetGoldBounty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetGoldBounty) | int GetGoldBounty() | No Description Set | | [GetHealthRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetHealthRegen) | float GetHealthRegen() | 返回每秒生命回复，英雄生命回复受到属性影响 | | [GetIncreasedAttackSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetIncreasedAttackSpeed) | float GetIncreasedAttackSpeed() | 返回攻击速度增加值（IAS），英雄攻击速度受到敏捷影响 | | [GetIntellect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetIntellect) | float GetIntellect() | 返回智力值 | | [GetIntellectGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetIntellectGain) | float GetIntellectGain() | 返回智力成长 | | [GetKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetKills) | int GetKills() | 返回击杀数，数值存储在玩家资源(PlayerResource)中 | | [GetLastHits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetLastHits) | int GetLastHits() | 返回正补数，数值存储在玩家资源(PlayerResource)中 | | [GetManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetManaRegen) | float GetManaRegen() | 返回每秒魔法回复，英雄魔法回复受到属性影响 | | [GetMostRecentDamageTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetMostRecentDamageTime) | float GetMostRecentDamageTime() | 返回最近一次伤害时间 | | [GetMultipleKillCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetMultipleKillCount) | int GetMultipleKillCount() | 返回多杀计数 | | [GetNumAttackers](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetNumAttackers) | int GetNumAttackers() | No Description Set | | [GetPhysicalArmorValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetPhysicalArmorValue) | float GetPhysicalArmorValue() | 返回护甲值，英雄护甲受到属性影响 | | [GetPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetPlayerID) | int GetPlayerID() | 返回英雄拥有者的ID | | [GetPrimaryAttribute](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetPrimaryAttribute) | int GetPrimaryAttribute() | 返回英雄主要属性，0 = 力量, 1 = 敏捷, 2 = 智力. | | [GetPrimaryStatValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetPrimaryStatValue) | float GetPrimaryStatValue() | 返回主属性值 | | [GetRespawnTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetRespawnTime) | float GetRespawnTime() | 返回复活时间 | | [GetStatsBasedManaRegen](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetStatsBasedManaRegen) | float GetStatsBasedManaRegen() | 返回属性提供的魔法回复 | | [GetStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetStreak) | int GetStreak() | 返回连杀数 | | [GetStrength](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetStrength) | float GetStrength() | 返回力量值 | | [GetStrengthGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetStrengthGain) | float GetStrengthGain() | 返回力量成长 | | [GetTimeUntilRespawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.GetTimeUntilRespawn) | float GetTimeUntilRespawn() | 返回复活所需的时间 | | [HasAnyAvailableInventorySpace](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.HasAnyAvailableInventorySpace) | bool HasAnyAvailableInventorySpace() | 返回是否有可用物品栏空间 | | [HasFlyingVision](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.HasFlyingVision) | bool HasFlyingVision() | 返回是否拥有空中视野 | | [HasOwnerAbandoned](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.HasOwnerAbandoned) | bool HasOwnerAbandoned() | No Description Set | | [HasRoomForItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.HasRoomForItem) | int HasRoomForItem(string a, bool b, bool c) | 参数: const char\* pItemName, bool bIncludeStashCombines, bool bAllowSelling | | [HeroLevelUp](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.HeroLevelUp) | void HeroLevelUp(bool a) | 升级英雄, 参数决定是否播放升级效果. | | [IncrementAssists](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementAssists) | void IncrementAssists() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementDeaths) | void IncrementDeaths() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementDenies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementDenies) | void IncrementDenies() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementKills](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementKills) | void IncrementKills(int kills) | 增加击杀数，击杀者为当前英雄，传入ID为受害者ID。数值存储在玩家资源(PlayerResource)中 | | [IncrementLastHitMultikill](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementLastHitMultikill) | void IncrementLastHitMultikill() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementLastHits](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementLastHits) | void IncrementLastHits() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementLastHitStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementLastHitStreak) | void IncrementLastHitStreak() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementNearbyCreepDeaths](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementNearbyCreepDeaths) | void IncrementNearbyCreepDeaths() | 数值存储在玩家资源(PlayerResource)中 | | [IncrementStreak](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IncrementStreak) | void IncrementStreak() | 数值存储在玩家资源(PlayerResource)中 | | [IsBuybackDisabledByReapersScythe](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IsBuybackDisabledByReapersScythe) | bool IsBuybackDisabledByReapersScythe() | 是否被死神镰刀禁止买活 | | [IsReincarnating](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.IsReincarnating) | bool IsReincarnating() | 是否处于重生中 | | [KilledHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.KilledHero) | void KilledHero(handle a, handle b) | 参数: 英雄, 加害者 | | [ModifyAgility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ModifyAgility) | void ModifyAgility(float a) | 增加指定值到英雄基础敏捷，随后调用CalculateStatBonus | | [ModifyGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ModifyGold) | int ModifyGold(int goldAmmt, bool reliable, int reason) | 给予英雄金钱。参数: int 金钱数, bool 是否为可靠金钱, int 原因 | | [ModifyIntellect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ModifyIntellect) | void ModifyIntellect(float a) | 增加指定值到英雄基础智力，随后调用CalculateStatBonus | | [ModifyStrength](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ModifyStrength) | void ModifyStrength(float a) | 增加指定值到英雄基础力量，随后调用CalculateStatBonus | | [PerformTaunt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.PerformTaunt) | void PerformTaunt() | No Description Set | | [RecordLastHit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.RecordLastHit) | void RecordLastHit() | No Description Set | | [RespawnHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.RespawnHero) | void RespawnHero(bool buyback, bool unknown1, bool unknown2) |  | | [SetAbilityPoints](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetAbilityPoints) | void SetAbilityPoints(int a) | 设置当前可用技能点 | | [SetBaseAgility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBaseAgility) | void SetBaseAgility(float a) | 设置基础敏捷 | | [SetBaseIntellect](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBaseIntellect) | void SetBaseIntellect(float a) | 设置基础智力 | | [SetBaseStrength](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBaseStrength) | void SetBaseStrength(float a) | 设置基础力量 | | [SetBuybackCooldownTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBuybackCooldownTime) | void SetBuybackCooldownTime(float a) | 设置买活冷却时间 | | [SetBuyBackDisabledByReapersScythe](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBuyBackDisabledByReapersScythe) | void SetBuyBackDisabledByReapersScythe(bool a) | 设置是否被死神镰刀禁止买活 | | [SetBuybackGoldLimitTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetBuybackGoldLimitTime) | void SetBuybackGoldLimitTime(float a) | 设置买活后金钱惩罚时间 | | [SetCustomDeathXP](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetCustomDeathXP) | void SetCustomDeathXP(int a) | GameRules boolean must be set for this to work!