```
declare const __filename: string
declare const __dirname: string

declare function require(filePath: string): any;

declare interface TMissionLinkItem {

}

//fileStream 打开模式
declare const fmCreate: number
declare const fmOpenRead: number
declare const fmOpenWrite: number
declare const fmOpenReadWrite: number
declare const fmExclusive: number
declare const fmShareCompat: number
declare const fmShareExclusive: number
declare const fmShareDenyWrite: number
declare const fmShareDenyRead: number
declare const fmShareDenyNone: number


type TStoreItemList = any

//移动客户端功能按钮配置
declare interface IMobileFunctionButton {
      Tag: number
      IconFile: string
      ScriptFlag: string
      ShowDock: boolean
}

declare class TArgs {
      Int: number[]
      Str: string[]
      Count: number
      DataString: string
      Clear(): void;
      Add(str: string | number): void;
      constructor();
      ParseParams(s: string): void;
}

declare function CreateTArgs(params: string): TArgs;

declare function random(range: number): number;

declare function randomRange(min: number, max: number): number;

declare function format(str: string, param: any[]): string

declare function Debug(str: string): void;

declare function Trunc(value: number): number;

declare function ConvertDataSetToArray(dataSet: TDataSet, timeMode: "unixtime" |
"date"): any[]

declare enum TItemWhere {
      wDress, wWeapon, wRightHand, wNecklace, wHelmet, wArmringL, wArmringR,
wRingL, wRingR, wBujuk, wBelt, wBoots, wCharm, wMask, wFashion, wMount, wShield,
      wJewelry1, wJewelry2, wJewelry3, wJewelry4, wJewelry5, wJewelry6, //首饰盒
      wZodiac1, wZodiac2, wZodiac3, wZodiac4, wZodiac5, wZodiac6, wZodiac7,
wZodiac8, wZodiac9, wZodiac10, wZodiac11, wZodiac12
```

```
}

declare enum TAttackMode { amAll, amPeaceful, amDear, amMaster, amGroup,
amGuild, amRed, amNation, amCamp }

declare enum TReAliveEvent { aeRing, aeMagic, aeBuff, aeCmd, aeDeathTimeOut,
aeDoorAutoFix, aeScript, aeMonAutoAlive }

declare enum TFunctionFlag { ffJewelryBox, ffZodiac }


//键盘键值
declare enum TKeyBoardCode {
      // 通用按钮
      vkLButton = 0x01,  //   1
      vkRButton = 0x02,  //   2
      vkCancel = 0x03,  //   3
      vkMButton = 0x04,  //   4
      vkXButton1 = 0x05,  //   5
      vkXButton2 = 0x06,  //   6
      vkBack = 0x08,  //   8
      vkTab = 0x09,  //   9
      vkLineFeed = 0x0A,  //  10
      vkClear = 0x0C,  //  12
      vkReturn = 0x0D,  //  13
      vkShift = 0x10,  //  16
      vkControl = 0x11,  //  17
      vkMenu = 0x12,  //  18
      vkPause = 0x13,  //  19
      vkCapital = 0x14,  //  20
      vkKana = 0x15,  //  21
      vkHangul = 0x15,  //  21
      vkJunja = 0x17,  //  23
      vkFinal = 0x18,  //  24
      vkHanja = 0x19,  //  25
      vkKanji = 0x19,  //  25
      vkConvert = 0x1C,  //  28
      vkNonConvert = 0x1D,  //  29
      vkAccept = 0x1E,  //  30
      vkModeChange = 0x1F,  //  31
      vkEscape = 0x1B,  //  27
      vkSpace = 0x20,  //  32
      vkPrior = 0x21,  //  33
      vkNext = 0x22,  //  34
      vkEnd = 0x23,  //  35
      vkHome = 0x24,  //  36
      vkLeft = 0x25,  //  37
      vkUp = 0x26,  //  38
      vkRight = 0x27,  //  39
      vkDown = 0x28,  //  40
      vkSelect = 0x29,  //  41
      vkPrint = 0x2A,  //  42
      vkExecute = 0x2B,  //  43
      vkSnapshot = 0x2C,  //  44
      vkInsert = 0x2D,  //  45
      vkDelete = 0x2E,  //  46
      vkHelp = 0x2F,  //  47
      // 0-9
      vk0 = 0x30,  //  48
      vk1 = 0x31,  //  49
      vk2 = 0x32,  //  50
      vk3 = 0x33,  //  51
      vk4 = 0x34,  //  52
      vk5 = 0x35,  //  53
```

```
vk6 = 0x36,  //  54
vk7 = 0x37,  //  55
vk8 = 0x38,  //  56
vk9 = 0x39,  //  57
// A-Z
vkA = 0x41,  //  65
vkB = 0x42,  //  66
vkC = 0x43,  //  67
vkD = 0x44,  //  68
vkE = 0x45,  //  69
vkF = 0x46,  //  70
vkG = 0x47,  //  71
vkH = 0x48,  //  72
vkI = 0x49,  //  73
vkJ = 0x4A,  //  74
vkK = 0x4B,  //  75
vkL = 0x4C,  //  76
vkM = 0x4D,  //  77
vkN = 0x4E,  //  78
vkO = 0x4F,  //  79
vkP = 0x50,  //  80
vkQ = 0x51,  //  81
vkR = 0x52,  //  82
vkS = 0x53,  //  83
vkT = 0x54,  //  84
vkU = 0x55,  //  85
vkV = 0x56,  //  86
vkW = 0x57,  //  87
vkX = 0x58,  //  88
vkY = 0x59,  //  89
vkZ = 0x5A,  //  90

//特定按键
vkLWin = 0x5B,  //  91
vkRWin = 0x5C,  //  92
vkApps = 0x5D,  //  93
vkSleep = 0x5F,  //  95

//小键盘
vkNumpad0 = 0x60,  //  96
vkNumpad1 = 0x61,  //  97
vkNumpad2 = 0x62,  //  98
vkNumpad3 = 0x63,  //  99
vkNumpad4 = 0x64,  // 100
vkNumpad5 = 0x65,  // 101
vkNumpad6 = 0x66,  // 102
vkNumpad7 = 0x67,  // 103
vkNumpad8 = 0x68,  // 104
vkNumpad9 = 0x69,  // 105
vkMultiply = 0x6A,  // 106
vkAdd = 0x6B,  // 107
vkSeparator = 0x6C,  // 108
vkSubtract = 0x6D,  // 109
vkDecimal = 0x6E,  // 110
vkDivide = 0x6F,  // 111

//F1-F24
vkF1 = 0x70,  // 112
vkF2 = 0x71,  // 113
vkF3 = 0x72,  // 114
vkF4 = 0x73,  // 115
vkF5 = 0x74,  // 116
vkF6 = 0x75,  // 117
vkF7 = 0x76,  // 118
```

```
	vkF8 = 0x77,  // 119
	vkF9 = 0x78,  // 120
	vkF10 = 0x79,  // 121
	vkF11 = 0x7A,  // 122
	vkF12 = 0x7B,  // 123
	vkF13 = 0x7C,  // 124
	vkF14 = 0x7D,  // 125
	vkF15 = 0x7E,  // 126
	vkF16 = 0x7F,  // 127
	vkF17 = 0x80,  // 128
	vkF18 = 0x81,  // 129
	vkF19 = 0x82,  // 130
	vkF20 = 0x83,  // 131
	vkF21 = 0x84,  // 132
	vkF22 = 0x85,  // 133
	vkF23 = 0x86,  // 134
	vkF24 = 0x87,  // 135
}

//鼠标按键定义
declare enum TMouseButton {
	mbLeft,
	mbRight,
	mbMiddle
}

//客户端飘血类型
declare enum TBleedType {
	btNone = 0, //未知
	btNormal = 1, //普通
	btCritical = 2, //会心一击(无视防御) + 会心额外伤害
	btPunchHit = 3, //致命一击(双倍伤害 + 致命额外伤害)
	btMapDamageEvent = 4, //地图伤害事件
	btDamageRebound = 5,//伤害反弹
	btSpell = 6, //施法
	btMapRecovery = 7, //地图自动恢复
	btPoison = 8, //中毒
	btLevelChange = 9,//等级变更
	btUseMedicine = 10, //使用药品
	btAutoRecovery = 11, //自动恢复 站立休息一定时间会恢复
	btHealing = 12, //治愈术
	btHongMoSuit = 13, //虹魔套装 吸血
	btRunTired = 14, //跑步掉血
	btHealthStone = 15, //血魔石增加
	btHunger = 16, //饥饿系统
	btMagicProtected = 17,//护身戒指减伤
}

//飘血伤害类型
declare enum TDamageValueType {
	dvtHP = 0,
	dvtMP = 1
}

declare enum TBuffStatusType {
	stNone,
	stStone, // 石化
	stDenyMove, // 禁止移动 行走 和 跑步都不允许
	stDenyWalk, // 禁止行走
	stDenyRun, // 禁止跑步
	stDenyHorseRun, //禁止骑马
	stSuperMan, // 无敌
	stSuperManForMon, // 无敌只对怪物
	stRelive, //复活
```

```
	stUnRelive, //防复活
	stObserver, // 隐身
	stObserverForMon, // 隐身只对怪物
	stDenyUseMagic, //禁止使用技能
	stDenyUseItem, //禁止使用物品
	stUnParalysis, //防麻痹
	stUnLockRun, //防蛛网
	stUnTrap, //防陷阱
	stStruckAction, //强制后仰
	stNoStruckAction//强制无后仰
}

//间隔作用的Buff类型。
declare enum TBuffIntervalType {
	biNone,
	biCustom,
	biHP,
	biMP,
	biExp
}

//属性类型buff
declare enum TBuffAbilityType {
	atNone,
	atMaxHP,
	atMaxMP,
	atDC,
	atDCMax,
	atMC,
	atMCMax,
	atSC,
	atSCMax,
	atTC,
	atTCMax,
	atPC,
	atPCMax,
	atWC,
	atWCMax,
	atAC,
	atACMax,
	atMAC,
	atMacMax,
	atHitPoint,
	atHitSpeed,
	atSpeedPoint,
	atLucky,
	atHealthRecover,
	atSpellRecover,
	atAntiMagic,
	atAntiPoison,
	atPoisonRecover,
	atAbsorbing,
	atRebound,
	atAttackAdd,
	atPunchHit,
	atCriticalHit,
	atPunchHitDef,
	atCriticalHitDef
}

//货币类型
declare enum TMoneyType {
	mtNone,
	mtGold, //金币
```

```
	mtGameGold, //元宝
	mtGamePoint,//游戏点
	mtGameGird,//灵符
	mtGameDiamond,//金刚石
	mtPaoDian,//泡点值
	mtGlory //荣耀点
}

//拾取结果
declare enum TPickUpResult {
	purPickUp, //继续拾取
	purPickStop, //禁止拾取
	purPickStopDelete, //禁止拾取并删除
}

declare function ReplaceStr(srcStr: string, oldPartten: string, newPartten:
string | number): string

declare function max(a: number, b: number): number

//比较字符串返回差异值
declare function CompareStr(a: string, b: string): number

declare function Randomize(): void;
declare function FloatToStr(value: number): string;

//带下划线的表示兼容层 不建议据需使用
declare function _Copy(str: string, startIndex: number, count: number): string

declare function _Pos(subStr: string, srcStr: string): number

declare function CharCodeIn(src: string, low: string, high: string): number

declare interface TimeDef {
	year: number,
	month: number,
	day: number,
	hour: number,
	minute: number,
	second: number,
	msecond: number
}


declare type QueryDBAsyncResult = (error: string, dataset: TDataSet) => void
declare type ExecDBAsyncResult = (error: string, affectRow: number) => void
declare type HttpAsyncResult = (error: string, result: string) => void


declare function RegisterMonster(race: number, clazz: typeof TMonster): boolean;

declare function GetTickCount(): number;

declare function ExtractStrings(spliteChar: string[], skipFirstChar: string[],
Value: string, List: TStringList): void;




declare class TActor {
	GetRaceType(): number;
	readonly RaceType: number;
	GetMapName(): string;
```

```
readonly MapName: string;
GetMap(): TEnvirnoment;
readonly Map: TEnvirnoment;
GetName(): string;
Name: string;
SetName(Value: string): void;
GetMapY(): number;
readonly MapY: number;
GetMapX(): number;
readonly MapX: number;
GetGender(): number;
Gender: number;
SetGender(Value: number): void;
GetDirection(): number;
Direction: number;
SetDirection(Value: number): void;
GetHair(): number;
Hair: number;
SetHair(Value: number): void;
GetJob(): number;
Job: number;
SetJob(Value: number): void;
GetNation(): number;
Nation: number;
SetNation(Value: number): void;
GetCamp(): number;
Camp: number;
SetCamp(Value: number): void;
GetGold(): number;
Gold: number;
SetGold(Value: number): void;
GetMaxGold(): number;
readonly MaxGold: number;
GoldChanged(): void;
GetPkPoint(): number;
PkPoint: number;
SetPkPoint(Value: number): void;
GetPkLevel(): number;
readonly PkLevel: number;
GetGhost(): boolean;
readonly Ghost: boolean;
GetDeath(): boolean;
readonly Death: boolean;
GetHasHorse(): boolean;
readonly HasHorse: boolean;
GetHorseType(): number;
HorseType: number;
SetHorseType(Value: number): void;
GetOnHorse(): boolean;
OnHorse: boolean;
SetOnHorse(Value: boolean): void;
GetGuild(): TGuild;
readonly Guild: TGuild;
GetCastle(): TUserCastle;
readonly Castle: TUserCastle;
GetHasGuild(): boolean;
readonly HasGuild: boolean;
GetGuildName(): string;
readonly GuildName: string;
GetGuildRankName(): string;
readonly GuildRankName: string;
GetIsGuildMaster(): boolean;
readonly IsGuildMaster: boolean;
GetISCastleMaster(): boolean;
```

```
readonly ISCastleMaster: boolean;
GetISCastleGuild(): boolean;
readonly ISCastleGuild: boolean;
GetInSafeZone(): boolean;
readonly InSafeZone: boolean;
GetCheck(Index: number): boolean;
SetCheck(Index: number, Value: boolean): void;
GetSlaveAllCount(): number;
readonly SlaveCount: number;
GetSlave(Index: number): TActor;
GetSlaveLevel(): number;
SlaveLevel: number;
SetSlaveLevel(Value: number): void;
GetItemCount(ItemName: string): number;
GetPoseCreate(): TActor;
readonly PoseCreate: TActor;
GetHP(): number;
HP: number;
SetHP(Value: number): void;
GetMaxHP(): number;
MaxHP: number;
SetMaxHP(Value: number): void;
GetMP(): number;
MP: number;
SetMP(Value: number): void;
GetMaxMP(): number;
MaxMP: number;
SetMaxMP(Value: number): number;
GetDCMin(): number;
DCMin: number;
SetDCMin(Value: number): void;
GetDCMax(): number;
DCMax: number;
SetDCMax(Value: number): void;
GetMCMin(): number;
MCMin: number;
SetMCMin(Value: number): void;
GetMCMax(): number;
MCMax: number;
SetMCMax(Value: number): void;
GetSCMin(): number;
SCMin: number;
SetSCMin(Value: number): void;
GetSCMax(): number;
SCMax: number;
SetSCMax(Value: number): void;
GetTCMin(): number;
TCMin: number;
SetTCMin(Value: number): void;
GetTCMax(): number;
TCMax: number;
SetTCMax(Value: number): void;
GetPCMin(): number;
PCMin: number;
SetPCMin(Value: number): void;
GetPCMax(): number;
PCMax: number;
SetPCMax(Value: number): void;
GetWCMin(): number;
WCMin: number;
SetWCMin(Value: number): void;
GetWCMax(): number;
WCMax: number;
SetWCMax(Value: number): void;
```

```
GetACMin(): number;
ACMin: number;
SetACMin(Value: number): void;
GetACMax(): number;
ACMax: number;
SetACMax(Value: number): void;
GetMACMin(): number;
MACMin: number;
SetMACMin(Value: number): void;
GetMACMax(): number;
MACMax: number;
SetMACMax(Value: number): void;
GetSpeedPoint(): number;
SpeedPoint: number;
SetSpeedPoint(Value: number): void;
GetHitPoint(): number;
HitPoint: number;
SetHitPoint(Value: number): void;
GetHitSpeed(): number;
readonly HitSpeed: number;
GetWalkSpeed(): number;
WalkSpeed: number;
SetWalkSpeed(Value: number): void;
GetWalkStep(): number;
WalkStep: number;
SetWalkStep(Value: number): void;
GetWalkWait(): number;
WalkWait: number;
SetWalkWait(Value: number): void;
GetAttackSpeed(): number;
AttackSpeed: number;
SetAttackSpeed(Value: number): void;
GetLuck(): number;
readonly Luck: number;
GetLevel(): number;
Level: number;
SetLevel(Value: number): void;
GetExpHitter(): TActor;
ExpHitter: TActor;
SetExpHitter(Value: TActor): void;
LastHitter(): TActor;
SetLastHitter(Value: TActor): void;
GetTargetActor(): TActor;
TargetActor: TActor;
SetTargetActor(Value: TActor): void;
SetTargetActorEx(Value: TActor, LockTime: number): void;
GetMaxBagSize(): number;
MaxBagSize: number;
SetMaxBagSize(Value: number): void;
GetItemSize(): number;
readonly ItemSize: number;
GetBagItem(Index: number): TUserItem;
GetArmItem(AItemWhere: number): TUserItem;
readonly ArmItem: TUserItem;
GetDress(): TUserItem;
readonly Dress: TUserItem;
GetWepon(): TUserItem;
readonly Wepon: TUserItem;
GetWeapon(): TUserItem;
readonly Weapon: TUserItem;
GetNeckLace(): TUserItem;
readonly NeckLace: TUserItem;
GetZodiacs(index: number): TUserItem;
readonly Zodiacs: TUserItem;
```

```
GetJewelrys(index: number): TUserItem;
readonly Jewelrys: TUserItem;
GetHelmet(): TUserItem;
readonly Helmet: TUserItem;
GetRightHand(): TUserItem;
readonly RightHand: TUserItem;
GetArmringLeft(): TUserItem;
readonly ArmringLeft: TUserItem;
GetArmringRight(): TUserItem;
readonly ArmringRight: TUserItem;
GetRingLeft(): TUserItem;
readonly RingLeft: TUserItem;
GetRingRight(): TUserItem;
readonly RingRight: TUserItem;
GetBelt(): TUserItem;
readonly Belt: TUserItem;
GetBoots(): TUserItem;
readonly Boots: TUserItem;
GetCharm(): TUserItem;
readonly Charm: TUserItem;
GetMask(): TUserItem;
readonly Mask: TUserItem;
GetBujuk(): TUserItem;
readonly Bujuk: TUserItem;
GetFashion(): TUserItem;
readonly Fashion: TUserItem;
GetMount(): TUserItem;
readonly Mount: TUserItem;
GetShield(): TUserItem;
readonly Shield: TUserItem;
ClearSkill(): void;
DelNoJobSkill(): void;