}} | | [SetGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetGold) | void SetGold(int amount, bool reliableGold) | 设置英雄拥有者的金钱 | | [SetPlayerID](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetPlayerID) | void SetPlayerID(int a) | No Description Set | | [SetRespawnPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetRespawnPosition) | void SetRespawnPosition(Vector a) | No Description Set | | [SetTimeUntilRespawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SetTimeUntilRespawn) | void SetTimeUntilRespawn(float a) | No Description Set | | [ShouldDoFlyHeightVisual](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.ShouldDoFlyHeightVisual) | bool ShouldDoFlyHeightVisual() | No Description Set | | [SpendGold](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.SpendGold) | void SpendGold(int amount, int reason) | 参数: int 金钱, int 原因 | | [UnitCanRespawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.UnitCanRespawn) | bool UnitCanRespawn() | No Description Set | | [UpgradeAbility](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.UpgradeAbility) | void UpgradeAbility(handle a) | 升级指定技能，需要英雄拥有该技能和技能点 | | [WillReincarnate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Hero.WillReincarnate) | bool WillReincarnate() | No Description Set | |
|  |
| CDOTA\_BaseNPC\_Creature |
| **extends** [**CDOTA\_BaseNPC**](#CDOTA_BaseNPC) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddItemDrop](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.AddItemDrop) | void AddItemDrop(handle a) | 为该生物添加特别的道具掉落 | | [CreatureLevelUp](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.CreatureLevelUp) | void CreatureLevelUp(int a) | 升级目标生物指定级别 | | [IsChampion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.IsChampion) | bool IsChampion() | 是否为冠军生物 | | [SetArmorGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetArmorGain) | void SetArmorGain(float a) | 设置该生物每级增加的护甲 | | [SetAttackTimeGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetAttackTimeGain) | void SetAttackTimeGain(float a) | 设置该生物每级增加的攻击速度 | | [SetBountyGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetBountyGain) | void SetBountyGain(int a) | 设置该生物每级增加的奖励金币 | | [SetChampion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetChampion) | void SetChampion(bool a) | 将该生物标记为冠军生物 | | [SetDamageGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetDamageGain) | void SetDamageGain(int a) | 设置该生物每级增加的攻击伤害 | | [SetDisableResistanceGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetDisableResistanceGain) | void SetDisableResistanceGain(float a) | 设置该生物每级增加的控制抗性 | | [SetHPGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetHPGain) | void SetHPGain(int a) | 设置该生物每级增加的生命值 | | [SetHPRegenGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetHPRegenGain) | void SetHPRegenGain(float a) | 设置该生物每级增加的生命回复 | | [SetMagicResistanceGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetMagicResistanceGain) | void SetMagicResistanceGain(float a) | 设置该生物每级增加的魔法抗性 | | [SetManaGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetManaGain) | void SetManaGain(int a) | 设置该生物每级增加的魔法值 | | [SetManaRegenGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetManaRegenGain) | void SetManaRegenGain(float a) | 设置该生物每级增加的魔法回复 | | [SetMoveSpeedGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetMoveSpeedGain) | void SetMoveSpeedGain(int a) | 设置该生物每级增加的移动速度 | | [SetXPGain](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Creature.SetXPGain) | void SetXPGain(int a) | 设置该生物每级增加的经验值 | |
|  |
| CDOTA\_BaseNPC\_Building |
| **extends** [**CDOTA\_BaseNPC**](#CDOTA_BaseNPC) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetInvulnCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Building.GetInvulnCount) | int GetInvulnCount() | 返回建筑无敌计数 | | [SetInvulnCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_BaseNPC_Building.SetInvulnCount) | void SetInvulnCount(int a) | 设置建筑无敌计数 | |
|  |
| CRPG\_Unit |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ActionState](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.ActionState) | handle ActionState() | 返回该单位的ActionState物体 | | [ClearMovementTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.ClearMovementTarget) | void ClearMovementTarget() | 清除移动目标实体或点 | | [FindSensedEnemies](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.FindSensedEnemies) | table FindSensedEnemies() | 返回该单位视野锥或感应球体内所有敌方单位列表 | | [GetMaxSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetMaxSpeed) | float GetMaxSpeed() | 返回最大移动速度 | | [GetMaxStamina](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetMaxStamina) | float GetMaxStamina() | 返回最大耐力值 | | [GetMovementTargetEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetMovementTargetEntity) | handle GetMovementTargetEntity() | 返回移动目标实体，如果已经设置移动目标实体 | | [GetSensingSphereRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetSensingSphereRange) | float GetSensingSphereRange() | 返回单位的360度感应球体范围 | | [GetSightConeAngle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetSightConeAngle) | float GetSightConeAngle() | 返回单位视野锥的夹角 | | [GetSightConeRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetSightConeRange) | float GetSightConeRange() | 返回单位视野锥的范围 | | [GetStamina](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetStamina) | float GetStamina() | 返回当前耐力值 | | [GetTurnRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetTurnRate) | float GetTurnRate() | 返回单位转身速度，单位为度每秒 | | [GetUnitName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GetUnitName) | string GetUnitName() | 返回单位名字 | | [GrantItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.GrantItem) | void GrantItem(string a, bool b) | 授予单位物品 | | [IsBlocking](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.IsBlocking) | bool IsBlocking() | 单位是否正在格挡（不确定） | | [IsFacing](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.IsFacing) | bool IsFacing(Vector a, float b) | 参数为（Vector目标点，float允许角度差）如果单位朝向目标角度低于角度差则返还true | | [SetBlocking](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetBlocking) | void SetBlocking(bool a) | 设置单位格挡状态（不确定） | | [SetMaxSpeed](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetMaxSpeed) | void SetMaxSpeed(float a) | 设置单位最大速度 | | [SetMovementTargetEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetMovementTargetEntity) | void SetMovementTargetEntity(handle a, float b) | 参数为（目标实体，跟随范围）单位向目标实体移动直到抵达指定范围内 | | [SetMovementTargetPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetMovementTargetPosition) | void SetMovementTargetPosition(Vector a, float b) | 参数为（目标点，跟随范围）单位向目标点移动直到抵达指定范围内 | | [SetSensingSphereRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetSensingSphereRange) | void SetSensingSphereRange(float a) | 设置单位的360度感应球体范围 | | [SetSightConeAngle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetSightConeAngle) | void SetSightConeAngle(float a) | 设置单位的视野锥夹角 | | [SetSightConeRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetSightConeRange) | void SetSightConeRange(float a) | 设置单位的视野锥范围 | | [SetTurnRate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CRPG_Unit.SetTurnRate) | void SetTurnRate(float a) | 设置单位转向速率（度每秒） | |