DelSkill(ASkillName: string): void;
AddSkill: (ASkillName: string, SkillLevel?: number) => void;
FindSkill(ASkillName: string): TUserMagic;
ChangeSkillLevel(ASkillName: string, SkillLevel: number): void;
GetExp(): number;
Exp: number;
SetExp(Exp: number): void;
GetMaxExp(): number;
readonly MaxExp: number;
AddExp(Value: number): void;
GetWeight(): number;
readonly Weight: number;
GetMaxWeight(): number;
readonly MaxWeight: number;
GetWearWeight(): number;
readonly WearWeight: number;
GetMaxWearWeight(): number;
readonly MaxWearWeight: number;
GetHandWeight(): number;
readonly HandWeight: number;
GetMaxHandWeight(): number;
readonly MaxHandWeight: number;
CheckCanTakeOnItem(AUserItem: TUserItem, ItemWhere: number): boolean;
CheckCanTakeOffItem(ItemWhere: number): boolean;
TakeOnItem(AUserItem: TUserItem, ItemWhere: number): void;
TakeOffItem(nItemWhere: number): void;
AddItemToBag(Item: TUserItem): boolean;
Die(): void;
GoDie(ExpOwner: TActor, ItemOwner: TActor): void;
MakeGhost(): void;
IsPlayer(): boolean;
IsNPC(): boolean;
```

```
IsMonNPC(): boolean;
GetIsUnknowActor(): boolean;
IsUnknowActor: boolean;
SetIsUnknowActor(Value: boolean): void;
GetIsAdminMode(): boolean;
IsAdminMode: boolean;
SetIsAdminMode(Value: boolean): void;
GetSuperManMode(): boolean;
SuperManMode: boolean;
SetSuperManMode(Value: boolean): void;
GetObserverMode(): boolean;
ObserverMode: boolean;
SetObserverMode(Value: boolean): void;
GetPermission(): number;
Permission: number;
SetPermission(Value: number): void;
GetAttackMode(): number;
AttackMode: number;
SetAttackMode(Value: number): void;
DeleteUseItem(NWhere: number): void;
RepairItem(NWhere: number): void;
RepairAll(): void;
GetNameColor(): number;
NameColor: number;
SetNameColor(Value: number): void;
KillSlave: (ASlaveName: string, ACount?: number) => void;
GetSideSlaveCount(Range: number, ASlaveName: string): number;
readonly SideSlaveCount: number;
GetSlaveCount(ASlaveName: string): number;
GetKillMonExpRate(): number;
KillMonExpRate: number;
SetKillMonExpRate(Value: number): void;
ChangeKillMonExpRate(ARate: number, ATime: number): void;
GetPowerRate(): number;
readonly PowerRate: number;
ChangePowerRate(ARate: number, ATime: number): void;
GetIsAdmin(): boolean;
readonly IsAdmin: boolean;
GetISSysOp(): boolean;
readonly ISSysOp: boolean;
Kill: (AMode?: number) => void;
GetAlwaysShowHP(): boolean;
AlwaysShowHP: boolean;
SetAlwaysShowHP(Value: boolean): void;
RecalcAbilitys(): void;
GetMaster(): TActor;
Master: TActor;
SetMaster(Master: TActor): void;
GetMasterRoyaltyTick(): number;
MasterRoyaltyTick: number;
SetMasterRoyaltyTick(Value: number): void;
GetProtect(): boolean;
Protect: boolean;
SetProtect(Value: boolean): void;
GetTotalAbility(): number;
readonly TotalAbility: number;
GetDropName(): string;
DropName: string;
SetDropName(Value: string): void;
GetNoDropItem(): boolean;
NoDropItem: boolean;
SetNoDropItem(Value: boolean): void;
GetNoDropUseItem(): boolean;
NoDropUseItem: boolean;
```

```
      SetNoDropUseItem(Value: boolean): void;
      GetData(): Object;
      Data: Object;
      SetData(Value: Object): void;
      SearchViewRange(): void;
      ClearViewRange(): void;
      MagicAttackXY: (ATarget: TActor, ATargetX: number, ATargetY: number,
AMagic: number, ClientPlayerHaveSpellAction?: boolean) => void;
      MagicAttackXYEx: (ATarget: TActor, ATargetX: number, ATargetY: number,
AMagic: TUserMagic, ClientPlayerHaveSpellAction?: boolean) => void;
      MagicAttack: (ATarget: TActor, AMagic: number,
ClientPlayerHaveSpellAction?: boolean) => void;
      MagicAttackEx: (ATarget: TActor, AMagic: TUserMagic,
ClientPlayerHaveSpellAction?: boolean) => void;
      MagicAttackGroup: (ATarget: TActor, ATargetX: number, ATargetY: number,
AMagic: number, Level: number, Group: number, ClientPlayerHaveSpellAction?:
boolean) => void;
      Damage: (ATarget: TActor, AValue: number, MagicID?: number, MagicLevel?:
number, ShowBleedNumber?: boolean) => void;
      DamageDelay: (ATarget: TActor, AValue: number, Delay: number, MagicID?:
number, MagicLevel?: number, ShowBleedNumber?: boolean) => void;
      CreateAttackEvent: (AMapX: number, AMapY: number, ATime: number,
AInterval: number, ADamage: number, SkillEffectID?: number, MagId?: number,
MagLv?: number) => boolean;
      CreateAttackEvent2: (AMapX: number, AMapY: number, ATime: number,
AInterval: number, ADamage: number, UIEffectID?: number, MagId?: number, MagLv?:
number) => boolean;
      MoveTo: (DestX: number, DestY: number, ARange?: number) => boolean;
      CheckState(AState: number): boolean;
      SetState(AState: number, ATime: number, AValue: number): void;
      BlockInArea(Left: number, Top: number, Right: number, Bottom: number,
Duration: number): void;
      BreakBlockArea(): void;
      Push(ADirection: number, AStep: number): number;
      AddGameLog1(Event: string, ItemName: string, Amount: number, Log: string):
void;
      AddGameLog2(Event: string, Log: string): void;
      AddGameLogEx(Event: number, Int1: number, Int2: number, Log1: string,
Log2: string): void;
      GetSkillCount(): number;
      readonly SkillCount: number;
      GetSkills(Index: number): TUserMagic;
      Move(sMapName: string, nX: number, nY: number): boolean;
      ChangeToMonster(sMonName: string, nSec: number): void;
      CancelToMonster(): void;
      AddShowName(sName: string): void;
      DelShowName(sName: string): void;
      HaveShowName(sName: string): boolean;
      SetDropItemRate(Value: number): void;
      GetDropItemRate(): number;
      DropItemRate: number;
      GetSVar(Index: number): string;
      SVar: string;
      GetNVar(Index: number): number;
      NVar: number;
      SetSVar(Index: number, S: string): void;
      SetNVar(Index: number, Value: number): void;
      SetBlendColor(Value: number): void;
      GetBlendColor(): number;
      BlendColor: number;
      SetWeaponBlendColor(Value: number): void;
      GetWeaponBlendColor(): number;
      WeaponBlendColor: number;
      SendMessage: (Msg: string, Kind?: number) => void;
```

```
      GetSlaveList(): TActorList;
      readonly SlaveList: TActorList;
      GetSlaveRelax(): boolean;
      SlaveRelax: boolean;
      SetSlaveRelax(Value: boolean): void;
      SetThroughMonster(Value: boolean): void;
      GetThroughMonster(): boolean;
      ThroughMonster: boolean;
      SetThroughHuman(Value: boolean): void;
      GetThroughHuman(): boolean;
      ThroughHuman: boolean;
      SetThroughNPC(Value: boolean): void;
      GetThroughNPC(): boolean;
      ThroughNPC: boolean;
      SetThroughGuard(Value: boolean): void;
      GetThroughGuard(): boolean;
      ThroughGuard: boolean;
      ResetThroughMonster(): void;
      ResetThroughHuman(): void;
      ResetThroughNPC(): void;
      ResetThroughGuard(): void;
      SetMaxDropHp(Value: number): void;
      GetMaxDropHp(): number;
      MaxDropHp: number;
      RefFeature(): void;
      RefUseItem(): void;
      GetCustomEffect(index: number): number;
      SetCustomEffect(index: number, Value: number): void;
      GetGameMoney(MoneyType: number): number;
      GameMoney: number;
      SetGameMoney(MoneyType: number, Value: number, Log: string): void;
      CanAddGameMoney(MoneyType: number, Value: number): boolean;
      CanTakeGameMoney(MoneyType: number, Value: number): boolean;
      AddGameMoney(MoneyType: number, Value: number, Log: string): boolean;
      TakeGameMoney(MoneyType: number, Value: number, Log: string): boolean;
      GetTitleEffect(): number;
      TitleEffect: number;
      SetTitleEffect(Value: number): void;
      IsFriendlyTarget(Target: TActor): boolean;
      IsFrindlyTarget(Target: TActor): boolean;
      IsAttackTarget(Target: TActor): boolean;
      AddAbilityBuff(GroupID: number, AbilityType: number, DuraMs: number,
AblityValue: number, ValueIsRate: boolean): TBuff;
      AddStatusBuff: (GroupID: number, StatusType: TBuffStatusType, DuraMs:
number, Args1: number, Args2: number) => TBuff;
      AddIntervalBuff(GroupID: number, IntervalType: number, DuraMs: number,
Interval: number, Args1: number, Args2: number): TBuff;
      SetBuffIcon(BuffHandle: number, ImageFileName: string, ImageIndex: number,
HoverImageIndex: number, ShowMessage: string, DisappearMessage: string,
HintMessage: string, FlashOnDisappear: boolean, ShowTimeStr: boolean): void;
      DeleteBuff(BuffHandle: number): boolean;
      DeleteBuffByGroupID(Min: number, Max: number): number;
      DeleteBuffByTag(Tag: number): number;
      GetBuffCount(): number;
      readonly BuffCount: number;
      GetBuffByIndex(Index: number): TBuff;
      readonly BuffByIndex: TBuff;
      AddBuffByID(ID: number, GroupID: number): TBuff;
      AddBuffByName(BuffName: string, GroupID: number): TBuff;
      GetDenyAutoAddHP(): boolean;
      DenyAutoAddHP: boolean;
      SetDenyAutoAddHP(Value: boolean): void;
      GetDenyAutoAddMP(): boolean;
      DenyAutoAddMP: boolean;
```

```
      SetDenyAutoAddMP(Value: boolean): void;
      GetHandle(): number;
      readonly Handle: number;
      GetHashCode(): number;
      readonly HashCode: number;
      ShowBleedNumberForDebug(ImgFile: string, NumImgIndex: number, Prefix:
string, BleedNumber: number, Suffix: string, DuraTime: number, StartX: number,
StartY: number, StopX: number, StopY: number): void;
      ShowBleedNumber: (TemplateID: number, BleedNumber: number, DelayMS?:
number) => void;
      GetAddedAbility(): TAddedAbility;
      readonly AddedAbility: TAddedAbility;
      GetTriggerAINpcExecute(): boolean;
      TriggerAINpcExecute: boolean;
      SetTriggerAINpcExecute(Value: boolean): void;
      GetTriggerSelectMagicBeforeAttack(): boolean;
      TriggerSelectMagicBeforeAttack: boolean;
      SetTriggerSelectMagicBeforeAttack(Value: boolean): void;
      Mutiny(): void;
      FlashMove(X: number, Y: number, IgnorActor: boolean): boolean;
      FlashPush(Direction: number, MaxStep: number, IgnorActor: boolean):
boolean;
      ShowEffectEx: (EffectID: number, OffsetX?: number, OffsetY?: number,
OthersCanSee?: boolean, LoopCount?: number, ClearIfDead?: boolean) => void;
      ShowEffectEx2: (EffectID: number, OffsetX?: number, OffsetY?: number,
OthersCanSee?: boolean, LoopCount?: number, ClearIfDead?: boolean) => void;
      GetTriggerRevivalEvent(): boolean;
      TriggerRevivalEvent: boolean;
      SetTriggerRevivalEvent(Value: boolean): void;
      GetTriggerDieEvent(): boolean;
      TriggerDieEvent: boolean;
      SetTriggerDieEvent(Value: boolean): void;
      GetTriggerKillerEvent(): boolean;
      TriggerKillerEvent: boolean;
      SetTriggerKillerEvent(Value: boolean): void;
      GetTriggerDamageEvent(): boolean;
      TriggerDamageEvent: boolean;
      SetTriggerDamageEvent(Value: boolean): void;
      GetRunInterval(): number;
      RunInterval: number;
      SetRunInterval(RunInterval: number): void;
      GetConfuseBitTimeBeAttacked(): boolean;
      ConfuseBitTimeBeAttacked: boolean;
      SetConfuseBitTimeBeAttacked(ConfuseBitTimeBeAttacked: boolean): void;
      GetAttackRange(): number;
      AttackRange: number;
      SetAttackRange(Value: number): void;
      GetLastUseMagicID(): number;
      readonly LastUseMagicID: number;
      GetLastUseMagicLevel(): number;
      readonly LastUseMagicLevel: number;
      GetViewRange(): number;
      ViewRange: number;
      SetViewRange(Value: number): void;
      MissionToXY(X: number, Y: number, Range: number): boolean;
      FlyEffect: (Target: TActor, SkillEffectID: number, Speed: number,
FlyOverEffect?: number) => void;
      PlayDir8Effect: (SkillEffectID: number, Dir?: number) => void;
      FlyEffectNoTarget(MapX: number, MapY: number, SkillEffectID: number,
Speed: number, FlyOverEffect: number): void;
      DelayGotoMS(ID: number, AIntervalMS: number, AChangeMapDelete: boolean,
Once: boolean): boolean;
      DelayGoto(ID: number, AIntervalTime: number, AChangeMapDelete: boolean,
Once: boolean): boolean;
```

```
      ClearDelayGoto(ID: number): void;
      HasDelayGoto(ID: number): boolean;
      DelayCallMethod(AMethod: string, AIntervalTime: number, AChangeMapDelete:
boolean): void;
      DeleteDelayCallMethod(AMethod: string): void;
      SetHudHPStr(HudHPStr: string): void;

}
declare class TPlayObject extends TActor {
      GetHomeMapName(): string;
      readonly HomeMapName: string;
      GetHomeX(): number;
      readonly HomeX: number;
      GetHomeY(): number;
      readonly HomeY: number;
      SendCustomMessage(AMessageToken: number, AParam1: number, AParam2: number,
AParam3: number, AData: string): void;
      SendGuildMessage(Msg: string): void;
      SendTopMessage(Message: string, Mode: number): void;
      SendCenterMessage: (Message: string, Mode: number, Time?: number) => void;
      SendCountDownMessage: (Message: string, Mode: number, Falg?: number,
ChangMapDelete?: boolean) => void;
      DeleteCountDownMessage(Falg: number): void;
      OpenURL(AUrl: string, Width: number, Height: number): void;
      RequestURL(AUrl: string): void;
      GetAccount(): string;
      readonly Account: string;
      GetIsNewHuman(): boolean;
      readonly IsNewHuman: boolean;
      GetLoginTime(): number;
      readonly LoginTime: number;
      GetSessionID(): number;
      readonly SessionID: number;
      GetDearName(): string;
      DearName: string;
      SetDearName(Name: string): void;
      CallMethod(Npc: TActor, Method: string): void;
      GetMarried(): boolean;
      readonly Married: boolean;
      GetMasterName(): string;
      MasterName: string;
      SetMasterName(Name: string): void;
      GetISMaster(): boolean;
      readonly ISMaster: boolean;
      GetMarryCount(): number;
      readonly MarryCount: number;
      GetReNewLevel(): number;
      ReNewLevel: number;
      SetReNewLevel(Value: number): void;
      GetBonusPoint(): number;
      BonusPoint: number;
      SetBonusPoint(Value: number): void;
      GetMemberType(): number;
      MemberType: number;
      SetMemberType(Value: number): void;
      GetMemberLevel(): number;
      MemberLevel: number;
      SetMemberLevel(Value: number): void;
      GetGameGold(): number;
      GameGold: number;
      SetGameGold(Value: number): void;