|  |
| CDOTABaseGameMode |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ClientLoadGridNav](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.ClientLoadGridNav) | void ClientLoadGridNav() | 通知客户端需要载入网格信息，可用于允许客户端确定允许建筑的区域等 | | [SetAlwaysShowPlayerInventory](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetAlwaysShowPlayerInventory) | void SetAlwaysShowPlayerInventory(bool a) | 在HUD中显示玩家英雄的物品栏，不管选择了哪个单位 | | [SetAnnouncerDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetAnnouncerDisabled) | void SetAnnouncerDisabled(bool enabled) | Disables the dota announcer | | [SetBotThinkingEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetBotThinkingEnabled) | void SetBotThinkingEnabled(bool a) | 允许/禁止机器人思考，需要与Dota PvP高度相似的三路线地图、商店等 | | [SetBuybackEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetBuybackEnabled) | void SetBuybackEnabled(bool a) | 完全允许/禁止买活 | | [SetCameraDistanceOverride](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetCameraDistanceOverride) | void SetCameraDistanceOverride(float a) | 设置默认的镜头距离。Dota默认为1134 | | [SetCustomBuybackCooldownEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetCustomBuybackCooldownEnabled) | void SetCustomBuybackCooldownEnabled(bool a) | 开启该选项来允许自定义买活冷却时间 | | [SetCustomBuybackCostEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetCustomBuybackCostEnabled) | void SetCustomBuybackCostEnabled(bool a) | 开启该选项来允许自定义买活花费 | | [SetCustomHeroMaxLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetCustomHeroMaxLevel) | void SetCustomHeroMaxLevel(int maxLevel) | 定义最大英雄等级（默认为25） | | [SetCustomXPRequiredToReachNextLevel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetCustomXPRequiredToReachNextLevel) | void SetCustomXPRequiredToReachNextLevel(handle a) | 定义英雄经验值表(table) | | [SetFixedRespawnTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetFixedRespawnTime) | void SetFixedRespawnTime(float time) | Sets the dota respawn time. -1 for default behavior | | [SetFogOfWarDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetFogOfWarDisabled) | void SetFogOfWarDisabled(bool a) | 开关战争迷雾 | | [SetGoldSoundDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetGoldSoundDisabled) | void SetGoldSoundDisabled(bool a) | 是否禁止获取金钱时的声音提示 | | [SetOverrideSelectionEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetOverrideSelectionEntity) | void SetOverrideSelectionEntity(handle unit) | 覆盖默认选择实体，替代所有玩家的英雄 | | [SetRecommendedItemsDisabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetRecommendedItemsDisabled) | void SetRecommendedItemsDisabled(bool a) | 是否禁止显示商店中的推荐购买物品 | | [SetRemoveIllusionsOnDeath](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetRemoveIllusionsOnDeath) | void SetRemoveIllusionsOnDeath(bool a) | 使幻象死亡时立即消失，而不是延迟数秒 | | [SetTopBarTeamValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetTopBarTeamValue) | void SetTopBarTeamValue(int a, int b) | 设置顶端的队伍数值 | | [SetTopBarTeamValuesOverride](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetTopBarTeamValuesOverride) | void SetTopBarTeamValuesOverride(bool a) | 是否覆盖顶端的队伍数值 | | [SetTopBarTeamValuesVisible](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetTopBarTeamValuesVisible) | void SetTopBarTeamValuesVisible(bool a) | 开关顶端的队伍数值 | | [SetTowerBackdoorProtectionEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetTowerBackdoorProtectionEnabled) | void SetTowerBackdoorProtectionEnabled(bool a) | 开关偷塔保护 | | [SetUseCustomHeroLevels](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTABaseGameMode.SetUseCustomHeroLevels) | void SetUseCustomHeroLevels(bool a) | 开关自定义英雄英雄经验表，该表必须提前被定义 | |
|  |
| CDotaQuest |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***A quest, as seen in the Tutorial and Frostivus*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddSubquest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.AddSubquest) | void AddSubquest(handle a) | 为该任务添加子任务 | | [CompleteQuest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.CompleteQuest) | void CompleteQuest() | 标记该任务完成 | | [GetSubquest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.GetSubquest) | handle GetSubquest(int a) | 用索引号找到该任务的子任务 | | [GetSubquestByName](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.GetSubquestByName) | handle GetSubquestByName(string a) | 用任务名称找到该任务的子任务 | | [RemoveSubquest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.RemoveSubquest) | void RemoveSubquest(handle a) | 从该任务移除一个子任务 | | [SetTextReplaceString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.SetTextReplaceString) | void SetTextReplaceString(string a) | 设置该任务的文本取代字符串 | | [SetTextReplaceValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaQuest.SetTextReplaceValue) | void SetTextReplaceValue(int a, int b) | 设置任务数值 | |
|  |
| CDotaSubquestBase |
| **extends** [**CDotaQuest**](#CDotaQuest) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CompleteSubquest](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaSubquestBase.CompleteSubquest) | void CompleteSubquest() | 完成子任务 | | [SetTextReplaceString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaSubquestBase.SetTextReplaceString) | void SetTextReplaceString(string a) | 设置该子任务的文本取代字符串 | | [SetTextReplaceValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDotaSubquestBase.SetTextReplaceValue) | void SetTextReplaceValue(int a, int b) | 设置子任务数值 | |
|  |
| CPhysicsComponent |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [ExpensiveInstantRayCast](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPhysicsComponent.ExpensiveInstantRayCast) | bool ExpensiveInstantRayCast(Vector a, Vector b, handle c) | 进行一个立即（即阻挡）的光线投射，稍后会有handle/queue版本，并不计划实际使用这个！ | |
|  |
| CPointTemplate |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [DeleteCreatedSpawnGroups](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPointTemplate.DeleteCreatedSpawnGroups) | void DeleteCreatedSpawnGroups() | 删除任何该点模板的刷新组。提示：并不删除点模板 | | [ForceSpawn](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPointTemplate.ForceSpawn) | void ForceSpawn() | 强制刷新该点模板指向的全部实体 | | [GetSpawnedEntities](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPointTemplate.GetSpawnedEntities) | handle GetSpawnedEntities() | 获取最近被刷新的实体 | | [SetSpawnCallback](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPointTemplate.SetSpawnCallback) | void SetSpawnCallback(handle a, handle b) | 设置刷新回调（hCallbackFunc, hCallbackScope, hCallbackData ），当模板产生实体时回调。被刷新的实体作为数组被传入 | |
|  |
| CBodyComponent |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddImpulseAtPosition](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.AddImpulseAtPosition) | void AddImpulseAtPosition(Vector a, Vector b) | 在指定的世界位置施加一个物理推动 | | [AddVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.AddVelocity) | void AddVelocity(Vector a, Vector b) | 为物理物件增加一个线速度或角速度 | | [DetachFromParent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.DetachFromParent) | void DetachFromParent() | 从其父项（parent）中分离 | | [GetSequence](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.GetSequence) | <> GetSequence() | 返回激活的序列（sequence） | | [IsAttachedToParent](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.IsAttachedToParent) | bool IsAttachedToParent() | 是否依附父项（parent） | | [LookupSequence](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.LookupSequence) | <> LookupSequence(string a) | 输入名字返回序列（sequence）ID | | [SequenceDuration](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SequenceDuration) | float SequenceDuration(string a) | 按秒数返回指定序列（sequence）的持续时间 | | [SetAngularVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SetAngularVelocity) | void SetAngularVelocity(Vector a) | No Description Set | | [SetAnimation](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SetAnimation) | void SetAnimation(string a) | 输入动画名称播放动作，需要模型拥有该动画 | | [SetBodyGroup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SetBodyGroup) | void SetBodyGroup(string a) | No Description Set | | [SetMaterialGroup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SetMaterialGroup) | void SetMaterialGroup(utlstringtoken a) | No Description Set | | [SetVelocity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBodyComponent.SetVelocity) | void SetVelocity(Vector velocity) | No Description Set | |