      GetPaoDianPoint(): number;
      PaoDianPoint: number;
      SetPaoDianPoint(Value: number): void;
```

```
      GetGamePoint(): number;
      GamePoint: number;
      SetGamePoint(Value: number): void;
      GetGameGlory(): number;
      GameGlory: number;
      SetGameGlory(Value: number): void;
      GetGameDiaMond(): number;
      GameDiaMond: number;
      SetGameDiaMond(Value: number): void;
      GetGameGird(): number;
      GameGird: number;
      SetGameGird(Value: number): void;
      GetCustomItemCount(): number;
      readonly CustomItemCount: number;
      GetCustomItem(Index: number): TUserItem;
      ShowEffect(EffectID: number): void;
      CheckTextList(ATextFile: string, AText: string): boolean;
      CheckNameList(ATextFile: string): boolean;
      CheckAccountList(ATextFile: string): boolean;
      ClearList(ATextFile: string): void;
      AddTextList(ATextFile: string, AText: string): void;
      AddNameList(ATextFile: string): void;
      AddAccountList(ATextFile: string): void;
      AddGuildList(ATextFile: string): void;
      AddIPList(ATextFile: string): void;
      DelTextList(ATextFile: string, AText: string): void;
      DelNameList(ATextFile: string): void;
      DelGuildList(ATextFile: string): void;
      DelAccountList(ATextFile: string): void;
      DelIPList(ATextFile: string): void;
      MapMove(MapName: string, MapX: number, MapY: number): void;
      MapMoveEx(Envir: TEnvirnoment, MapX: number, MapY: number): void;
      ForceMapMove(Envir: TEnvirnoment, MapX: number, MapY: number): void;
      RestBonusPoint(): void;
      TakeCastGold(AGold: number): void;
      AutoGetExp(AMap: string, ATime: number, APoint: number, IsSafeZone:
boolean): void;
      StopAutoGetExp(): void;
      OffLinePlay(AIntervalTime: number, AExpPoint: number): void;
      KickOffLine(): void;
      PlaySound(ASoundFile: string): void;
      PlayVideo(AVideoFile: string): void;
      Recallmob: (AMonName: string, MonLvl?: number, RoyaltySec?: number, X?:
number, Y?: number) => TActor;
      GroupMoveEx(Envir: TEnvirnoment, MapX: number, MapY: number, boForce:
boolean): void;
      GroupMapMove(MapName: string, MapX: number, MapY: number): void;
      GuildMapMoveEx(Envir: TEnvirnoment, MapX: number, MapY: number): void;
      GuildMapMove(MapName: string, MapX: number, MapY: number): void;
      RandomMove: (MapName?: string) => void;
      RandomMoveEx(AEnvir: TEnvirnoment): void;
      GetCreditPoint(): number;
      CreditPoint: number;
      SetCreditPoint(Value: number): void;
      StartAutoAddGameGold(APoint: number, AInterval: number): void;
      StopAutoAddGameGold(): void;
      StartAutoSubGameGold(APoint: number, AInterval: number): void;
      StopAutoSubGameGold(): void;
      ChangeReNewLevel(Value: number, NewLevel: number, BounsuPoint: number):
void;
      Kick(): void;
      ReAlive(): void;
      UpdateItem(Item: TUserItem): void;
      DeleteItem: (Item: TUserItem, Count?: number) => void;
```

```
      DeleteBagItem(Item: TUserItem, Count: number): number;
      UpdateMagicLvExp(Magic: TUserMagic): void;
      UpdateMagic(Magic: TUserMagic): void;
      SetMagicCDTime(MagicId: number, CDTime: number): void;
      GetLogonTick(): number;
      readonly LogonTick: number;
      GetIPAddress(): string;
      readonly IPAddress: string;
      GetIPLocal(): string;
      readonly IPLocal: string;
      GetTimeRecall(): boolean;
      TimeRecall: boolean;
      GetPayMent(): number;
      readonly PayMent: number;
      GetGroupOwner(): TPlayObject;
      readonly GroupOwner: TPlayObject;
      GetGroupCount(): number;
      readonly GroupCount: number;
      GetGroupMember(Index: number): TPlayObject;
      GetISGroupMaster(): boolean;
      readonly ISGroupMaster: boolean;
      JoinGroup(APlayer: TPlayObject, ShowDialog: boolean): void;
      LeaveGroup(): void;
      GetRankLevelName(): string;
      RankLevelName: string;
      SetRankLevelName(Value: string): void;
      GetTitleName(): string;
      TitleName: string;
      SetTitleName(Value: string): void;
      GetShowRankLevelName(): boolean;
      ShowRankLevelName: boolean;
      SetShowRankLevelName(Value: boolean): void;
      UpdateName(): void;
      GetContribution(): number;
      Contribution: number;
      SetContribution(Value: number): void;
      GetLocked(): boolean;
      Locked: boolean;
      SetLocked(Value: boolean): void;
      GetStallState(): number;
      readonly StallState: number;
      SetTimeRecall(Value: boolean, Minutes: number): void;
      SetTimeRecallEx(AMap: string, Minutes: number, MapX: number, MapY:
number): void;
      GetProperties(): number;
      readonly Properties: number;
      GetFeature(): number;
      readonly Feature: number;
      GetFeatureEx(): number;
      readonly FeatureEx: number;
      GetStoragePwd(): string;
      StoragePwd: string;
      SetStoragePwd(Value: string): void;
      GetAllowDeal(): boolean;
      AllowDeal: boolean;
      SetAllowDeal(Value: boolean): void;
      GetAllowGuild(): boolean;
      AllowGuild: boolean;
      SetAllowGuild(Value: boolean): void;
      GetAllowGroup(): boolean;
      AllowGroup: boolean;
      SetAllowGroup(Value: boolean): void;
      GetAllowGroupReCall(): boolean;
      AllowGroupReCall: boolean;
```

```
      SetAllowGroupReCall(Value: boolean): void;
      GetAllowGuildReCall(): boolean;
      AllowGuildReCall: boolean;
      SetAllowGuildReCall(Value: boolean): void;
      GetAllowReAlive(): boolean;
      AllowReAlive: boolean;
      SetAllowReAlive(Value: boolean): void;
      GetShowFashion(): boolean;
      ShowFashion: boolean;
      SetShowFashion(Value: boolean): void;
      GetMonDropItems(MonName: string): void;
      AddExtendButton: (Name: string, Hint: string, Command: string, ImageIndex:
number, X?: number, Y?: number) => void;
      AddTopExtendButton(Name: string, Hint: string, Command: string,
ImageIndex: number, X: number, Y: number): void;
      RemoveExtendButton(Name: string): void;
      OpenMarket(): void;
      OpenBag(): void;
      ReloadBag(): void;
      OpenShop(): void;
      OpenMailBox(): void;
      OpenMapView(): void;
      OpenRefineBox(): void;
      ShowSighIcon(MethodID: number, Hint: string): void;
      ShowSighIcon1(Method: string, Hint: string): void;
      Give: (ItemName: string, Count?: number, Upgrade?: boolean) => void;
      GiveItem: (ItemName: string, Upgrade?: boolean) => TUserItem;
      GiveItemByIndex: (ItemIndex: number, Upgrade?: boolean) => TUserItem;
      Say(Msg: string): void;
      SayEx(UIName: string, Msg: string): void;
      UpdateSay(UIName: string, Msg: string): void;
      CloseWindow(UIName: string): void;
      CloseAllWindow(): void;
      MessageBox(Msg: string): void;
      Question(Msg: string, YMethod: string, NMethod: string): void;
      OpenBox(BoxID: number): void;
      OpenBoxEx(Shape: number, BoxID: number): void;
      OpenShuffle(BoxID: number): void;
      ShowProgress(Caption: string, ATime: number, EventID: number, ActCancel:
boolean): boolean;
      ShowProgressEx(Caption: string, ATime: number, NPC: TNormNpc,
SucessFunction: string, FailFunction: string, ActCancel: boolean): boolean;
      CloseProgress(): void;
      GetAllowSendMessage(): boolean;
      AllowSendMessage: boolean;
      SetAllowSendMessage(Value: boolean): void;
      GetMailCount(): number;
      readonly MailCount: number;
      GetMailUnreadCount(): number;
      readonly MailUnreadCount: number;
      CheckItemSoulLevelUp(AItem: TUserItem): void;
      GetStorageItemsCount(): number;
      readonly StorageItemsCount: number;
      GetStorageItem(Index: number): TUserItem;
      AddItemToStorage(AItem: TUserItem): boolean;
      DeleteStorageItem1(Index: number): boolean;
      DeleteStorageItem2(AItem: TUserItem): boolean;
      TakebackStorageItem(Index: number): boolean;
      GetBigStorageItemsCount(): number;
      readonly BigStorageItemsCount: number;
      GetBigStorageItem(Index: number): TUserItem;
      AddItemToBigStorage(AItem: TUserItem): boolean;
      DeleteBigStorageItem1(Index: number): boolean;
      DeleteBigStorageItem2(AItem: TUserItem): boolean;
```

```
      TakebackBigStorageItem(Index: number): boolean;
      GetMachineCode(): string;
      readonly MachineCode: string;
      AddTitle(ATitle: string): boolean;
      RemoveTitle(ATitle: string): boolean;
      SetActiveTitle(ATitle: string): void;
      GetTitleCount(): number;
      readonly TitleCount: number;
      GetTitleItems(AIndex: number): THumTitle;
      readonly TitleItems: THumTitle;
      GetActiveTitle(): THumTitle;
      ActiveTitle: THumTitle;
      FlashWindow(): void;
      GetMissions(): TMissions;
      readonly Missions: TMissions;
      LockMoveItem(): void;
      UnLockMoveItem(): void;
      LockMoveItem1(ATime: number): void;
      PlayDice: (ATag: number, APoint1: number, APoint2: number, APoint3:
number, AutoCloseTime?: number) => void;
      AddSidebarButton(ACaption: string, AName: string): void;
      DeleteSidebarButton(AName: string): void;
      ClearSidebarButton(): void;
      SetChatText(AValue: string): void;
      GetFunctionState: (ff: TFunctionFlag) => boolean;
      SetFunctionState: (ff: TFunctionFlag, boState: boolean) => void;
      SendControlVisible(Name: string, Visible: boolean, Auto: boolean): void;
      GetTotalExpRate(): number;
      readonly TotalExpRate: number;
      GetTotalItemRate(): number;
      readonly TotalItemRate: number;
      GetTotalGoldRate(): number;
      readonly TotalGoldRate: number;
      LockClient(Caption: string, CallBack: string): void;
      UnLockClient(): void;
      SetChatColor(Color: number): void;
      GetChatColor(): number;
      ChatColor: number;
      setMaxExp(Value: number): void;
      UIGuide(UIName: string, Hint: string, GuideIdent: string): void;
      MoveItemToUIContainer(UniTag: number, Item: TUserItem, Count: number):
void;
      RemoveItemFromUIContainer(UniTag: number): void;
      GetPromotionFlag(): string;
      readonly PromotionFlag: string;
      GetSVar(index: number): string;
      SetSVar(Index: number, Value: string): void;
      GetNVar(index: number): number;
      SetNVar(Index: number, Value: number): void;
      GetPVar(index: number): number;
      SetPVar(Index: number, Value: number): void;
      VarString(AVarName: string): TVarValue;
      VarInteger(AVarName: string): TVarValue;
      VarDateTime(AVarName: string): TVarValue;
      VarFloat(AVarName: string): TVarValue;
      VarBoolean(AVarName: string): TVarValue;
      ClearVarTable(): void;
      RemoveVar(AName: string): void;
      GoHome(): void;
      RecallHuman(AHumName: string): void;
      SetClientUIProperty(UIName: string, UIProperty: string): void;
      LoadUI(UI: string): void;
      GetPlayerID(): number;
      readonly PlayerID: number;
```

```
	GetCustomSaveStr(): string;
	CustomSaveStr: string;
	SetCustomSaveStr(Value: string): void;
	StartAutoFight(): void;
	StopAutoFight(): void;
	UpdateAssistantIni(IniText: string): void;
	UpdateItemContainerVisable(UniTags: string, Visable: boolean): void;
	SendBeedNumberForDebug(BleedActor: TActor, ImgFile: string, NumImgIndex:
number, Prefix: string, BleedNumber: number, Suffix: string, DuraTime: number,
StartX: number, StartY: number, StopX: number, StopY: number): void;
	SendBleedNumber: (BleedActor: TActor, TemplateID: number, BleedNumber:
number, DelayMS?: number) => void
	UpdateUIEditText(UniTag: number, Text: string): void;
	GetServerEntryFlag(): string;
	readonly ServerEntryFlag: string;
	HasRobotRun(Name: string): boolean;
	GetClientScreenWidth(): number;
	readonly ClientScreenWidth: number;
	GetClientScreenHeight(): number;
	readonly ClientScreenHeight: number;
	GetNotOnlineAddExp(): boolean;
	readonly NotOnlineAddExp: boolean;
	GetStationTime(): number;
	readonly StationTime: number;
	GetStationTick(): number;
	readonly StationTick: number;
	UpdateMagicOpenState(Magic: TUserMagic): void;
	GetItemFliterShowName(Idx: number): boolean;
	ItemFliterShowName: boolean;
	SetItemFliterShowName(Idx: number, Value: boolean): void;
	GetItemFliterAutoPickUp(Idx: number): boolean;
	ItemFliterAutoPickUp: boolean;
	SetItemFliterAutoPickUp(Idx: number, Value: boolean): void;
	GetItemFliterSpecialShow(Idx: number): boolean;
	ItemFliterSpecialShow: boolean;
	SetItemFliterSpecialShow(Idx: number, Value: boolean): void;
	UpdateItemFliter(): void;
	GetIsH5Client(): boolean;
	readonly IsH5Client: boolean;
	SetClientMagicUseTime(Magic: TUserMagic, Value: number): void;
	SetClientMagicOffsetUseTime(Magic: TUserMagic, Value: number): void;
	QueryClientQuickBarItemCount(Ident: string): void;
	V: any;
	R: any;
}
declare class TNormNpc extends TActor {
	CloseDialog(PlayObject: TPlayObject): void;
	Say(PlayObject: TPlayObject, Msg: string): void;
	SayEx(PlayObject: TPlayObject, UIName: string, Msg: string): void;
	UpdateSay(PlayObject: TPlayObject, UIName: string, Msg: string): void;
	MessageBox(PlayObject: TPlayObject, Msg: string): void;
	SayExMobile: (PlayObject: TPlayObject, UIName: string, Text: string,
Scale: number, boCenter?: boolean) => void;
	Question(PlayObject: TPlayObject, Msg: string, YMethod: string, NMethod:
string): void;
	Give: (Actor: TActor, ItemName: string, Count?: number, Upgrade?: boolean)
=> void;
	GiveItem: (Actor: TActor, ItemName: string, Upgrade?: boolean) =>
TUserItem;
	GiveItemByIndex(Actor: TActor, ItemIndex: number, Upgrade: boolean):
TUserItem;
	Take: (Actor: TActor, ItemName: string, Count?: number) => void;
	TakeItem(Actor: TActor, Item: TUserItem): boolean;
	RepairActorItem(Actor: TActor, ItemWhere: number): void;
```

```
      RepairActorAllItem(Actor: TActor): void;
      SetEffigyState(Job: number, Properties: number, Feature: number,
FeatureEx: number, Offset: number): void;
      GetTag(): number;
      readonly Tag: number;
      GetRankLevelName(): string;
      RankLevelName: string;
      SetRankLevelName(Value: string): void;

}
declare class TGuild {
      GetName(): string;
      readonly Name: string;
      GetGuildID(): number;
      readonly GuildID: number;
      GetChiefName(): string;
      readonly ChiefName: string;
      GetContestPoint(): number;
      ContestPoint: number;
      SetContestPoint(Value: number): void;
      GetTeamFight(): boolean;
      readonly TeamFight: boolean;
      GetMemberMaxLimit(): number;
      MemberMaxLimit: number;
      SetMemberMaxLimit(Value: number): void;
      GetMemberCount(): number;
      readonly MemberCount: number;
      GetCount(): number;
      readonly Count: number;
      IsFull(): boolean;
      GetBuildPoint(): number;
      BuildPoint: number;
      SetBuildPoint(Value: number): void;
      GetAurae(): number;
      Aurae: number;
      SetAurae(Value: number): void;
      GetStability(): number;
      Stability: number;
      SetStability(Value: number): void;
      GetFlourishing(): number;
      Flourishing: number;
      SetFlourishing(Value: number): void;
      GetFountainOpen(): boolean;
      FountainOpen: boolean;
      SetFountainOpen(Value: boolean): void;
      AddRankName(ARankNo: number, ARankName: string): boolean;
      UpdateRankName(AOldName: string, ANewName: string): boolean;
      DropRankName(ARankName: string): boolean;
      AddMemberWithRank(ARankName: string, APlayName: string): boolean;
      AddMember(APlayName: string): boolean;
      DeleteMember(APlayName: string): boolean;
      ChangeChief(APlayName: string): boolean;
      AddAlly0(AGuildName: string): boolean;
      AddAlly1(AAllyGuild: TGuild): boolean;
      DeleteAlly0(AGuildName: string): boolean;
      DeleteAlly1(AAllyGuild: TGuild): boolean;
      VarString(AVarName: string): TVarValue;
      VarInteger(AVarName: string): TVarValue;
      VarDateTime(AVarName: string): TVarValue;
      VarFloat(AVarName: string): TVarValue;
      VarBoolean(AVarName: string): TVarValue;
      GetRankNames(AList: TStringList): void;
      GetRankMembers(ARankName: string, AList: TStringList): void;
      GetCustomSaveStr(): string;
```

```
	CustomSaveStr: string;
	SetCustomSaveStr(Value: string): void;
	V: any;
	R: any;
}
declare class TUserCastle {
	GetMapCastle(): TEnvirnoment;
	readonly MapCastle: TEnvirnoment;
	GetMapPalace(): TEnvirnoment;
	readonly MapPalace: TEnvirnoment;
	GetMapSecret(): TEnvirnoment;
	readonly MapSecret: TEnvirnoment;
	GetName(): string;
	Name: string;
	SetName(Value: string): void;
	GetTotalGold(): number;
	TotalGold: number;
	SetTotalGold(Value: number): void;
	GetTodayIncome(): number;
	TodayIncome: number;
	SetTodayIncome(Value: number): void;
	GetGuild(): TGuild;
	Guild: TGuild;
	SetGuild(AGuild: TGuild): void;
	GetHomeMap(): string;
	readonly HomeMap: string;
	GetHomeX(): number;
	readonly HomeX: number;
	GetHomeY(): number;
	readonly HomeY: number;
	GetWarDate(): number;
	readonly WarDate: number;
	GetUnderWar(): boolean;
	readonly UnderWar: boolean;
	InCastleWarArea(Envir: TEnvirnoment, nX: number, nY: number): boolean;
	IsMember(Cert: TActor): boolean;
	IsMasterGuild(Guild: TGuild): boolean;
	IsAttackGuild(Guild: TGuild): boolean;
	IsAttackAllyGuild(Guild: TGuild): boolean;
	IsDefenseGuild(Guild: TGuild): boolean;
	IsDefenseAllyGuild(Guild: TGuild): boolean;
	StartWall(): void;
	StopWall(): void;
	AddAttacker(Guild: TGuild): boolean;
	AddAttackerByTime(Guild: TGuild, ATime: number): boolean;
	AddAllAttacker(): void;
	AddAllAttackerByTime(ATime: number): void;
	RepairDoor(): boolean;
	GetLeftWall(): TActor;
	readonly LeftWall: TActor;
	GetCenterWall(): TActor;
	readonly CenterWall: TActor;
	GetRightWall(): TActor;
	readonly RightWall: TActor;
	RepairWall(nWallIndex: number): boolean;
	GetMainDoorOpened(): boolean;
	readonly MainDoorOpened: boolean;
	GetMainDoor(): TActor;
	readonly MainDoor: TActor;
	OpenDoor(): void;
	CloseDoor(): void;
	HireArcher(Index: number): boolean;
	HireGuard(Index: number): boolean;
	UnHireArcher(Index: number): boolean;
```

```
	UnHireGuard(Index: number): boolean;
	GetGuard(Index: number): TActor;
	GetArcher(Index: number): TActor;
	GetAttackWarCount(): number;
	readonly AttackWarCount: number;
	GetAttackWarDate(Index: number): number;
	GetAttackWarGuild(Index: number): TGuild;
	GetAttackGuildCount(): number;
	readonly AttackGuildCount: number;
	GetAttackGuild(Index: number): TGuild;
	GetChangeDateTime(): number;
	readonly ChangeDateTime: number;
	GetStartWarTick(): number;
	readonly StartWarTick: number;
	GetStopWarTick(): number;
	readonly StopWarTick: number;

}
declare class TStdItem {
	GetName(): string;
	readonly Name: string;
	GetIndex(): number;
	readonly Index: number;
	GetStdMode(): number;
	readonly StdMode: number;
	GetShape(): number;
	readonly Shape: number;
	GetWeight(): number;
	readonly Weight: number;
	GetAniCount(): number;
	readonly AniCount: number;
	GetSource(): number;
	readonly Source: number;
	GetReserved(): number;
	readonly Reserved: number;
	GetLooks(): number;
	readonly Looks: number;
	GetDuraMax(): number;
	readonly DuraMax: number;
	GetAC(): number;
	readonly AC: number;
	GetAC2(): number;
	readonly AC2: number;
	GetMAC(): number;
	readonly MAC: number;
	GetMAC2(): number;
	readonly MAC2: number;
	GetDC(): number;
	readonly DC: number;
	GetDC2(): number;
	readonly DC2: number;
	GetMC(): number;
	readonly MC: number;
	GetMC2(): number;
	readonly MC2: number;
	GetSC(): number;
	readonly SC: number;
	GetSC2(): number;
	readonly SC2: number;
	GetTC(): number;
	readonly TC: number;
	GetTC2(): number;
	readonly TC2: number;
	GetPC(): number;
```

```
	readonly PC: number;
	GetPC2(): number;
	readonly PC2: number;
	GetWC(): number;
	readonly WC: number;
	GetWC2(): number;
	readonly WC2: number;
	GetNeed(): number;
	readonly Need: number;
	GetNeedLevel(): number;
	readonly NeedLevel: number;
	GetPrice(): number;
	readonly Price: number;
	GetJob(): number;
	readonly Job: number;
	GetColor(): number;
	readonly Color: number;
	GetTypeID(): number;
	readonly TypeID: number;

}
declare class TUserItem {
	GetName(): string;
	readonly Name: string;
	GetDisplayName(): string;
	readonly DisplayName: string;
	Rename(NewName: string): boolean;
	RevertName(): boolean;
	GetColor(): number;
	Color: number;
	SetColor(Value: number): void;
	GetQuality(): number;
	Quality: number;
	SetQuality(Value: number): void;
	MakeHole(): boolean;
	OpenSoul(): boolean;
	CopyFrom(Source: TUserItem): boolean;
	CopyFromNotChangeMakeIndex(Source: TUserItem): boolean;
	GetStdMode(): number;
	readonly StdMode: number;
	GetNeed(): number;
	Need: number;
	SetNeed(Value: number): void;
	GetNeedLevel(): number;
	NeedLevel: number;
	SetNeedLevel(Value: number): void;
	GetPrice(): number;
	readonly Price: number;
	GetPrice2(): number;
	readonly Price2: number;
	GetShape(): number;
	Shape: number;
	SetShape(Value: number): void;
	GetLook(): number;
	Look: number;
	SetLook(Value: number): void;
	GetAniCount(): number;
	AniCount: number;
	SetAniCount(Value: number): void;
	GetSource(): number;
	readonly Source: number;
	GetJob(): number;
	readonly Job: number;
	GetMakeIndex(): number;
```

```
readonly MakeIndex: number;
GetMakeIndexLowPart(): number;
readonly MakeIndexLowPart: number;
GetMakeIndexHighPart(): number;
readonly MakeIndexHighPart: number;
GetMakeIndexString(): string;
readonly MakeIndexString: string;
GetUUID(): string;
readonly UUID: string;
GetItemIndex(): number;
readonly ItemIndex: number;
GetDura(): number;
Dura: number;
SetDura(Value: number): void;
GetDuraMax(): number;
DuraMax: number;
SetDuraMax(Value: number): void;
GetMaxDate(): number;
MaxDate: number;
SetMaxDate(Value: number): void;
GetAddAC(): number;
AddAC: number;
SetAddAC(Value: number): void;
GetAddMAC(): number;
AddMAC: number;
SetAddMAC(Value: number): void;
GetAddDC(): number;
AddDC: number;
GetAddDCStr(): string;
readonly AddDCStr: string;
SetAddDC(Value: number): void;
GetAddMC(): number;
AddMC: number;
SetAddMC(Value: number): void;
GetAddSC(): number;
AddSC: number;
SetAddSC(Value: number): void;
GetAddTC(): number;
AddTC: number;
SetAddTC(Value: number): void;
GetAddPC(): number;
AddPC: number;
SetAddPC(Value: number): void;
GetAddWC(): number;
AddWC: number;
SetAddWC(Value: number): void;
GetAddLAC(): number;
AddLAC: number;
SetAddLAC(Value: number): void;
GetAddLMAC(): number;
AddLMAC: number;
SetAddLMAC(Value: number): void;
GetAddLDC(): number;
AddLDC: number;
SetAddLDC(Value: number): void;
GetAddLMC(): number;
AddLMC: number;
SetAddLMC(Value: number): void;
GetAddLSC(): number;
AddLSC: number;
SetAddLSC(Value: number): void;
GetAddLTC(): number;
AddLTC: number;
SetAddLTC(Value: number): void;
```

```
GetAddLPC(): number;
AddLPC: number;
SetAddLPC(Value: number): void;
GetAddLWC(): number;
AddLWC: number;
SetAddLWC(Value: number): void;
GetBind(): boolean;
Bind: boolean;
SetBind(Value: boolean): void;
GetNeverDrop(): boolean;
NeverDrop: boolean;
SetNeverDrop(Value: boolean): void;
GetNoRepair(): boolean;
NoRepair: boolean;
SetNoRepair(Value: boolean): void;
GetNoStore(): boolean;
NoStore: boolean;
SetNoStore(Value: boolean): void;
GetAddHitPoint(): number;
AddHitPoint: number;
SetAddHitPoint(Value: number): void;
GetAddHitSpeed(): number;
AddHitSpeed: number;
SetAddHitSpeed(Value: number): void;
GetAddSpeedPoint(): number;
AddSpeedPoint: number;
SetAddSpeedPoint(Value: number): void;
GetAddAntiMagic(): number;
AddAntiMagic: number;
SetAddAntiMagic(Value: number): void;
GetAddAntiPoison(): number;
AddAntiPoison: number;
SetAddAntiPoison(Value: number): void;
GetAddHealthRecover(): number;
AddHealthRecover: number;
SetAddHealthRecover(Value: number): void;
GetAddSpellRecover(): number;
AddSpellRecover: number;
SetAddSpellRecover(Value: number): void;
GetAddPoisonRecover(): number;
AddPoisonRecover: number;
SetAddPoisonRecover(Value: number): void;
GetAddLuck(): number;
AddLuck: number;
SetAddLuck(Value: number): void;
GetUpgrade(): number;
Upgrade: number;
SetUpgrade(Value: number): void;
GetMaxUpgrade(): number;
MaxUpgrade: number;
SetMaxUpgrade(Value: number): void;
GetAddPoint(Idx: number): TAddPointItem;
GetAddLevel(Idx: number): TAddLevelItem;
SetAddHold(Idx: number, Value: number): void;
SetAddHole(Idx: number, Value: number): void;
GetAddHold(Idx: number): number;
GetAddHole(Idx: number): number;
AddHole: number;
GetOpenedSoul(): boolean;
readonly OpenedSoul: boolean;
GetSoulLevel(): number;
SoulLevel: number;
SetSoulLevel(Value: number): void;
GetSoulExp(): number;
```

```
        SoulExp: number;
        SetSoulExp(Value: number): void;
        SetSoulMaxExp(Value: number): void;
        GetSoulMaxExp(): number;
        SoulMaxExp: number;
        GetEffectType(): number;
        EffectType: number;
        SetEffectType(Value: number): void;
        GetTotalAbility(): number;
        readonly TotalAbility: number;
        GetItemOutWay(): number;
        ItemOutWay: number;
        SetItemOutWay(Value: number): void;
        MakeString(): string;
        MakeString2(): string;
        SetFromDateTime(Value: number): void;
        GetFromDateTime(): number;
        FromDateTime: number;
        GetCustomEffect(): number;
        CustomEffect: number;
        SetCustomEffect(Value: number): void;
        GetCustomEffect2(): number;
        CustomEffect2: number;
        SetCustomEffect2(Value: number): void;
        GetCustomEffect3(): number;
        CustomEffect3: number;
        SetCustomEffect3(Value: number): void;
        GetCustomCaption(Index: number): string;
        SetCustomCaption(Index: number, Value: string): void;
        GetCustomText(Index: number): string;
        SetCustomText(Index: number, Value: string): void;
        GetCustomDesc(): string;
        CustomDesc: string;
        SetCustomDesc(Value: string): void;
        GetOutWay1(Index: number): number;
        OutWay1: number;
        SetOutWay1(Index: number, Value: number): void;
        GetOutWay2(Index: number): number;
        OutWay2: number;
        SetOutWay2(Index: number, Value: number): void;
        GetOutWay3(Index: number): number;
        OutWay3: number;
        SetOutWay3(Index: number, Value: number): void;
        GetOutWay4(Index: number): number;
        OutWay4: number;
        SetOutWay4(Index: number, Value: number): void;
        GetCustomValue(Index: number): number;
        CustomValue: number;
        SetCustomValue(Index: number, Value: number): void;
        GetDropEff(): number;
        DropEff: number;
        SetDropEff(Value: number): void;
        GetState(): TItemState;
        readonly State: TItemState;
        MakeCopy(): TUserItem;
        GetIsOpenOutWay4(): boolean;
        IsOpenOutWay4: boolean;
        SetIsOpenOutWay4(Value: boolean): void;
        GetCompareScore(): number;
        CompareScore: number;
        SetCompareScore(Value: number): void;

}
declare class TEnvirnoment {
```

```
GetName(): string;
readonly Name: string;
GetMapName(): string;
readonly MapName: string;
GetMapID(): string;
readonly MapID: string;
GetDuplicate(): boolean;
readonly Duplicate: boolean;
GetDark(): boolean;
Dark: boolean;
SetDark(Value: boolean): void;
GetDay(): boolean;
Day: boolean;
SetDay(Value: boolean): void;
GetDarkness(): boolean;
Darkness: boolean;
SetDarkness(Value: boolean): void;
GetDayLight(): boolean;
DayLight: boolean;
SetDayLight(Value: boolean): void;
GetSafe(): boolean;
Safe: boolean;
SetSafe(Value: boolean): void;
GetFight(): boolean;
Fight: boolean;
SetFight(Value: boolean): void;
GetFight3(): boolean;
Fight3: boolean;
SetFight3(Value: boolean): void;
GetQuiz(): boolean;
Quiz: boolean;
SetQuiz(Value: boolean): void;
GetNoReconect(): boolean;
NoReconect: boolean;
SetNoReconect(Value: boolean): void;
GetReConnectMap(): string;
ReConnectMap: string;
SetReConnectMap(Value: string): void;
GetMUSIC(): boolean;
MUSIC: boolean;
SetMUSIC(Value: boolean): void;
GetExpRate(): number;
ExpRate: number;
SetExpRate(Value: number): void;
GetPKWinLevel(): number;
PKWinLevel: number;
SetPKWinLevel(Value: number): void;
GetPKWinExp(): number;
PKWinExp: number;
SetPKWinExp(Value: number): void;
GetPKLostLevel(): number;
PKLostLevel: number;
SetPKLostLevel(Value: number): void;
GetPKLostExp(): number;
PKLostExp: number;
SetPKLostExp(Value: number): void;
GetDecHP(): number;
DecHP: number;
SetDecHP(Value: number): void;
GetDecHPTime(): number;
DecHPTime: number;
SetDecHPTime(Value: number): void;