|  |
| CBaseAnimating |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***A class containing functions involved in animations*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetAttachmentAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.GetAttachmentAngles) | Vector GetAttachmentAngles(int a) | 获得指定ID附件的角度，返回vector(p,y,r) | | [GetAttachmentOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.GetAttachmentOrigin) | Vector GetAttachmentOrigin(int a) | 获得指定ID附件的源vector | | [IsSequenceFinished](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.IsSequenceFinished) | bool IsSequenceFinished() | 查询主序列是否播放结束 | | [ScriptLookupAttachment](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.ScriptLookupAttachment) | int ScriptLookupAttachment(string a) | 获得指定名序列的ID | | [SetBodygroup](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.SetBodygroup) | void SetBodygroup(int a, int b) | 设置一个bodygroup | | [SetModelScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.SetModelScale) | void SetModelScale(float scale) | 设置模型放大率。如果模型缩放为1，使用SetModelScale(10.0)将放大率设置成10.0. | | [SetPoseParameter](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseAnimating.SetPoseParameter) | float SetPoseParameter(string a, float b) | 将指定的位置参数（pose parameter）设定为指定值 | |
|  |
| CBaseCombatCharacter |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetEquippedWeapons](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseCombatCharacter.GetEquippedWeapons) | table GetEquippedWeapons() | GetEquippedWeapons(): 获取一个装备武器的数组 | | [GetWeaponCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseCombatCharacter.GetWeaponCount) | int GetWeaponCount() | GetWeaponCount(): 获取当前装备的武器数量 | |
|  |
| ProjectileManager |
| ***The projectile manager, it manages projectiles.*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CreateLinearProjectile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\ProjectileManager.CreateLinearProjectile) | int CreateLinearProjectile(table infoTable) | 创建一个线性投射物并返回其ID | | [CreateTrackingProjectile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\ProjectileManager.CreateTrackingProjectile) | void CreateTrackingProjectile(handle a) | 创建一个追踪投射物 | | [DestroyLinearProjectile](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\ProjectileManager.DestroyLinearProjectile) | void DestroyLinearProjectile(int a) | 摧毁指定ID的线性投射物 | | [ProjectileDodge](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\ProjectileManager.ProjectileDodge) | void ProjectileDodge(handle a) | 使得特定单位躲避投射物 | |
|  |
| CBaseTrigger |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [Disable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseTrigger.Disable) | void Disable() | 禁止触发器 | | [Enable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseTrigger.Enable) | void Enable() | 允许触发器 | | [IsTouching](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseTrigger.IsTouching) | bool IsTouching(handle a) | 检查指定实体是否正在接触触发器 | |
|  |
| CEnvEntityMaker |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SpawnEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvEntityMaker.SpawnEntity) | void SpawnEntity() | 在生成器的位置创建一个实体 | | [SpawnEntityAtEntityOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvEntityMaker.SpawnEntityAtEntityOrigin) | void SpawnEntityAtEntityOrigin(handle a) | 在指定实体的位置立刻创建一个实体 | | [SpawnEntityAtLocation](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvEntityMaker.SpawnEntityAtLocation) | void SpawnEntityAtLocation(Vector a, Vector b) | 在指定位置和方向创建一个实体，方向是角度制欧拉角(pitch, yaw, roll) | | [SpawnEntityAtNamedEntityOrigin](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvEntityMaker.SpawnEntityAtNamedEntityOrigin) | void SpawnEntityAtNamedEntityOrigin(string a) | 在指定名字实体的位置创建一个实体 | |
|  |
| CDOTAVoteSystem |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [StartVote](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAVoteSystem.StartVote) | void StartVote(handle a) | 开始投票，详细由传入的table制定 | |
|  |
| CMarkupVolumeTagged |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [HasTag](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CMarkupVolumeTagged.HasTag) | bool HasTag(string a) | 查询该卷是否带有指定标签 | |
|  |
| CScriptPrecacheContext |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddResource](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptPrecacheContext.AddResource) | void AddResource(string a) | 预缓存指定资源 | | [GetValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptPrecacheContext.GetValue) | table GetValue(string a) | 读取一个键值（spawn key） | |
|  |
| CScriptKeyValues |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetValue](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptKeyValues.GetValue) | table GetValue(string a) | Reads a spawn key | |
|  |
| CScriptParticleManager |
| ***No Description Set*** |
|  |
| ***Global accessor variable:* ParticleManager** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CreateParticle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.CreateParticle) | int CreateParticle(string particleName, int particleAttach, handle owningEntity) | 创建一个新的粒子特效 | | [CreateParticleForPlayer](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.CreateParticleForPlayer) | int CreateParticleForPlayer(string particleName, int particleAttach, handle owningEntity, handle owningPlayer) | 创建一个只对指定玩家播放的粒子特效 | | [DestroyParticle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.DestroyParticle) | void DestroyParticle(int particleID, bool immediately) | Destroys particle. | | [GetParticleReplacement](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.GetParticleReplacement) | string GetParticleReplacement(string a, handle b) | No Description Set | | [ReleaseParticleIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.ReleaseParticleIndex) | void ReleaseParticleIndex(int particleId) | 施放指定粒子特效索引号 | | [SetParticleAlwaysSimulate](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.SetParticleAlwaysSimulate) | void SetParticleAlwaysSimulate(int a) | No Description Set | | [SetParticleControl](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.SetParticleControl) | void SetParticleControl(int particleId, int controlIndex, Vector controlData) | 设置控制粒子特效的控制点数据 | | [SetParticleControlEnt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptParticleManager.SetParticleControlEnt) | void SetParticleControlEnt(int a, int b, handle c, int d, string e, Vector f, bool g) | No Description Set | |
|  |
| CScriptHeroList |
| ***No Description Set*** |
|  |