GetIncHP(): number;
IncHP: number;
```

```
SetIncHP(Value: number): void;
GetIncHPTime(): number;
IncHPTime: number;
SetIncHPTime(Value: number): void;
GetDecGameGold(): number;
DecGameGold: number;
SetDecGameGold(Value: number): void;
GetDecGameGoldTime(): number;
DecGameGoldTime: number;
SetDecGameGoldTime(Value: number): void;
GetDecGamePoint(): number;
DecGamePoint: number;
SetDecGamePoint(Value: number): void;
GetDecGamePointTime(): number;
DecGamePointTime: number;
SetDecGamePointTime(Value: number): void;
GetIncGameGold(): number;
IncGameGold: number;
SetIncGameGold(Value: number): void;
GetIncGameGoldTime(): number;
IncGameGoldTime: number;
SetIncGameGoldTime(Value: number): void;
GetIncGamePoint(): number;
IncGamePoint: number;
SetIncGamePoint(Value: number): void;
GetIncGamePointTime(): number;
IncGamePointTime: number;
SetIncGamePointTime(Value: number): void;
GetRunHuman(): boolean;
RunHuman: boolean;
SetRunHuman(Value: boolean): void;
GetRunMon(): boolean;
RunMon: boolean;
SetRunMon(Value: boolean): void;
GetNeedHole(): boolean;
NeedHole: boolean;
SetNeedHole(Value: boolean): void;
GetNoRecall(): boolean;
NoRecall: boolean;
SetNoRecall(Value: boolean): void;
GetNoGuildRecall(): boolean;
NoGuildRecall: boolean;
SetNoGuildRecall(Value: boolean): void;
GetNoDearRecall(): boolean;
NoDearRecall: boolean;
SetNoDearRecall(Value: boolean): void;
GetNoMasterRecall(): boolean;
NoMasterRecall: boolean;
SetNoMasterRecall(Value: boolean): void;
GetNoRandomMove(): boolean;
NoRandomMove: boolean;
SetNoRandomMove(Value: boolean): void;
GetNoDrug(): boolean;
NoDrug: boolean;
SetNoDrug(Value: boolean): void;
GetMine(): boolean;
Mine: boolean;
SetMine(Value: boolean): void;
GetNoPostionMove(): boolean;
NoPostionMove: boolean;
SetNoPostionMove(Value: boolean): void;
GetFightPK(): boolean;
FightPK: boolean;
SetFightPK(Value: boolean): void;
```

```
	GetNoHorse(): boolean;
	NoHorse: boolean;
	SetNoHorse(Value: boolean): void;
	GetMonCount(): number;
	readonly MonCount: number;
	GetHumCount(): number;
	readonly HumCount: number;
	ClearMon(ANoDrop: boolean, AMonName: string): void;
	GetMonCountEx: (AMonName?: string) => number;
	readonly MonCountEx: number;
	GetMonCountExclude(ExcludeMonName: string): number;
	readonly MonCountExclude: number;
	GetMonCountInRange(MapX: number, MapY: number, Range: number, AMonName:
string): number;
	readonly MonCountInRange: number;
	GetActorListInRange: (MapX: number, MapY: number, Range: number,
ActorName?: string) => TActorList;
	readonly ActorListInRange: TActorList;
	GetActorListAt: (MapX: number, MapY: number, ActorName?: string) =>
TActorList;
	readonly ActorListAt: TActorList;
	GetFreeing(): boolean;
	readonly Freeing: boolean;
	GetStartTime(): number;
	readonly StartTime: number;
	CanMove(MapX: number, MapY: number, boIgnoreActor: boolean): boolean;
	SetInTroubleModeActive(boDenySay: boolean, boDenyAutoColor: boolean,
boDenyViewUserInfo: boolean, WeaponShape: number, DressShape: number, SameName:
string): void;
	SetInTroubleModeClose(): void;
	SetDisplayName(S: string): void;
	GetDisplayName(): string;
	DisplayName: string;
	GetMapWidth(): number;
	readonly MapWidth: number;
	GetMapHeight(): number;
	readonly MapHeight: number;
	ThrowMonItems(MapX: number, MapY: number, Range: number, MonName: string,
ItemOwner: TPlayObject, ForbidPickUpTime: number, AppendDropRatePercent:
number): number;

}
declare class TUserMagic {
	GetName(): string;
	readonly Name: string;
	GetMagID(): number;
	readonly MagID: number;
	GetMagIdx(): number;
	readonly MagIdx: number;
	GetJob(): number;
	readonly Job: number;
	GetLevel(): number;
	Level: number;
	SetLevel(Value: number): void;
	GetStrengthen(): number;
	Strengthen: number;
	SetStrengthen(Value: number): void;
	GetTranPoint(): number;
	TranPoint: number;
	SetTranPoint(Value: number): void;
	GetUseTime(): number;
	UseTime: number;
	SetUseTime(Value: number): void;
	GetCDTime(): number;
```

```
        readonly CDTime: number;
        GetOpened(): boolean;
        Opened: boolean;
        SetOpened(Value: boolean): void;
        GetMagicInfo(): TMagic;
        readonly MagicInfo: TMagic;
        GetKey(): TKeyBoardCode;
        Key: TKeyBoardCode;
        SetKey(Key: TKeyBoardCode): void;
        GetKeyCtrl(): boolean;
        KeyCtrl: boolean;
        SetKeyCtrl(Value: boolean): void;
        GetLeftTopIcon(): number;
        LeftTopIcon: number;
        SetLeftTopIcon(Icon: number): void;
        GetRightTopIcon(): number;
        RightTopIcon: number;
        SetRightTopIcon(Icon: number): void;
        GetLeftBottomIcon(): number;
        LeftBottomIcon: number;
        SetLeftBottomIcon(Icon: number): void;
        GetRightBottompIcon(): number;
        RightBottompIcon: number;
        SetRightBottompIcon(Icon: number): void;

    }
    declare class TBuff {
        GetTag(): number;
        readonly Tag: number;
        GetID(): number;
        readonly ID: number;
        GetHandle(): number;
        readonly Handle: number;
        GetGroupID(): number;
        readonly GroupID: number;
        GetBuffValue(): number;
        readonly BuffValue: number;
        GetValueRate(): boolean;
        readonly ValueRate: boolean;
        GetDestoryTick(): number;
        DestoryTick: number;
        SetDestoryTick(value: number): void;
        AffectInterval(): number;
        GetActor(): TActor;
        Actor: TActor;
        SetActor(Value: TActor): void;
        GetCustomInt(Index: number): number;
        CustomInt: number;
        SetCustomInt(Index: number, value: number): void;
        SetCustomStr(Str: string): void;
        GetCustomStr(): string;
        CustomStr: string;
        GetCategory(): number;
        readonly Category: number;
        GetBuffAbilityType(): number;
        readonly BuffAbilityType: number;
        GetBuffStatusType(): number;
        readonly BuffStatusType: number;
        GetBuffIntervalType(): number;
        readonly BuffIntervalType: number;
        GetArgs1(): number;
        readonly Args1: number;
        GetArgs2(): number;
        readonly Args2: number;
```

```
	SetGroup(value: number): void;

}
declare class THumTitle {
	GetIndex(): number;
	readonly Index: number;
	GetName(): string;
	readonly Name: string;
	GetLooks(): number;
	readonly Looks: number;
	GetTitleID(): number;
	readonly TitleID: number;
	GetHP(): number;
	readonly HP: number;
	GetMP(): number;
	readonly MP: number;
	GetAC(): number;
	readonly AC: number;
	GetACMax(): number;
	readonly ACMax: number;
	GetMAC(): number;
	readonly MAC: number;
	GetMACMax(): number;
	readonly MACMax: number;
	GetDC(): number;
	readonly DC: number;
	GetDCMax(): number;
	readonly DCMax: number;
	GetMC(): number;
	readonly MC: number;
	GetMCMax(): number;
	readonly MCMax: number;
	GetSC(): number;
	readonly SC: number;
	GetSCMax(): number;
	readonly SCMax: number;
	GetTC(): number;
	readonly TC: number;
	GetTCMax(): number;
	readonly TCMax: number;
	GetPC(): number;
	readonly PC: number;
	GetPCMax(): number;
	readonly PCMax: number;
	GetWC(): number;
	readonly WC: number;
	GetWCMax(): number;
	readonly WCMax: number;
	GetHitPoint(): number;
	readonly HitPoint: number;
	GetAntiPoison(): number;
	readonly AntiPoison: number;
	GetPoisonRecover(): number;
	readonly PoisonRecover: number;
	GetHealthRecover(): number;
	readonly HealthRecover: number;
	GetSpellRecover(): number;
	readonly SpellRecover: number;
	GetAntiMagic(): number;
	readonly AntiMagic: number;
	GetLimit(): number;
	Limit: number;
	SetLimit(Value: number): void;
```

```
}
declare class TMissionItem {
      GetRecordID(): string;
      readonly RecordID: string;
      GetKind(): number;
      readonly Kind: number;
      GetMissionID(): string;
      readonly MissionID: string;
      GetSubject(): string;
      Subject: string;
      SetSubject(AValue: string): void;
      GetContent(): string;
      Content: string;
      SetContent(AValue: string): void;
      GetRewards(): string;
      Rewards: string;
      SetRewards(AValue: string): void;
      GetNeedType(): number;
      NeedType: number;
      SetNeedType(AValue: number): void;
      GetNeedName(): string;
      NeedName: string;
      SetNeedName(AValue: string): void;
      GetNeedQuality(): number;
      NeedQuality: number;
      SetNeedQuality(AValue: number): void;
      GetNeedMax(): number;
      NeedMax: number;
      SetNeedMax(AValue: number): void;
      GetNeedProgress(): number;
      NeedProgress: number;
      SetNeedProgress(AValue: number): void;
      GetTargetNPC(): number;
      TargetNPC: number;
      SetTargetNPC(AValue: number): void;
      GetStart(): number;
      Start: number;
      SetStart(AValue: number): void;
      GetLimit(): number;
      Limit: number;
      SetLimit(AValue: number): void;
      GetState(): number;
      State: number;
      SetState(AValue: number): void;
      Update(): void;
      Submit(): void;
      Cancel(): void;
      Delete(): void;

}
declare class TMissions {
      Clear(): void;
      Contain(ARecordID: string): boolean;
      TryGet(ARecordID: string): TMissionItem;
      ContainMissionIDInDB(AMissionID: string): boolean;
      ContainMissionID(AMissionID: string): boolean;
      TryGetByMissionID(AMissionID: string): TMissionItem;
      ContainLink(AMissionID: string): boolean;
      TryGetLink(AMissionID: string): TMissionLinkItem;
      Add1(AKind: number, AMissionID: string, ATargetNPC: number): TMissionItem;
      Add2(AKind: number, AMissionID: string, ANeedType: number, ASubject:
string, AContent: string, ARewards: string, ANeedName: string, ANeedMax: number,
ANeedQuality: number, ALimit: number, ATargetNPC: number, AAutoComplete:
boolean): TMissionItem;
```

```
      AddLink1(AKind: number, AMissionID: string, ATargetNPC: number):
TMissionLinkItem;
      AddLink2(AKind: number, AMissionID: string, ASubject: string, AContent:
string, ATargetNPC: number, ANeedLevel: number, ANeedReLevel: number):
TMissionLinkItem;
      Delete(ARecordID: string): void;
      DeleteLink(AMissionID: string): void;
      Refresh(): void;
      GetCount(): number;
      readonly Count: number;
      GetItems(AIndex: number): TMissionItem;
      readonly Items: TMissionItem;
      IndexOf(ARecordID: string): number;
      GetLinkCount(): number;
      readonly LinkCount: number;
      GetLinkItems(AIndex: number): TMissionLinkItem;
      readonly LinkItems: TMissionLinkItem;
      IndexOfLink(AMissionID: string): number;

}
declare class TAddPointItem {
      GetValue(): number;
      Value: number;
      SetValue(Value: number): void;
      GetValueType(): number;
      ValueType: number;
      SetValueType(Value: number): void;

}
declare class TAddLevelItem {
      GetValue(): number;
      Value: number;
      SetValue(Value: number): void;
      GetValueType(): number;
      ValueType: number;
      SetValueType(Value: number): void;
      GetState(): number;
      State: number;
      SetState(Value: number): void;

}
declare class TAddedAbility {
      GetHP(): number;
      HP: number;
      SetHP(Value: number): void;
      GetMP(): number;
      MP: number;
      SetMP(Value: number): void;
      GetAC(): number;
      AC: number;
      SetAC(Value: number): void;
      GetACMax(): number;
      ACMax: number;
      SetACMax(Value: number): void;
      GetMAC(): number;
      MAC: number;
      SetMAC(Value: number): void;
      GetMACMax(): number;
      MACMax: number;
      SetMACMax(Value: number): void;
      GetDC(): number;
      DC: number;
      SetDC(Value: number): void;
      GetDCMax(): number;
```

```
DCMax: number;
SetDCMax(Value: number): void;
GetMC(): number;
MC: number;
SetMC(Value: number): void;
GetMCMax(): number;
MCMax: number;
SetMCMax(Value: number): void;
GetSC(): number;
SC: number;
SetSC(Value: number): void;
GetSCMax(): number;
SCMax: number;
SetSCMax(Value: number): void;
GetTC(): number;
TC: number;
SetTC(Value: number): void;
GetTCMax(): number;
TCMax: number;
SetTCMax(Value: number): void;
GetPC(): number;
PC: number;
SetPC(Value: number): void;
GetPCMax(): number;
PCMax: number;
SetPCMax(Value: number): void;
GetWC(): number;
WC: number;
SetWC(Value: number): void;
GetWCMax(): number;
WCMax: number;
SetWCMax(Value: number): void;
GetLuck(): number;
Luck: number;
SetLuck(Value: number): void;
GetHitPoint(): number;
HitPoint: number;
SetHitPoint(Value: number): void;
GetHitSpeed(): number;
HitSpeed: number;
SetHitSpeed(Value: number): void;
GetAntiPoison(): number;
AntiPoison: number;
SetAntiPoison(Value: number): void;
GetPoisonRecover(): number;
PoisonRecover: number;
SetPoisonRecover(Value: number): void;
GetHealthRecover(): number;
HealthRecover: number;
SetHealthRecover(Value: number): void;
GetSpellRecover(): number;
SpellRecover: number;
SetSpellRecover(Value: number): void;
GetAntiMagic(): number;
AntiMagic: number;
SetAntiMagic(Value: number): void;
GetDamageAbsorb(): number;
DamageAbsorb: number;
SetDamageAbsorb(Value: number): void;
GetDamageAdd(): number;
DamageAdd: number;
SetDamageAdd(Value: number): void;
GetPunchHit(): number;
PunchHit: number;
```

```
        SetPunchHit(Value: number): void;
        GetCriticalHit(): number;
        CriticalHit: number;
        SetCriticalHit(Value: number): void;
        GetRebound(): number;
        Rebound: number;
        SetRebound(Value: number): void;
        GetExpRate(): number;
        ExpRate: number;
        SetExpRate(Value: number): void;
        GetGoldRate(): number;
        GoldRate: number;
        SetGoldRate(Value: number): void;
        GetItemRate(): number;
        ItemRate: number;
        SetItemRate(Value: number): void;
        GetAppendDamage(): number;
        AppendDamage: number;
        SetAppendDamage(Value: number): void;
        GetSpeedPoint(): number;
        SpeedPoint: number;
        SetSpeedPoint(Value: number): void;
        GetCriticalHitDef(): number;
        CriticalHitDef: number;
        SetCriticalHitDef(Value: number): void;
        GetPunchHitDef(): number;
        PunchHitDef: number;
        SetPunchHitDef(Value: number): void;
        GetAppendDamageDef(): number;
        AppendDamageDef: number;
        SetAppendDamageDef(Value: number): void;
        GetHPRate(): number;
        HPRate: number;
        SetHPRate(Value: number): void;
        GetMPRate(): number;
        MPRate: number;
        SetMPRate(Value: number): void;
        GetWeaponStrong(): number;
        WeaponStrong: number;
        SetWeaponStrong(Value: number): void;
        GetHoly(): number;
        Holy: number;
        SetHoly(Value: number): void;
        GetWearWeight(): number;
        WearWeight: number;
        SetWearWeight(Value: number): void;
        GetMaxWeight(): number;
        MaxWeight: number;
        SetMaxWeight(Value: number): void;
        GetPunchHitAppendDamage(): number;
        PunchHitAppendDamage: number;
        SetPunchHitAppendDamage(Value: number): void;
        GetCriticalHitAppendDamage(): number;
        CriticalHitAppendDamage: number;
        SetCriticalHitAppendDamage(Value: number): void;
        GetFatalHit(): number;
        FatalHit: number;
        SetFatalHit(Value: number): void;
        GetPunchHitWRate(): number;
        PunchHitWRate: number;
        SetPunchHitWRate(Value: number): void;

}
declare class TM2Core {
```

```
      GetGVar(index: number): number;
      GVar: number;
      GetAVar(index: number): string;
      AVar: string;
      GetIVar(index: number): number;
      IVar: number;
      GetUVar(index: number): string;
      UVar: string;
      SetGVar(index: number, Value: number): void;
      SetAVar(index: number, Value: string): void;
      SetIVar(index: number, Value: number): void;
      SetUVar(index: number, Value: string): void;
      GetPlayCount(): number;
      readonly PlayCount: number;
      GetPlayer(index: number): TPlayObject;
      Kick(APlayName: string): void;
      KickAll(): void;
      ClearMapMon: (AMapName: string, ANoDrop?: boolean, AMonName?: string) =>
void;
      ShowEffect(Map: string, MapX: number, MapY: number, Effect: number,
IntervalTime: number): void;
      ShowEffect2(Map: string, MapX: number, MapY: number, EffectID: number,
LoopCount: number): void;
      CreateAttackEvent: (Map: string, AMapX: number, AMapY: number, ATime:
number, AInterval: number, ADamage: number, SkillEffectID?: number, MagId?:
number, MagLv?: number) => boolean;
      CreateAttackEvent2: (Map: string, AMapX: number, AMapY: number, ATime:
number, AInterval: number, ADamage: number, UIEffectID?: number, MagId?: number,
MagLv?: number) => boolean;
      DeleteAttackEvent(AMap: TEnvirnoment, AMapX: number, AMapY: number, Owner:
TActor, MagId: number): number;
      RecallMap(ASourceMap: string, ADestMap: string): void;
      MonGen: (AMap: string, AMonName: string, AMonCount: number, AMapX: number,
AMapY: number, ARange: number, ACamp?: number, ANation?: number, ATag?: number,
ARevivalEvent?: boolean, ADieEvent?: boolean, AKillerEvent?: boolean,
ADamageEvent?: boolean) => TActorList;
      MonGenEx(AMap: TEnvirnoment, AMonName: string, AMonCount: number, AMapX:
number, AMapY: number, ARange: number, ACamp: number, ANation: number, ATag:
number, ARevivalEvent: boolean, ADieEvent: boolean, AKillerEvent: boolean,
ADamageEvent: boolean): TActorList;
      MobPlace: (AMissionMap: string, AMissionX: number, AMissionY: number,
AMonName: string, AX: number, AY: number, ACount?: number, ARange?: number,
ACamp?: number, ANation?: number, ATag?: number, ARevivalEvent?: boolean,
ADieEvent?: boolean, AKillerEvent?: boolean, ADamageEvent?: boolean) =>
TActorList;
      MobPlaceEx(AMissionMap: TEnvirnoment, AMissionX: number, AMissionY:
number, AMonName: string, AX: number, AY: number, ACount: number, ARange:
number, ACamp: number, ANation: number, ATag: number, ARevivalEvent: boolean,
ADieEvent: boolean, AKillerEvent: boolean, ADamageEvent: boolean): TActorList;
      GetServerName(): string;
      readonly ServerName: string;
      GetServerID(): number;
      readonly ServerID: number;
      GetDBEngine(): TDBEngine;
      readonly DBEngine: TDBEngine;
      GetToptenz(): TToptenz;
      readonly Toptenz: TToptenz;
      WriteLog(Log: string): void;
      RefreshDBItems(): void;
      GetStartTime(): number;
      readonly StartTime: number;
      GetEnvirPath(): string;
      readonly EnvirPath: string;
      GetPath(): string;
```

```
      readonly Path: string;
      BroadcastMoveMessage(Message: string, _Type: number, DuraTick: number):
void;
      GetClientSpeed(): number;
      ClientSpeed: number;
      SetClientSpeed(Value: number): void;
      RandomRange(Min: number, Max: number): number;
      BuildDB(): void;
      Lock(): void;
      UnLock(): void;
      Load(): void;
      Save(isStop: boolean): void;
      ReLoadScriptEngine(): void;
      CreateIdentStringList(Ident: string): TStringList;
      GetIdentStringList(Ident: string): TStringList;
      readonly IdentStringList: TStringList;
      FreeIdentStringList(Ident: string): void;
      CreateStringList(): TStringList;
      DebugOutStringList(): void;
      DefString(AVarName: string): TVarValue;
      DefInteger(AVarName: string): TVarValue;
      DefDateTime(AVarName: string): TVarValue;
      DefFloat(AVarName: string): TVarValue;
      DefBoolean(AVarName: string): TVarValue;
      ClearVarTable(): void;
      RemoveVar(AName: string): void;
      Broadcast(Msg: string): void;
      BroadcastSay(Msg: string, FontColor: number, Backgound: number): void;
      BroadcastTopMessage(Message: string): void;
      BroadcastCenterMessage: (Message: string, DuraTick?: number) => void;
      BroadcastCountDownMessage(Message: string): void;
      StartQuest(AQManagerMethod: string): void;
      FindStdItem(ItemIndex: number): TStdItem;
      FindItemIndex(Name: string): number;
      FindMap(AMap: string): TEnvirnoment;
      AddMapRoute(Name: string, Source: string, SourceX: number, SourceY:
number, Dest: string, DestX: number, DestY: number, UsefulLife: number, Looks:
number, ShowName: boolean): boolean;
      DelMapRoute(Name: string): boolean;
      CreateDuplicateMap(Name: string, UsefulLife: number): TEnvirnoment;
      CloseDuplicateMap(MapName: string): boolean;
      CloseDuplicateMapEx(Map: TEnvirnoment): boolean;
      AddNpc(Name: string, Map: string, MapX: number, MapY: number, Appr:
number, AUnitName: string): TNormNpc;
      RemoveNpc(Name: string, Map: string, MapX: number, MapY: number): boolean;
      AddMonNpc: (MonName: string, NpcName: string, Map: string, MapX: number,
MapY: number, AUnitName?: string, ACamp?: number, ANation?: number,
AttackDiffCamp?: boolean, AttackDiffNation?: boolean, AttackRed?: boolean,
AttackMon?: boolean, MissionX?: number, MissionY?: number, MissionRange?:
number) => TActor;
      AddMonNpcEx: (MonName: string, NpcName: string, Map: TEnvirnoment, MapX:
number, MapY: number, AUnitName?: string, ACamp?: number, ANation?: number,
AttackDiffCamp?: boolean, AttackDiffNation?: boolean, AttackRed?: boolean,
AttackMon?: boolean, MissionX?: number, MissionY?: number, MissionRange?:
number) => TActor;
      RemoveMonNpc(NpcName: string, Map: string, MapX: number, MapY: number):
boolean;
      RemoveMonNpcEx(NpcName: string, Map: TEnvirnoment, MapX: number, MapY:
number): boolean;
      FindPlayer(AName: string): TPlayObject;
      FindPlayerEx(AAccount: string, AName: string): TPlayObject;
      CreateGuild(AName: string, AChief: string): TGuild;
      FindGuild(sGuildName: string): TGuild;
      GetGuildCount(): number;
```

```
      readonly GuildCount: number;
      GetGuildByIndex(index: number): TGuild;
      readonly GuildByIndex: TGuild;
      DeleteGuild(sGuildName: string): boolean;
      DeleteGuildForce(sGuildName: string): boolean;
      AddGuildWar(SourceGuild: string, DesTGuild: string): boolean;
      FindCastle(Name: string): TUserCastle;
      FindNpc(NpcName: string): TNormNpc;
      FindMerchant(MapName: string, MerchantName: string): TNormNpc;
      FindMerchantByTag(ATag: number): TNormNpc;
      SavePlayerToFile(Source: TPlayObject, AFileName: string): void;
      AddClonePlayer(Source: TPlayObject, AName: string, Envir: TEnvirnoment,
MapX: number, MapY: number): TActor;
      AddClonePlayerFromFile(AFileName: string, AName: string, Envir:
TEnvirnoment, MapX: number, MapY: number): TActor;
      DelayGoto(ID: number, AIntervalTime: number, Once: boolean): boolean;
      ClearDelayGoto(ID: number): void;
      RenamePlayer(PlayObject: TPlayObject, NewName: string): void;
      ThrowItem: (MapName: string, ItemName: string, MapX: number, MapY: number,
Range: number, Count: number, Owner?: TActor, ForbidTime?: number,
CallBackIdent?: number) => boolean;
      ThrowItemEx: (Map: TEnvirnoment, ItemName: string, MapX: number, MapY:
number, Range: number, Count: number, Owner?: TActor, ForbidTime?: number,
CallBackIdent?: number) => boolean;
      RandomUpgrade(UserItem: TUserItem): boolean;
      RandomUpgradePoint(UserItem: TUserItem): boolean;
      RandomUpgradeLevel(UserItem: TUserItem): void;
      AddRobot(ExecTime: number, AMethod: string): void;
      SlaveMutiny(ATime: number): void;
      GetLastThrowItemList(): TUserItemListReadOnly;
      readonly LastThrowItemList: TUserItemListReadOnly;
      LoadUserItemFromString(S: string): TUserItem;
      SaveUserItemToString(UserItem: TUserItem): string;
      SaveItemToStore(Item: TUserItem, ChrName: string, EventType: number,
Param: string): string;
      LoadItemFromStore(Index: string): TUserItem;
      GetStoreItemType(Index: string): number;
      readonly StoreItemType: number;
      GetAllStoreItem(ChrName: string): TStoreItemList;
      readonly AllStoreItem: TStoreItemList;
      GetStoreItemsWithType(ChrName: string, EventType: number): TStoreItemList;
      readonly StoreItemsWithType: TStoreItemList;
      GetStoreItemsOnlyType(EventType: number): TStoreItemList;
      readonly StoreItemsOnlyType: TStoreItemList;
      DeleteStoreItem(Index: string): void;
      CopyUserItem(Item: TUserItem): TUserItem;
      DeleteStoreItems(IndexList: TStringList): number;
      GetIPLocal(IP: string): string;
      readonly IPLocal: string;
      QueryPerformanceCounter(): number;
      QueryPerformanceFrequency(): number;
      LoadLibrary(FileName: string): number;
      GetProcAddress(hModule: number, ProcName: string): number;
      readonly ProcAddress: number;
      FreeLibrary(hModule: number): boolean;
      ASyncHttpPostJson(Ident: string, URL: string, Json: string, TimeOut:
number): void;
      ASyncHttpGet(Ident: string, URL: string, TimeOut: number): void;
      TryGetResource(AName: string): TStringList;
      SendMail(AFrom: string, ATo: string, ASubject: string, AContent: string):
void;
      SendChangeClientSpeed(): void;
      Sleep(MS: number): void;
      GetMagicIdByName(Name: string): number;
```

```
      DirectoryExists(Name: string): boolean;
      CreateDir(Name: string): boolean;
      FileExists(Name: string): boolean;
      VarString(AVarName: string): TVarValue;
      VarInteger(AVarName: string): TVarValue;
      VarDateTime(AVarName: string): TVarValue;
      VarFloat(AVarName: string): TVarValue;
      VarBoolean(AVarName: string): TVarValue;
      CreateFastIniFile(AFileName: string): TFastIniFile;
      CreateActorList(): TActorList;
      MakeMaskString(AStr: string): string;
      GetManagerNpc(): TNormNpc;
      readonly ManagerNpc: TNormNpc;
      GetQFunctionNpc(): TNormNpc;
      readonly QFunctionNpc: TNormNpc;
      ForceDirectories(Path: string): boolean;
      DeleteFile(Path: string): boolean;
      CreateFileStream(Path: string, Mode: number): TFileStream;
      FileStreamOpenMode(): string;
      DeleteMapRoute(RouterName: string): boolean;
      GetTickCount(): number;
      readonly TickCount: number;
      CreateFastIniFromStrings(List: TStringList): TFastIniFile;
      ClearScriptCreateObject(): void;
      CreateUserItemList(AName: string, SaveToDB: boolean): TUserItemList;
      GetUserItemList(AName: string): TUserItemList;
      readonly UserItemList: TUserItemList;
      DeleteUserItemList(AName: string): void;
      CreateUserItemByName(ItemName: string): TUserItem;
      DestoryUserItem(Item: TUserItem, Force: boolean): void;
      ReadDirFiles(Path: string, Ext: string, IncludeSubDir: boolean):
TStringList;
      ASyncHttpPostJsonEx: (url: string, postJson: string, timeout: number, cb:
HttpAsyncResult) => void;
      ASyncHttpGetEx: (url: string, timeout: number, cb: HttpAsyncResult) =>
void;
      UpdatePlayerAccount(Account: string, PlayerName: string, NewAccount:
string, Log: string): void;
      GetCustomSaveStr(): string;
      CustomSaveStr: string;
      SetCustomSaveStr(Value: string): void;
      GetAdminUI(): TProjectUIAdmin;
      readonly AdminUI: TProjectUIAdmin;
      HasRobotRun(RobotName: string): boolean;
      HasDelayGoto(ID: number): boolean;
      IsvaildUserItem(Item: TUserItem): boolean;
      FixRecordMapPointItemShowName(Item: TUserItem): void;
      CreateBlockEvent(MapName: string, X: number, Y: number, Time: number,
UIEffect: number): void;
      GetActorByHandle(ActorHandle: number): TActor;
      readonly ActorByHandle: TActor;
      V: any;
      R: any;
      onNpcClicked: (Npc: TNormNpc, PlayObject: TPlayObject, ClickLabel: string,
AUnitName: string, AParams: string) => void;
      onPlayerLevelUp: (PlayObject: TPlayObject, Level: number) => void;
      onPlayerLogin: (PlayObject: TPlayObject, OnlineAddExp: boolean) => void;
      onPlayerInitialization: (PlayObject: TPlayObject) => void;
      onPlayerReconnection: (PlayObject: TPlayObject, OnlineAddExp: boolean) =>
void;
      onPlayerOffLine: (PlayObject: TPlayObject, OnlineAddExp: boolean) => void;
      onGetLineNoticeMessage: (PlayObject: TPlayObject, Message: string) =>
string;
      onEngineStart: (Text: string, NewText: number) => number;
```

```
      onPlayerHelp: (PlayObject: TPlayObject) => void;
      onPlayerHot: (PlayObject: TPlayObject) => void;
      onPlayerPayHome: (PlayObject: TPlayObject) => void;
      onPlayerReNameState: (State: number, PlayObject: TPlayObject, OldName:
string, Newname: string) => void;
      onScriptEngineInit: (isReload: boolean) => void;
      onScriptEngineFinal: (isReload: boolean) => void;
      onLogicLoopBegin: () => void;