| ***Global accessor variable:* HeroList** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetAllHeroes](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptHeroList.GetAllHeroes) | table GetAllHeroes() | 返回当前世界全部英雄 | | [GetHero](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptHeroList.GetHero) | handle GetHero(int heroId) | 获取英雄列表中第N位英雄 | | [GetHeroCount](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CScriptHeroList.GetHeroCount) | int GetHeroCount() | 返回当前世界英雄数目 | |
|  |
| CNativeOutputs |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddOutput](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CNativeOutputs.AddOutput) | void AddOutput(string a, string b) | 增加一个输出 | | [Init](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CNativeOutputs.Init) | void Init(int a) | 初始化输出 | |
|  |
| CEnvProjectedTexture |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [SetFarRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvProjectedTexture.SetFarRange) | void SetFarRange(float a) | 设置光照最大距离 | | [SetLinearAttenuation](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvProjectedTexture.SetLinearAttenuation) | void SetLinearAttenuation(float a) | 设置光照线性淡化值 | | [SetNearRange](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvProjectedTexture.SetNearRange) | void SetNearRange(float a) | 设置光照最小距离 | | [SetQuadraticAttenuation](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvProjectedTexture.SetQuadraticAttenuation) | void SetQuadraticAttenuation(float a) | 设置光照二次项淡化值 | | [SetVolumetrics](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CEnvProjectedTexture.SetVolumetrics) | void SetVolumetrics(bool a, float b, float c, int d, float e) | 开关体积光参数: bool bOn, float flIntensity, float flNoise, int nPlanes, float flPlaneOffset | |
|  |
| CInfoData |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [QueryColor](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryColor) | Vector QueryColor(utlstringtoken a, Vector b) | 查询该键值颜色数据 | | [QueryFloat](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryFloat) | float QueryFloat(utlstringtoken a, float b) | 查询该键值浮点（float）数据 | | [QueryInt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryInt) | int QueryInt(utlstringtoken a, int b) | 查询该键值整型（int）数据 | | [QueryNumber](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryNumber) | float QueryNumber(utlstringtoken a, float b) | 查询该键值数值数据 | | [QueryString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryString) | string QueryString(utlstringtoken a, string b) | 查询该键值字符串（string）数据 | | [QueryVector](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CInfoData.QueryVector) | Vector QueryVector(utlstringtoken a, Vector b) | 查询该键值矢量（vector）数据 | |
|  |
| CPhysicsProp |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [DisableMotion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPhysicsProp.DisableMotion) | void DisableMotion() | 禁止该元件运动 | | [EnableMotion](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CPhysicsProp.EnableMotion) | void EnableMotion() | 允许该元件运动 | |
|  |
| CDOTAGamerules |
| **'** |
|  |
| ***Global accessor variable:* GameRules** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [Defeated](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.Defeated) | void Defeated() | 摧毁遗迹等 | | [DidMatchSignoutTimeOut](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.DidMatchSignoutTimeOut) | bool DidMatchSignoutTimeOut() | 当游戏结束，等待弹窗时为真 | | [GetCustomGameDifficulty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetCustomGameDifficulty) | int GetCustomGameDifficulty() | 返回自定义游戏模式难度 | | [GetDifficulty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetDifficulty) | int GetDifficulty() | 返回自定义游戏模式难度 | | [GetDroppedItem](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetDroppedItem) | handle GetDroppedItem(int dropIndex) | 获得第X个掉落物品 | | [GetGameModeEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetGameModeEntity) | handle GetGameModeEntity() | 设置游戏模式实体 | | [GetGameTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetGameTime) | float GetGameTime() | 返回游戏开始后经过的秒数，暂停时间不计算在内 | | [GetMatchSignoutComplete](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetMatchSignoutComplete) | bool GetMatchSignoutComplete() | 是否已经接受包含回报信息的弹窗 | | [GetNianFightStartTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetNianFightStartTime) | float GetNianFightStartTime() | 获得年兽大战开始时间 | | [GetNianTotalDamageTaken](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetNianTotalDamageTaken) | int GetNianTotalDamageTaken() | 芳晓活动，获得年兽受到的总伤害 | | [GetTimeOfDay](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.GetTimeOfDay) | float GetTimeOfDay() | 获取一天中的时间（游戏时间） | | [IsDaytime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.IsDaytime) | bool IsDaytime() | 是否是白天 | | [MakeTeamLose](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.MakeTeamLose) | void MakeTeamLose(int team) | 使指定队伍失败 | | [NumDroppedItems](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.NumDroppedItems) | int NumDroppedItems() | 返回当前掉落在地面的物品数量 | | [Playtesting\_UpdateAddOnKeyValues](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.Playtesting_UpdateAddOnKeyValues) | void Playtesting\_UpdateAddOnKeyValues() | 从磁盘数据中更新自定义英雄、单位、技能的键值 | | [ResetDefeated](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.ResetDefeated) | void ResetDefeated() | 遗迹摧毁后重新开始 | | [ResetToHeroSelection](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.ResetToHeroSelection) | void ResetToHeroSelection() | 重新开始时选择英雄 | | [SendCustomMessage](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SendCustomMessage) | void SendCustomMessage(string message, int teamID, int unknown(1?)) | 在左侧文本框显示一行文本，通常用来提示死亡、反补队友、买活等信息。使用严格HTML格式(>;br<;,>;u<;,>;font<;) | | [SetCreepMinimapIconScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetCreepMinimapIconScale) | void SetCreepMinimapIconScale(float scale) | 在小地图上缩放中立生物图标 | | [SetCustomGameDifficulty](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetCustomGameDifficulty) | void SetCustomGameDifficulty(int a) | 设置自定义游戏模式难度等级 | | [SetCustomGameEndDelay](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetCustomGameEndDelay) | void SetCustomGameEndDelay(float delay) | Sets the delay time until the game ends | | [SetCustomVictoryMessageDuration](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetCustomVictoryMessageDuration) | void SetCustomVictoryMessageDuration(float duration) | Sets how long the custom victory message should last | | [SetFirstBloodActive](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetFirstBloodActive) | void SetFirstBloodActive(bool a) | 设置第一滴血是否已经产生 | | [SetGameWinner](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetGameWinner) | void SetGameWinner(int team) | 使指定队伍胜利 | | [SetGoldPerTick](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetGoldPerTick) | void SetGoldPerTick(int a) | 设置每个时间间隔获得的金币 | | [SetGoldTickTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetGoldTickTime) | void SetGoldTickTime(float a) | 设置获得金币的时间周期 | | [SetHeroMinimapIconScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetHeroMinimapIconScale) | void SetHeroMinimapIconScale(int iconSize) | 设置小地图英雄图标尺寸 | | [SetHeroRespawnEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetHeroRespawnEnabled) | void SetHeroRespawnEnabled(bool canRespawn) | 设置是否使用默认英雄复活规则 | | [SetHeroSelectionTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetHeroSelectionTime) | void SetHeroSelectionTime(float time) | 设置选择英雄的时间 | | [SetHideKillMessageHeaders](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetHideKillMessageHeaders) | void SetHideKillMessageHeaders(bool hide) | Sets whether or not the kill banners should be hidden | | [SetNianFightStartTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetNianFightStartTime) | void SetNianFightStartTime(float a) | 设置年兽大战开始时间 | | [SetOverlayHealthBarUnit](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetOverlayHealthBarUnit) | void SetOverlayHealthBarUnit(handle unit, int style) | 在前端蒙版生命条上显示单位生命值 | | [SetPostGameTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetPostGameTime) | void SetPostGameTime(float time) | 设置在结束游戏后服务器与玩家断线前的时间 | | [SetPreGameTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetPreGameTime) | void SetPreGameTime(float time) | 设置选择英雄与开始游戏之间的时间 | | [SetRuneMinimapIconScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetRuneMinimapIconScale) | void SetRuneMinimapIconScale(float scale) | 缩放小地图神符图标 | | [SetRuneSpawnTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetRuneSpawnTime) | void SetRuneSpawnTime(float time) | 设置神符刷新时间 | | [SetSafeToLeave](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetSafeToLeave) | void SetSafeToLeave(bool safeToLeave) | 标记游戏可安全离开 | | [SetSameHeroSelectionEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetSameHeroSelectionEnabled) | void SetSameHeroSelectionEnabled(bool enabled) | 允许选择重复英雄 | | [SetTimeOfDay](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetTimeOfDay) | void SetTimeOfDay(float time) | 设置一天中的时间（游戏时间） | | [SetTreeRegrowTime](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetTreeRegrowTime) | void SetTreeRegrowTime(float time) | 设置树重新生长的时间（秒） | | [SetUseBaseGoldBountyOnHeroes](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetUseBaseGoldBountyOnHeroes) | void SetUseBaseGoldBountyOnHeroes(bool a) | 英雄将使用基础NPC规则来决定赏金，而不是DOTA指定规则 | | [SetUseCustomHeroXPValues](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetUseCustomHeroXPValues) | void SetUseCustomHeroXPValues(bool a) | 允许英雄提供指定数目的经验值（必须先设置） | | [SetUseUniversalShopMode](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.SetUseUniversalShopMode) | void SetUseUniversalShopMode(bool enabled) | 为真时，所有物品当处于任意商店范围内时都能购买到，包括秘密商店物品 | | [State\_Get](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTAGamerules.State_Get) | <> State\_Get() | 获取当前游戏规则状态 | |
|  |
| CToneMapControllerComponent |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetBloomScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.GetBloomScale) | float GetBloomScale() | 获取该光泽贴图控制器（tonemap controller）的泛光缩放（bloomscale) | | [GetMaxExposure](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.GetMaxExposure) | float GetMaxExposure() | 获取该光泽贴图控制器（tonemap controller）的最大曝光（max exposure） | | [GetMinExposure](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.GetMinExposure) | float GetMinExposure() | 获取该光泽贴图控制器（tonemap controller）的最小曝光（min exposure） | | [SetBloomScale](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.SetBloomScale) | void SetBloomScale(float a) | 设置该光泽贴图控制器（tonemap controller）的泛光缩放（bloomscale) | | [SetMaxExposure](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.SetMaxExposure) | void SetMaxExposure(float a) | 设置该光泽贴图控制器（tonemap controller）的最大曝光（max exposure） | | [SetMinExposure](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CToneMapControllerComponent.SetMinExposure) | void SetMinExposure(float a) | 设置该光泽贴图控制器（tonemap controller）的最小曝光（min exposure） | |
|  |
| CDebugOverlayScriptHelper |
| ***No Description Set*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [Axis](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Axis) | void Axis(Vector a, Quaternion b, float c, bool d, float e) | 绘制一个轴线。在世界空间中指定出发点和方向 | | [Box](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Box) | void Box(Vector a, Vector b, int c, int d, int e, int f, bool g, float h) | 绘制一个世界空间轴向盒。在世界空间中指定边界 | | [BoxAngles](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.BoxAngles) | void BoxAngles(Vector a, Vector b, Vector c, Quaternion d, int e, int f, int g, int h, bool i, float j) | 在出发点绘制一个定向盒。在世界空间中指定边界 | | [Capsule](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Capsule) | void Capsule(Vector a, Quaternion b, float c, float d, int e, int f, int g, int h, bool i, float j) | 绘制一个胶囊体。在世界空间中指定基部 | | [Circle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Circle) | void Circle(Vector a, Quaternion b, float c, int d, int e, int f, int g, bool h, float i) | 绘制一个圆。在世界空间中指定中心 | | [CircleScreenOriented](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.CircleScreenOriented) | void CircleScreenOriented(Vector a, float b, int c, int d, int e, int f, bool g, float h) | 绘制一个指向屏幕的圆。在世界空间中指定中心 | | [Cone](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Cone) | void Cone(Vector a, Vector b, float c, float d, int e, int f, int g, int h, bool i, float j) | 绘制一个线框锥形。在世界空间中指定结束点和方向 | | [Cross](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Cross) | void Cross(Vector a, float b, int c, int d, int e, int f, bool g, float h) | 绘制一个对齐屏幕的十字。在世界空间中指定出发点 | | [Cross3D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Cross3D) | void Cross3D(Vector a, float b, int c, int d, int e, int f, bool g, float h) | 绘制一个对齐世界的十字。在世界空间中指定出发点 | | [Cross3DOriented](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Cross3DOriented) | void Cross3DOriented(Vector a, Quaternion b, float c, int d, int e, int f, int g, bool h, float i) | 绘制一个指向的十字。在世界空间中指定出发点 | | [DrawTickMarkedLine](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.DrawTickMarkedLine) | void DrawTickMarkedLine(Vector a, Vector b, float c, int d, int e, int f, int g, int h, bool i, float j) | 绘制一个短划线。在世界空间中指定结束点 | | [EntityAttachments](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.EntityAttachments) | void EntityAttachments(ehandle a, float b) | 绘制实体附件 | | [EntityAxis](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.EntityAxis) | void EntityAxis(ehandle a, float b, bool c, float d) | 绘制实体源坐标轴 | | [EntityBounds](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.EntityBounds) | void EntityBounds(ehandle a, int b, int c, int d, int e, bool f, float g) | 绘制实体边界 | | [EntitySkeleton](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.EntitySkeleton) | void EntitySkeleton(ehandle a, float b) | 绘制实体骨架 | | [EntityText](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.EntityText) | void EntityText(ehandle a, int b, string c, int d, int e, int f, int g, float h) | 在实体上绘制文本 | | [FilledRect2D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.FilledRect2D) | void FilledRect2D(Vector2D a, Vector2D b, int c, int d, int e, int f, float g) | 绘制一个2D矩形填充的屏幕空间。按照像素对齐 | | [HorzArrow](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.HorzArrow) | void HorzArrow(Vector a, Vector b, float c, int d, int e, int f, int g, bool h, float i) | 绘制一个水平箭头。在世界空间中指定结束点 | | [Line](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Line) | void Line(Vector a, Vector b, int c, int d, int e, int f, bool g, float h) | 绘制一个两点间连线 | | [Line2D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Line2D) | void Line2D(Vector2D a, Vector2D b, int c, int d, int e, int f, float g) | 绘制一个屏幕空间中亮点间连线 | | [PopDebugOverlayScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.PopDebugOverlayScope) | void PopDebugOverlayScope() | 弹出蒙版分组用的标记，被标记的蒙版可被成批删除 | | [PushAndClearDebugOverlayScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.PushAndClearDebugOverlayScope) | void PushAndClearDebugOverlayScope(utlstringtoken a) | 推入一个蒙版分组用的标记。