      onLogicLoopEnd: () => void;
      onMonitorRevival: (Envir: TEnvirnoment, ActorObject: TActor, Tag: number)
=> void;
      onMonitorDie: (Envir: TEnvirnoment, ActorObject: TActor, Killer: TActor,
Tag: number) => void;
      onMonitorKill: (Envir: TEnvirnoment, ActorObject: TActor, Player:
TPlayObject, Tag: number) => void;
      onMonitorDamage: (ActorObject: TActor, ADamageSource: TActor, Tag: number,
Value: number) => void;
      onMonitorDamageEx: (ActorObject: TActor, ADamageSource: TActor, Tag:
number, SkillID: number, SkillLevel: number, Value: number) => number;
      onOpenMember: (PlayObject: TPlayObject) => void;
      onProcessCommand: (Play: TPlayObject, Command: string, Params: string,
Processed: boolean) => boolean;
      onGuildInitialize: (AGuild: TGuild) => void;
      onAsyncHttpPostResult: (Ident: string, URL: string, ResultText: string,
ErrorStr: string) => void;
      onAsyncHttpGetResult: (Ident: string, URL: string, ResultText: string,
ErrorStr: string) => void;
      onExecuteExtendButton: (Play: TPlayObject, CommandText: string) => void;
      onPlayerSpeedException: (Play: TPlayObject, Kick: boolean) => boolean;
      onScriptButtonClick: (PlayObject: TPlayObject, Flag: string) => void;
      onPlayerReAlive: (Player: TPlayObject, Event: TReAliveEvent) => void;
      onKillPlayer: (Killer: TPlayObject, Player: TPlayObject) => void;
      onSlaveKillPlayer: (Master: TPlayObject, Slave: TActor, Player:
TPlayObject) => void;
      onPlayerDie: (Player: TPlayObject, Killer: TActor) => void;
      onKillMonster: (Player: TPlayObject, Monster: TActor) => void;
      onSlaveKillMonster: (Player: TPlayObject, Slave: TActor, Monster: TActor)
=> void;
      onDuelEnd: (Winner: TPlayObject, Loser: TPlayObject) => void;
      onBagItemEvent: (Player: TPlayObject, Item: TUserItem, EventType: number,
EventID: number) => void;
      onBuyShopItem: (Player: TPlayObject, UserItem: TUserItem, Kind: number,
ShopType: number, Count: number, Price: number) => void;
      onClickSighIcon: (Player: TPlayObject, MethodID: number) => void;
      onGetBoxItem: (Player: TPlayObject, Item: TUserItem, BoxID: number) =>
void;
      onProgressEvent: (Player: TPlayObject, EventID: number) => void;
      onProgressFaild: (Player: TPlayObject, EventID: number) => void;
      onItemSoulLevelUp: (Player: TPlayObject, UserItem: TUserItem) => void;
      onActiveTitleChanged: (Player: TPlayObject) => void;
      onSideBarButtonClick: (Player: TPlayObject, AName: string) => void;
      onGuildExtendButtonClick: (Player: TPlayObject) => void;
      onBuyShopItemEnd: (Player: TPlayObject, ItemName: string, Kind: number,
ShopType: number, Count: number, ActualCount: number, Price: number) => void;
      onGuildAddedMember: (Guild: TGuild, Player: TPlayObject) => void;
      onGuildRemovedMember: (Guild: TGuild, Player: string) => void;
      onGroupAddedMember: (Owner: TPlayObject, Member: TPlayObject) => void;
      onGroupRemovedMember: (Owner: TPlayObject, Member: TPlayObject) => void;
      onBeforeGuildAddMember: (Guild: TGuild, PlayObject: TPlayObject, boAccept:
boolean) => boolean;
      onScriptAllowLevelUp: (PlayObject: TPlayObject, Level: number, Allow:
boolean) => boolean;
      onPlayerAction: (PlayObject: TPlayObject, Action: number, Allow: boolean)
=> boolean;
```

```
      onSendHotKey: (Player: TPlayObject, Akey: number, KeyCtrl: boolean,
KeyAlt: boolean, ATargetActor: TActor, X: number, Y: number) => void;
      onPlayerDropItem: (Player: TPlayObject, Item: TUserItem, X: number, Y:
number, ISUseItem: boolean, Accept: boolean) => boolean;
      onPlayerThrowItem: (Player: TPlayObject, Item: TUserItem, X: number, Y:
number, Accept: boolean) => boolean;
      onMonDropItem: (Actor: TActor, Monster: TActor, UserItem: TUserItem,
Envir: TEnvirnoment, X: number, Y: number, Accept: boolean) => boolean;
      onDropItemByMonName: (Player: TPlayObject, Monster: string, UserItem:
TUserItem, Accept: boolean) => boolean;
      onMineDropItem: (Player: TPlayObject, UserItem: TUserItem, Envir:
TEnvirnoment, X: number, Y: number, Accept: boolean) => boolean;
      onBeforeCollect: (Player: TPlayObject, AMon: TActor, Accept: boolean) =>
boolean;
      onCollect: (Player: TPlayObject, AMon: TActor, Accept: boolean) =>
boolean;
      onButchItem: (Player: TPlayObject, Monster: TActor, UserItem: TUserItem,
Accept: boolean) => boolean;
      onStdModeFunc: (Player: TPlayObject, UserItem: TUserItem, Accept: boolean)
=> boolean;
      onItemClickItem: (Player: TPlayObject, Source: TUserItem, Dest: TUserItem,
Accept: boolean) => boolean;
      onPickupItem: (Player: TPlayObject, Envir: TEnvirnoment, UserItem:
TUserItem, Accept: boolean) => boolean;
      onItemClickUseItem: (Player: TPlayObject, ItemWhere: TItemWhere, Source:
TUserItem, Dest: TUserItem, Accept: boolean) => boolean;
      onTakeOnItem: (Player: TPlayObject, UserItem: TUserItem, ItemWhere:
TItemWhere, Accept: boolean) => boolean;
      onTakeOffItem: (Player: TPlayObject, UserItem: TUserItem, ItemWhere:
TItemWhere, Accept: boolean) => boolean;
      onAfterTakeOnItem: (Player: TPlayObject, TakeOnUserItem: TUserItem,
TakeOffItem: TUserItem, ItemWhere: TItemWhere) => void;
      onAfterTakeOffItem: (Player: TPlayObject, TakeOffItem: TUserItem,
ItemWhere: TItemWhere) => void;
      onGetExp: (Player: TPlayObject, Exp: number, ResultExp: number) => number;
      onGetExpEx: (Player: TPlayObject, ExpActor: TActor, Exp: number,
ResultExp: number) => number;
      onRefineButtonClick: (Player: TPlayObject, Handled: boolean) => boolean;
      onRefineItem: (Player: TPlayObject, Item1: TUserItem, Item2: TUserItem,
Item3: TUserItem, Handled: boolean) => boolean;
      onBeforeGuildRemoveMember: (Guild: TGuild, Player: string, Accept:
boolean) => boolean;
      onStallPutOn: (Player: TPlayObject, Item: TUserItem, GoldPrice: number,
GameGoldPrice: number, GamePoint: number, Accept: boolean) => boolean;
      onStallPutOff: (Player: TPlayObject, Item: TUserItem, GoldPrice: number,
GameGoldPrice: number, GamePoint: number, Accept: boolean) => boolean;
      onStallBuyItem: (Player: TPlayObject, StallName: string, Item: TUserItem,
Gold: number, GameGold: number, GamePoint: number, Accept: boolean) => boolean;
      onStallUpdateItem: (Player: TPlayObject, Item: TUserItem, OldGold: number,
OldGameGold: number, OldGamePoint: number, Gold: number, GameGold: number,
GamePoint: number, Accept: boolean) => boolean;
      onStallExtractGold: (Player: TPlayObject, Gold: number, GameGold: number,
GamePoint: number, Accept: boolean) => boolean;
      onAfterStopStall: (Player: TPlayObject) => void;
      onBeforeStartStall: (Player: TPlayObject, Accept: boolean) => boolean;
      onMailBeforeSend: (Player: TPlayObject, SendTo: string, Item: TUserItem,
Accept: boolean) => boolean;
      onMailAfterSend: (Player: TPlayObject, SendTo: string) => void;
      onMailReceived: (Player: TPlayObject, MailFrom: string, Subject: string)
=> void;
      onSlaveLevelUp: (Master: TActor, Slave: TActor, NewLevel: number, Accept:
boolean) => boolean;
      onPlayAddSkill: (Player: TPlayObject, Magic: TUserMagic, Accept: boolean)
=> boolean;
```

```
      onMagicSpell: (Player: TPlayObject, UserMagic: TUserMagic, nTargetX:
number, nTargetY: number, Target: TActor, Handled: boolean) => boolean;
      onPlayerAttack: (Player: TPlayObject, UserMagic: TUserMagic, Target:
TActor, Accept: boolean) => boolean;
      onRideOnHorse: (Player: TPlayObject, RideOn: boolean, Accept: boolean) =>
boolean;
      onCustomMessage: (Player: TPlayObject, AMessageToken: number, AParam1:
number, AParam2: number, AParam3: number, AData: string) => void;
      onChangeAttackMode: (Player: TPlayObject, OldMode: TAttackMode, NewMode:
TAttackMode) => TAttackMode;
      onAltAndLButtonClickBagItem: (Player: TPlayObject, Item: TUserItem) =>
void;
      onDiceEvent: (Player: TPlayObject, ATag: number, APoint1: number, APoint2:
number, APoint3: number) => void;
      onBeforeOpenStall: (Player: TPlayObject, AAccept: boolean) => boolean;
      onChangeToMonsterStart: (Actor: TActor, TargetMonName: string, nSec:
number) => void;
      onChangeToMonsterEnd: (Actor: TActor, TargetMonName: string) => void;
      onUseExpStoneItem: (Player: TPlayObject, Item: TUserItem, CanUse: boolean)
=> boolean;
      onClickGameShop: (PlayObject: TPlayObject, OpenDefalut: boolean) =>
boolean;
      onGetLevelExp: (Level: number, Exp: number) => number;
      onGetPlayerLevelMaxExp: (PlayObject: TPlayObject, Level: number, MaxExp:
number) => number;
      onAddBuff: (Actor: TActor, Buff: TBuff, Accept: boolean) => boolean;
      onRemoveBuff: (Actor: TActor, Buff: TBuff) => void;
      onCustomBuffAct: (Actor: TActor, Buff: TBuff) => void;
      onChatMessage: (PlayObject: TPlayObject, Message: string, Allow: boolean)
=> boolean;
      onCustomProgressEvent: (FunctionName: string, Npc: TNormNpc, Player:
TPlayObject, Args: string) => void;
      onLockClientPassword: (FunctionName: string, Player: TPlayObject,
Password: string, Args: string) => void;
      onAltAndLButtonClickUseItem: (Player: TPlayObject, Item: TUserItem,
AItemWhere: number) => void;
      onPickUpItemChangePicker: (PickSender: TPlayObject, Item: TUserItem, Gold:
number, WhoPicker: TPlayObject, ItemState: number) => [number, TPlayObject,
number];
      onCheckEnterMap: (Play: TPlayObject, SourceEnvir: TEnvirnoment, DestEnvir:
TEnvirnoment, SourceX: number, SourceY: number, DestX: number, DestY: number,
Accept: boolean) => boolean;
      onPlayerEnterMap: (Play: TPlayObject, SourceEnvir: TEnvirnoment,
DestEnvir: TEnvirnoment, SourceX: number, SourceY: number, DestX: number, DestY:
number) => void;
      onCloseDuplicatesMap: (Envir: TEnvirnoment) => void;
      onDeleteRoute: (AName: string) => void;
      onExternalNpcExecCommand: (Command: string, Args: string) => void;
      onGlobalEventExecute: (TimeID: number) => void;
      onGlobalEventRemove: (TimeID: number) => void;
      onPrivyEventExecute: (Play: TPlayObject, TimeID: number) => void;
      onPrivyEventRemove: (Play: TPlayObject, TimeID: number) => void;
      onMagicNpcExecute: (AMethod: string, ASource: TActor, ATarget: TActor,
ATargetX: number, ATargetY: number, AMouseX: number, AMouseY: number, AList:
TActorList, AMagic: TUserMagic, Processed: boolean) => boolean;
      onMonSelectMagicBeforeAttack: (AMon: TActor, ATarget: TActor, AMagicID:
number) => number;
      onSkillActionExecute: (Source: TActor, Target: TActor, AMethod: string,
nParam1: number, nParam2: number, sParam1: string, sParam2: string, MouseX:
number, MouseY: number) => void;
      onSkillActorSelConditionCustom: (Source: TActor, Target: TActor, AMethod:
string, N1: number, N2: number, N3: number, Processed: boolean) => boolean;
      onSkillUseCondtionCustom: (Source: TActor, AMethod: string, N1: number,
N2: number, N3: number, Consume: boolean, Processed: boolean) => boolean;
```

```
	onMissionFinish: (Player: TPlayObject, AMission: TMissionItem) => void;
	onAINpcExecute: (Actor: TActor) => void;
	onScriptEngineError: (E: Error) => void;
	onActorGhost: (Actor: TActor) => void;
	onQueryUserState: (Player: TPlayObject, QueryTarget: TActor) => void;
	onAsyncDBQueryResult: (Ident: string, DBName: string, SQL: string,
ErrorDesc: string, DataSet: TDataSet) => void;
	onAsyncDBExecResult: (Ident: string, DBName: string, SQL: string,
ErrorDesc: string, RowAffected: number) => void;
	onPlayerSaveCustomSaveStr: (Player: TPlayObject, isOfflineSave: boolean)
=> void;
	onPlayerLoadCustomSaveStr: (Player: TPlayObject) => void;
	onUpdatePlayerAccountResult: (OldAccount: string, PlayerName: string,
NewAccount: string, Sucess: boolean, ErrorDesc: string, Log: string) => void;
	onLoginResetData: (Player: TPlayObject, isOnlineAddExp: boolean) => void;
	onEngineStop: (M2Core: TM2Core) => void;
	onAdminUIButtonClick: (ButtonName: string) => void;
	onAdminUIEditOk: (Name: string, InputText: string) => void;
	onAdminUICheckBoxChange: (Name: string, Checked: boolean) => void;
	onClickLockedBagGrid: (Player: TPlayObject, GridIndex: number) => void;
	onStartAutoFight: (Player: TPlayObject) => void;
	onStopAutoFight: (Player: TPlayObject) => void;
	onUIActivedBagItemEvent: (Player: TPlayObject, UIName: string,
ClientItemBagIndex: number, Item: TUserItem, KeyCtrl: boolean, KeyAlt: boolean,
MouseButton: TMouseButton) => void;
	onClientBleedNumber: (Player: TPlayObject, Actor: TActor, DamageSource:
TActor, BleedNum: number, NumType: TDamageValueType, MagID: number, MagLv:
number, BleedType: TBleedType, Allow: boolean) => boolean;
	onAttackActorDamageIsZero: (Attacker: TActor, Target: TActor, Reason:
number, Damage: number) => number;
	onSlaveRoyalOver: (Slave: TActor, Reason: number, Accept: boolean) =>
boolean;
	onUseMedicine: (Player: TPlayObject, Item: TUserItem, Accept: boolean) =>
boolean;
	onThrowItem: (Map: TEnvirnoment, X: number, Y: number, CallBackIdent:
number, Item: TUserItem, Owner: TActor) => void;
	onCastleStartWar: (Castle: TUserCastle) => void;
	onCastleStopWar: (Castle: TUserCastle) => void;
	onCastleSeized: (Castle: TUserCastle, Guild: TGuild) => void;
	onSlaveCanPickUpItem: (Slave: TActor, Item: TMapItem, Accept: boolean) =>
boolean;
	onSlavePickUpItem: (Slave: TActor, Item: TMapItem, PickUpResult:
TPickUpResult) => TPickUpResult;
	onBeforePlayerReAlive: (Player: TPlayObject, Event: TReAliveEvent) =>
boolean;
	onRightClickMapPos: (Player: TPlayObject, MapX: number, MapY: number) =>
void;
	onActorReachedMissionXY: (Actor: TActor, MissionX: number, MissionY:
number, Range: number) => void;
	onQueryClientQuickBarItemCountResult: (Player: TPlayObject, Ident: string,
Count: number) => void;

}
declare class TVarValue {
	GetName(): string;
	readonly Name: string;
	GetVarType(): number;
	readonly VarType: number;
	GetAsString(): string;
	AsString: string;
	SetAsString(Value: string): void;
	GetAsFloat(): number;
	AsFloat: number;
	SetAsFloat(Value: number): void;
```

```
        GetAsInteger(): number;
        AsInteger: number;
        SetAsInteger(Value: number): void;
        Save(): void;
        GetAsDateTime(): number;
        AsDateTime: number;