删除所有使用该蒙版ID的蒙版 | | [PushDebugOverlayScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.PushDebugOverlayScope) | void PushDebugOverlayScope(utlstringtoken a) | 推入一个蒙版分组用的标记。被标记的蒙版可被成批删除 | | [RemoveAllInScope](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.RemoveAllInScope) | void RemoveAllInScope(utlstringtoken a) | 移除指定标记的全部蒙版，无视其生命周期 | | [SolidCone](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.SolidCone) | void SolidCone(Vector a, Vector b, float c, float d, int e, int f, int g, int h, bool i, float j) | 绘制一个固体锥。在世界空间中指定出发点和方向 | | [Sphere](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Sphere) | void Sphere(Vector a, float b, int c, int d, int e, int f, bool g, float h) | 绘制一个线框球体。在世界空间中指定中心 | | [SweptBox](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.SweptBox) | void SweptBox(Vector a, Vector b, Vector c, Vector d, Quaternion e, int f, int g, int h, int i, float j) | 绘制一个扫描盒。在世界空间中指定结束点并在本地空间中指定边界 | | [Text](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Text) | void Text(Vector a, int b, string c, float d, int e, int f, int g, int h, float i) | 绘制2D文本。在世界空间中指定出发点 | | [Texture](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Texture) | void Texture(string a, Vector2D b, Vector2D c, int d, int e, int f, int g, Vector2D h, Vector2D i, float j) | 绘制一个屏幕空间纹理。按照像素对齐 | | [Triangle](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.Triangle) | void Triangle(Vector a, Vector b, Vector c, int d, int e, int f, int g, bool h, float i) | 绘制一个填充三角形。在世界空间中指定顶部 | | [UnitTestCycleOverlayRenderType](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.UnitTestCycleOverlayRenderType) | void UnitTestCycleOverlayRenderType() | 转换蒙版混合类型, 用于单位测试 | | [VectorText3D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.VectorText3D) | void VectorText3D(Vector a, Quaternion b, string c, int d, int e, int f, int g, bool h, float i) | 绘制3D文本。在世界空间中指定出发点和方向 | | [VertArrow](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.VertArrow) | void VertArrow(Vector a, Vector b, float c, int d, int e, int f, int g, bool h, float i) | 绘制一个垂直箭头。在世界空间中指定结束点 | | [YawArrow](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDebugOverlayScriptHelper.YawArrow) | void YawArrow(Vector a, float b, float c, float d, int e, int f, int g, int h, bool i, float j) | 绘制一个指定偏航角的箭头。在世界空间中指定结束点 | |
|  |
| CBaseFlex |
| **extends** [**CBaseAnimating**](#CBaseAnimating) |
| ***Animated characters who have vertex flex capability (Hi hex6)*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetCurrentScene](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseFlex.GetCurrentScene) | handle GetCurrentScene() | 返回最早激活的场景实体的实例（如果存在的话） | | [GetSceneByIndex](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CBaseFlex.GetSceneByIndex) | handle GetSceneByIndex(int a) | 返回指定索引号的场景实体的实例 | |
|  |
| CSceneEntity |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***Choreographed scene which controls animation and/or dialog on one or more actors.*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [AddBroadcastTeamTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.AddBroadcastTeamTarget) | void AddBroadcastTeamTarget(int a) | 在记分板列表上增加一个组（使用索引号） | | [Cancel](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.Cancel) | void Cancel() | 取消场景重播 | | [EstimateLength](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.EstimateLength) | float EstimateLength() | 返回场景长度（秒） | | [FindCamera](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.FindCamera) | handle FindCamera() | 获取镜头 | | [FindNamedEntity](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.FindNamedEntity) | handle FindNamedEntity(string a) | 通过一个实体参照，比如!target，获取场景物体中的实际实体 | | [IsPaused](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.IsPaused) | bool IsPaused() | 场景是否被暂停 | | [IsPlayingBack](C:\\wiki\\Dota_2_Workshop_Tools\\Scripting\\API\\CSceneEntity.IsPlayingBack" \o "Dota 2 Workshop Tools/Scripting/API/CSceneEntity.IsPlayingBack) | bool IsPlayingBack() | 场景是否正播放 | | [LoadSceneFromString](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.LoadSceneFromString) | bool LoadSceneFromString(string a, string b) | 输入虚拟场景名和VCD字符串，载入场景 | | [RemoveBroadcastTeamTarget](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.RemoveBroadcastTeamTarget) | void RemoveBroadcastTeamTarget(int a) | 从记分板列表上移除一个组（使用索引号） | | [Start](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CSceneEntity.Start) | void Start(handle a) | 开始场景重播，参数为激活者实体（activatorEntity） | |
|  |
| GridNav |
| ***A class that can communicate with the gridnav, useful for seeing if stuff should be able to move*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GridPosToWorldCenterX](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.GridPosToWorldCenterX) | float GridPosToWorldCenterX(int a) | 获取指定X索引号的世界中心X位置（网格到世界） | | [GridPosToWorldCenterY](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.GridPosToWorldCenterY) | float GridPosToWorldCenterY(int a) | 获取指定Y索引号的世界中心Y位置（网格到世界） | | [IsBlocked](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.IsBlocked) | bool IsBlocked(Vector a) | 检查给定位置是否被阻挡 | | [IsNearbyTree](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.IsNearbyTree) | bool IsNearbyTree(Vector position, float radius, bool c) |  | | [IsTraversable](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.IsTraversable) | bool IsTraversable(Vector a) | 检查给予维持是否可通过 | | [RegrowAllTrees](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.RegrowAllTrees) | void RegrowAllTrees() |  | | [WorldToGridPosX](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.WorldToGridPosX) | int WorldToGridPosX(float a) | 获取指定世界X位置的X索引（世界到网格） | | [WorldToGridPosY](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\GridNav.WorldToGridPosY) | int WorldToGridPosY(float a) | 获取指定世界Y位置的Y索引（世界到网格） | |
|  |
| Convars |
| ***No Description Set*** |
|  |
| ***Global accessor variable:* ConVars** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [GetBool](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetBool) | table GetBool(string variableName) | GetBool(name) : 将指定的控制台变量（convar）作为boolean返回. | | [GetCommandClient](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetCommandClient) | handle GetCommandClient() | GetCommandClient() : 返回输入该控制台指令的玩家 | | [GetDOTACommandClient](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetDOTACommandClient) | handle GetDOTACommandClient() | GetDOTACommandClient() : 返回输入该控制台智力的DOTA玩家 | | [GetFloat](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetFloat) | table GetFloat(string name) | GetFloat(name) : 将指定的控制台变量（convar）作为float返回. 不存在时返回nil | | [GetInt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetInt) | table GetInt(string a) | GetInt(name) : 将指定的控制台变量（convar）作为int返回. 不存在时返回nil | | [GetStr](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.GetStr) | table GetStr(string variableName) | GetStr(name) : 将指定的控制台变量（convar）作为string返回. 不存在时返回nil | | [RegisterCommand](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.RegisterCommand) | void RegisterCommand(string variableName, handle function, string helpText, int flags) | RegisterCommand(name, fn, helpString, flags) : 注册一个控制台指令. | | [RegisterConvar](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.RegisterConvar) | void RegisterConvar(string name, string defaultValue, string helpText, int flags) | RegisterConvar(name, defaultValue, helpString, flags): 注册一个新的控制台变量. | | [SetBool](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.SetBool) | void SetBool(string variableName, bool value) | SetBool(name,val) : 将指定控制台变量（convar）设置为指定bool. | | [SetFloat](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.SetFloat) | void SetFloat(string variableName, float value) | SetFloat(name,val) : 将指定控制台变量（convar）设置为指定float. | | [SetInt](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.SetInt) | void SetInt(string a, int b) | SetInt(name,val) : 将指定控制台变量（convar）设置为指定int. | | [SetStr](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Convars.SetStr) | void SetStr(string a, string b) | SetStr(name,val) : 将指定控制台变量（convar）设置为指定string. | |
|  |
| Vector |
| ***Vector class*** |
|  |
| ***Global accessor variable:* Vector(x,y,z)** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [\_\_add](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._add) | Vector \_\_add(Vector a, Vector b) | 重载+.矢量加运算 | | [\_\_div](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._div) | Vector \_\_div(Vector a, Vector b) | 重载/.矢量除运算 | | [\_\_eq](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._eq) | bool \_\_eq(Vector a, Vector b) | 重载==.矢量比较运算 | | [\_\_len](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._len) | float \_\_len() | 重载#返回矢量长度 | | [\_\_mul](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._mul) | Vector \_\_mul(Vector a, Vector b) | 重载\*返回矢量乘运算 | | [\_\_sub](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._sub) | Vector \_\_sub(Vector a, Vector b) | 重载-.矢量减运算 | | [\_\_tostring](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._tostring) | string \_\_tostring() | 重载..转换矢量为字符串 | | [\_\_unm](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector._unm) | Vector \_\_unm() | 重载- operator | | [Cross](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector.Cross) | Vector Cross(Vector a, Vector b) | 矢量叉积 | | [Dot](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector.Dot) | float Dot(Vector a, Vector b) | 矢量点积 | | [Length](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector.Length) | float Length() | 矢量长度（模） | | [Length2D](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector.Length2D) | float Length2D() | 矢量XY平面上长度（模） | | [Normalized](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\Vector.Normalized) | Vector Normalized() | 返回单位矢量 | |
|  |
| CDOTA\_MapTree |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***A tree on the map*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [CutDown](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_MapTree.CutDown) | void CutDown(int TeamNumberKnownTo ) | Cuts down this tree. | | [CutDownRegrowAfter](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_MapTree.CutDownRegrowAfter) | void CutDownRegrowAfter(float RegrowAfter, int TeamNumberKnownTo ) | Cuts down this tree. Regrow it after some time | | [GrowBack](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_MapTree.GrowBack) | void GrowBack() | Grows back the tree if it was cut down. | | [IsStanding](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_MapTree.IsStanding) | bool IsStanding() | Returns true if the tree is standing, false if it has been cut down | |
|  |
| CDOTA\_SimpleObstruction |
| **extends** [**CBaseEntity**](#CBaseEntity) |
| ***An obstruction on the map that effects the gridnav*** |
| |  |  |  | | --- | --- | --- | | **函数名** | **函数原型** | **解释** | | [IsEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_SimpleObstruction.IsEnabled) | bool IsEnabled() | Returns whether the obstruction is currently active | | [SetEnabled](file:///C:\wiki\Dota_2_Workshop_Tools\Scripting\API\CDOTA_SimpleObstruction.SetEnabled) | void SetEnabled(bool enabled, bool b) | Enable or disable the obstruction |   **Retrieved from "**[**https://developer.valvesoftware.com/w/index.php?title=Dota\_2\_Workshop\_Tools:zh-cn/Scripting:zh-cn/API:zh-cn&oldid=188287**](https://developer.valvesoftware.com/w/index.php?title=Dota_2_Workshop_Tools:zh-cn/Scripting:zh-cn/API:zh-cn&oldid=188287)**"** |
| **[Categories](C:\\wiki\\Special:Categories" \o "Special:Categories):** |
| * **[Dota 2 创意工坊工具集](C:\\wiki\\Category:Dota_2_%E5%88%9B%E6%84%8F%E5%B7%A5%E5%9D%8A%E5%B7%A5%E5%85%B7%E9%9B%86" \o "Category:Dota 2 创意工坊工具集)** |
| * [**AMHC 汉化**](file:///C:\wiki\Category:AMHC_%25E6%25B1%2589%25E5%258C%2596) |
| Navigation menu |
| Personal tools |
| * [**Create account**](file:///C:\w\index.php%3ftitle=Special:UserLogin&returnto=Dota+2+Workshop+Tools:zh-cn\Scripting:zh-cn\API:zh-cn&type=signup) |
| * [**Log in**](file:///C:\w\index.php%3ftitle=Special:UserLogin&returnto=Dota+2+Workshop+Tools:zh-cn\Scripting:zh-cn\API:zh-cn) |
| Namespaces |
| * [**Page**](file:///C:\wiki\Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn) |
| * [**Discussion**](file:///C:\w\index.php%3ftitle=Talk:Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&action=edit&redlink=1) |
| Variants |
| Views |
| * [**View**](file:///C:\wiki\Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn) |
| * [**View source**](file:///C:\w\index.php%3ftitle=Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&action=edit) |
| * [**History**](file:///C:\w\index.php%3ftitle=Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&action=history) |
| Actions |
| Search |
| 窗体顶端 |
|  |
| 窗体底端 |
| Navigation |
| * [**Main Page**](file:///C:\wiki\Main_Page) |
| * [**Source SDK index**](file:///C:\wiki\SDK_Docs) |
| * [**Recent changes**](file:///C:\wiki\Special:RecentChanges) |
| * [**Random page**](file:///C:\wiki\Special:Random) |
| Support |
| * [**Getting help**](file:///C:\wiki\Help:Contents) |
| * [**Source SDK FAQ**](file:///C:\wiki\Category:Source_SDK_FAQ) |
| * [**Level Design FAQ**](file:///C:\wiki\Category:Level_Design_FAQ) |
| * [**SDK Help Forums**](http://forums.steampowered.com/forums/forumdisplay.php?f=191) |
| Steam Community |
| * [**Source SDK Hub**](http://steamcommunity.com/app/211) |
| * [**Steam Games**](http://www.steampowered.com) |
| Tools |
| * [**What links here**](file:///C:\wiki\Special:WhatLinksHere\Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn) |
| * [**Related changes**](file:///C:\wiki\Special:RecentChangesLinked\Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn) |
| * [**Special pages**](file:///C:\wiki\Special:SpecialPages) |
| * [**Printable version**](file:///C:\w\index.php%3ftitle=Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&printable=yes) |
| * [**Permanent link**](file:///C:\w\index.php%3ftitle=Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&oldid=188287) |
| * [**Page information**](file:///C:\w\index.php%3ftitle=Dota_2_Workshop_Tools:zh-cn\Scripting:zh-cn\API:zh-cn&action=info) |
| * **This page was last modified on 29 October 2014, at 16:27.** |
| * **This page has been accessed 895 times.** |
| * [**About Valve Developer Community**](file:///C:\wiki\Valve_Developer_Community:About) |
| * [**Terms of Use**](file:///C:\wiki\Valve_Developer_Community:Terms_of_use) |
| * [**Third Party Legal Notices**](file:///C:\wiki\Valve_Developer_Community:Third_Party_Legal_Notices) |
|  |
|  |