        SetAsDateTime(Value: number): void;
        GetAsBoolean(): boolean;
        AsBoolean: boolean;
        SetAsBoolean(Value: boolean): void;

    }
    declare class TToptenz {
        GetRichest(): TNumOrderItems;
        readonly Richest: TNumOrderItems;
        GetLevels(): TLevelOrderItems;
        readonly Levels: TLevelOrderItems;
        GetMaster(): TNumOrderItems;
        readonly Master: TNumOrderItems;
        GetEfficiency(): TNumOrderItems;
        readonly Efficiency: TNumOrderItems;
        GetWarrior(): TJobOrderItems;
        readonly Warrior: TJobOrderItems;
        GetWizard(): TJobOrderItems;
        readonly Wizard: TJobOrderItems;
        GetTaoist(): TJobOrderItems;
        readonly Taoist: TJobOrderItems;
        GetArcher(): TJobOrderItems;
        readonly Archer: TJobOrderItems;
        GetAssassin(): TJobOrderItems;
        readonly Assassin: TJobOrderItems;
        GetShaman(): TJobOrderItems;
        readonly Shaman: TJobOrderItems;
        Refresh(): void;

    }
    declare class TJobOrderItems extends TOrderItems {
        GetMinValue(AIndex: number): number;
        readonly MinValue: number;
        GetMaxValue(AIndex: number): number;
        readonly MaxValue: number;

    }
    declare class TLevelOrderItems extends TOrderItems {
        GetReLevel(AIndex: number): number;
        readonly ReLevel: number;
        GetLevel(AIndex: number): number;
        readonly Level: number;

    }
    declare class TNumOrderItems extends TOrderItems {
        GetValue(AIndex: number): number;
        readonly Value: number;

    }
    declare class TOrderItems {
        GetCount(): number;
        readonly Count: number;
        GetName(AIndex: number): string;
        readonly Name: string;
        GetJob(AIndex: number): number;
        readonly Job: number;
        GetGender(AIndex: number): number;
        readonly Gender: number;
```

```
      IndexOf(AName: string): number;

}
declare class TFastIniFile {
      ReadInteger(Sect: string, Name: string, Def: number): number;
      WriteInteger(Sect: string, Name: string, Value: number): void;
      ReadString(Sect: string, Name: string, Def: string): string;
      WriteString(Sect: string, Name: string, Value: string): void;
      ReadBoolean(Sect: string, Name: string, Def: boolean): boolean;
      WriteBoolean(Sect: string, Name: string, Value: boolean): void;
      SectionExists(Sect: string): boolean;
      ReadFixedDateTime(Sect: string, Name: string, Value: number): number;
      UpdateFile(): void;
      EraseSection(Sect: string): void;
      Free: () => void;
      GetInlineCommentsEnabled(): boolean;
      InlineCommentsEnabled: boolean;
      SetInlineCommentsEnabled(Value: boolean): void;
      ReadSections(AList: TStringList): void;
      GetStrings(AList: TStringList): void;
      ReadSectionValues(Name: string, AList: TStringList): void;

}
declare class TStringList {
      GetCount(): number;
      readonly Count: number;
      GetStrings(Index: number): string;
      readonly Strings: string;
      LoadFromFile(AFileName: string): void;
      SaveToFile(AFileName: string): void;
      Delete(Index: number): void;
      Clear(): void;
      IndexOf(Text: string): number;
      Add(Text: string): number;
      Free: () => void;
      SetCommaText(Text: string): void;
      GetCommaText(): string;
      CommaText: string;
      SetDelimiter(Text: string): void;
      GetDelimiter(): string;
      Delimiter: string;
      SetDelimitedText(Text: string): void;
      GetDelimitedText(): string;
      DelimitedText: string;
      Insert(Index: number, Text: string): void;
      GetText(): string;
      Text: string;
      SetText(Text: string): void;
      GetLineBreak(): string;
      LineBreak: string;
      SetLineBreak(Value: string): void;
      SetValues(Index: string, Value: string): void;
      GetValues(Index: string): string;
      Values: string;
      GetNames(Index: number): string;
      GetValueFromIndex(Index: number): string;
      SetValueFromIndex(Index: number, Value: string): void;

}
declare class TItemState {
      GetDisableMake(): boolean;
      DisableMake: boolean;
      SetDisableMake(Value: boolean): void;
      GetWriteLog(): boolean;
```

```
WriteLog: boolean;
SetWriteLog(Value: boolean): void;
GetDropHint(): boolean;
DropHint: boolean;
SetDropHint(Value: boolean): void;
GetButchHint(): boolean;
ButchHint: boolean;
SetButchHint(Value: boolean): void;
GetNoPickUp(): boolean;
NoPickUp: boolean;
SetNoPickUp(Value: boolean): void;
GetBoxHint(): boolean;
BoxHint: boolean;
SetBoxHint(Value: boolean): void;
GetShowNameClient(): boolean;
ShowNameClient: boolean;
SetShowNameClient(Value: boolean): void;
GetSpecialShow(): boolean;
SpecialShow: boolean;
SetSpecialShow(Value: boolean): void;
GetAutoPickUp(): boolean;
AutoPickUp: boolean;
SetAutoPickUp(Value: boolean): void;
GetNerverDrop(): boolean;
NerverDrop: boolean;
SetNerverDrop(Value: boolean): void;
GetBind(): boolean;
Bind: boolean;
SetBind(Value: boolean): void;
GetNoRepair(): boolean;
NoRepair: boolean;
SetNoRepair(Value: boolean): void;
GetNoStore(): boolean;
NoStore: boolean;
SetNoStore(Value: boolean): void;
GetOfflineFree(): boolean;
OfflineFree: boolean;
SetOfflineFree(Value: boolean): void;
GetNoDrop(): boolean;
NoDrop: boolean;
SetNoDrop(Value: boolean): void;
GetDeathFree(): boolean;
DeathFree: boolean;
SetDeathFree(Value: boolean): void;
GetDeathDrop(): boolean;
DeathDrop: boolean;
SetDeathDrop(Value: boolean): void;
GetNoTakeOff(): boolean;
NoTakeOff: boolean;
SetNoTakeOff(Value: boolean): void;
GetAutoBindAfterTakeOn(): boolean;
AutoBindAfterTakeOn: boolean;
SetAutoBindAfterTakeOn(Value: boolean): void;
GetCallMethodOnMonDrop(): boolean;
CallMethodOnMonDrop: boolean;
SetCallMethodOnMonDrop(Value: boolean): void;
GetCallMethodOnPlayerDrop(): boolean;
CallMethodOnPlayerDrop: boolean;
SetCallMethodOnPlayerDrop(Value: boolean): void;
GetCallMethodOnButch(): boolean;
CallMethodOnButch: boolean;
SetCallMethodOnButch(Value: boolean): void;
GetCallMethodOnPickUp(): boolean;
CallMethodOnPickUp: boolean;
```

```
	SetCallMethodOnPickUp(Value: boolean): void;
	GetCanInQuickBar(): boolean;
	CanInQuickBar: boolean;
	SetCanInQuickBar(Value: boolean): void;
	GetCanInJewelryBox(): boolean;
	CanInJewelryBox: boolean;
	SetCanInJewelryBox(Value: boolean): void;

}
declare class TActorList {
	GetCount(): number;
	readonly Count: number;
	Actor(index: number): TActor;
	Add(Item: TActor): number;
	Delete(Index: number): void;
	Free: () => void;

}
declare class TList {
	GetCount(): number;
	readonly Count: number;
	Items(index: number): Object;

}
declare class TDateUtils {
	MinuteOfTheHour(DateTime: number): number;
	SecondSpan(TimeA: number, TimeB: number): number;
	MilliSecondSpan(TimeA: number, TimeB: number): number;
	StrToDateTimeDef(format: string, Time: number): number;
	DaysBetween(TimeA: number, TimeB: number): number;
	MinutesBetween(TimeA: number, TimeB: number): number;
	HoursBetween(TimeA: number, TimeB: number): number;
	MilliSecondsBetween(TimeA: number, TimeB: number): number;
	DecodeDateTime: (DateTime: number) => TimeDef;
	EncodeDateTime(Y: number, M: number, D: number, H: number, N: number, S:
number, MS: number): number;
	DateTimeToStr(Time: number): string;
	Now(): number;
	IncDay(Time: number, ANumberOfDays: number): number;
	IncHour(Time: number, ANumberOfHour: number): number;
	IncMinute(Time: number, ANumberOfMinute: number): number;
	SecondsBetween(TimeA: number, TimeB: number): number;
	IncSecond(Time: number, ANumberOfSec: number): number;
	DateTimeToFileDate(Time: number): number;
	DayOfTheWeek(Time: number): number;
	DayOfWeek(Time: number): number;
	FormatDateTime(FormatStr: string, Time: number): string;
	StrToDateTime(TimeStr: string): number;
	FileDateToDateTime(FileDateTime: number): number;
	DateTimeToTimeStamp(Time: number): TTimeStamp;
	FreeTimeStamp(Time: TTimeStamp): void;
	StartOfTheMonth(Time: number): number;
	DaysInAMonth(Year: number, Month: number): number;
	UnixTimeToDateTime(UnixTime: number, AReturnUtc: boolean): number;
	DateTimeToUnixTime(AValue: number, AInputIsUTC: boolean): number;
	MilliSecondOf(AValue: number): number;
	MinuteOf(AValue: number): number;
	SecondOf(AValue: number): number;
	HourOf(AValue: number): number;
	DayOf(AValue: number): number;
	WeekOf(AValue: number): number;
	MonthOf(AValue: number): number;
	YearOf(AValue: number): number;
```

```
}
declare class TFileStream {
      WriteStrAnsi(Str: string): void;
      WriteBuffer(PBuffer: Object, Size: number): void;
      ReadBuffer(PBuffer: Object, Size: number): void;
      GetSize(): number;
      Size: number;
      SetSize(Value: number): void;
      GetPosition(): number;
      Position: number;
      SetPosition(Value: number): void;
      Free(): void;

}
declare class TMagic {
      GetwSpell(): number;
      readonly wSpell: number;
      GetbtDefSpell(): number;
      readonly btDefSpell: number;

}
declare class TTimeStamp {
      GetTime(): number;
      Time: number;
      SetTime(Value: number): void;
      GetDate(): number;
      Date: number;
      SetDate(Value: number): void;

}
declare class TDBEngine {
      ExecSQL(Name: string, SQLText: string): number;
      Query(Name: string, SQLText: string): TDataSet;
      QueryAsync(Ident: string, Name: string, SQLText: string): void;
      ExecAsync(Ident: string, Name: string, SQLText: string): void;
      ValueList(Name: string, SQLText: string): TStringList;
      BeginTran(Name: string): void;
      CommitTran(Name: string): void;
      RollbackTran(Name: string): void;
      AddConection(Name: string, DBServer: string, DBPort: number, DBUser:
string, DBPassword: string, DataBase: string): boolean;
      QueryAsyncEx: (name: string, sql: string, cb: QueryDBAsyncResult) => void;
      ExecAsyncEx: (name: string, sql: string, cb: ExecDBAsyncResult) => void;

}
declare class TDataSet {
      Open(): void;
      IsEmpty(): boolean;
      Eof(): boolean;
      Next(): void;
      FieldByName(Name: string): TField;
      GetRecordCount(): number;
      readonly RecordCount: number;
      Free(): void;
      GetFieldCount(): number;
      readonly FieldCount: number;
      GetField(Index: number): TField;

}
declare class TField {
      GetFieldName(): string;
      readonly FieldName: string;
      GetAsString(): string;
      readonly AsString: string;
```

```
	GetAsInteger(): number;
	readonly AsInteger: number;
	GetAsLargeInt(): number;
	readonly AsLargeInt: number;
	GetAsDateTime(): number;
	readonly AsDateTime: number;
	GetAsBoolean(): boolean;
	readonly AsBoolean: boolean;
	GetAsFloat(): number;
	readonly AsFloat: number;
	IsInt(): boolean;
	IsLargeInt(): boolean;
	IsFloat(): boolean;
	GetFieldType(): number;
	readonly FieldType: number;
	GetRawFieldType(): number;
	readonly RawFieldType: number;
	IsDateTime(): boolean;

}
declare class TMonster extends TActor {
	Run(): void;
	AttackTarget(): boolean;
	SearchTarget(): void;
	DelTargetCreat(): void;
	SelectTarget(): void;
	Initialize(): void;
	MakeGhost(): void;
	OnSetTargetCreat(Target: TActor): void;
	OnDelTargetCreat(Target: TActor): void;
	Say(PMessage: string): void;
	InViewRange(Target: TActor): boolean;
	ReAlive(): void;
	RecalcAbilitys(): void;
	Die(): void;
	ReadAddedAbility(PAblity: TAddedAbility): void;
	IsProtectTarget(Target: TActor): boolean;
	IsAttackTarget(Target: TActor): boolean;
	IsProperTarget(Target: TActor): boolean;
	IsProperFriend(Target: TActor): boolean;
	GetCustomRace(): number;
	readonly CustomRace: number;

}
declare class TUserItemList {
	DontAddToOtherListIfWantNeedMakeCopyGet(Index: number): TUserItem;
	MakeCopyAndAdd(Item: TUserItem): number;
	DeleteAndFreeItem(Index: number): void;
	GetCount(): number;
	readonly Count: number;

}
declare class TMissionLinkItem {
	GetKind(): number;
	Kind: number;
	GetMisssionID(): string;
	MisssionID: string;
	GetSubject(): string;
	Subject: string;
	GetContent(): string;
	Content: string;
	GetLevel(): number;
	Level: number;
	GetReLevel(): number;
```

```
      ReLevel: number;
      GetTargetNpc(): number;
      TargetNpc: number;
      SetKind(Value: number): void;
      SetMisssionID(Value: string): void;
      SetSubject(Value: string): void;
      SetContent(Value: string): void;
      SetLevel(Value: number): void;
      SetReLevel(Value: number): void;
      SetTargetNpc(Value: number): void;
      Delete(): void;

}
declare class TProjectUIAdmin {
      AddPage(PageName: string, CWidth: number, CHeight: number, CRowCount:
number): void;
      AddButton: (PageName: string, ControlName: string, Hint?: string,
confirm?: boolean) => void;
      AddCheckBox: (PageName: string, ControlName: string, Checked: boolean,
Hint?: string) => void;
      AddEdit: (PageName: string, ControlName: string, Text: string, Hint?:
string, confirm?: boolean) => void;
      AddLogPage(PageName: string): void;
      AddLog(PageName: string, LogText: string): void;
      UpdateInputText(ControlName: string, LogText: string): void;
      UpdateCheckBoxState(ControlName: string, Checked: boolean): void;
      ClearLogPage(PageName: string): void;

}
declare class TUserItemListReadOnly {
      GetItem(Index: number): TUserItem;
      GetCount(): number;
      readonly Count: number;

}
declare class TMapItem {
      GetName(): string;
      readonly Name: string;
      GetUserItem(): TUserItem;
      readonly UserItem: TUserItem;
      GetMoneyType(): number;
      readonly MoneyType: number;
      GetMapX(): number;
      readonly MapX: number;
      GetMapY(): number;
      readonly MapY: number;
      GetCount(): number;
      readonly Count: number;

}
declare class TMobilePlatform {
      AddFunctionButton(Player: TPlayObject, FunctionTag: number, IconFile:
string, ScriptFlag: string, ShowDock: boolean): void;
      DelFunctionButton(Player: TPlayObject, FunctionTag: number): void;
      SetFunctionLayout: (Player: TPlayObject, Layouts: IMobileFunctionButton[])
=> void;
      IsMobileClient(Player: TPlayObject): boolean;

}

declare const GameLib: TM2Core;
declare const DateUtils: TDateUtils
declare const MobilePlatform: TMobilePlatform
```