# LuaActionSupport（卡牌action的lua支持）

## 用法

在ActionName的ParentObjectID里写lua代码

如果是CardAction，ActionName的LocalizationKey以LuaCardAction开头启用lua支持

如果是CardOnCardAction，ActionName的LocalizationKey以LuaCardOnCardAction开头启用lua支持

#### 术语说明

额外参数，位于lua api函数最后，参数名为ext，参数类型为LuaTable的可选参数，用于传递使用较少，数量较多的一些参数

### luaAction简单示例

改stat  
##  
SimpleAccessTool[对应stat的uid].StatValue = SimpleAccessTool[对应stat的uid].StatValue \* 2   
SimpleAccessTool[对应stat的uid]. StatRate= SimpleAccessTool[对应stat的uid]. StatRate\* 2  
##  
  
生成卡或Encounter  
##  
SimpleAccessTool[卡id或encounter的id].Gen(生成次数,不填生成一次)  
##

## SimpleAccessTool快速工具

索引器 参数类型 string 传入uid

返回SimpleUniqueAccess

## SimpleUniqueAccess---UniqueIDScriptable快速操作接口

#### 方法Gen ：

参数count 生成次数

UniqueIDScriptable为carddata时生成count张对应卡

UniqueIDScriptable为encounter时生成一次encounter

额外参数：同CardAccessBridge的AddCard

索引器：传入字符串key，获取对应字段的值或者修改对应字段

获取值时返回SimpleObjAccess或SimpleUniqueAccess（如果获取的是UniqueIDScriptable），访问AccessObj可得到实际值

属性CardDescription 可读写字符串属性，修改卡牌描述

属性StatValue和StatRate 若对应UniqueIDScriptable为gamestat，获取或修改状态的值

子属性~Min，~Max，浮点数，修改对应的上下限

属性AccessObj，获取正在访问的对象

## SimpleObjAccess

索引器：同SimpleUniqueAccess

属性AccessObj，获取正在访问的对象

## 注意

全局表会重置，临时保存数据用全局表中的ModData表

## Debug（调试）

全局变量debug

info，debug，warn，error

为debug全局变量的这几个字段赋值会自动用UnityEngine.Debug类输出赋值的值

读取这几个字段会导致报错

## CardAccessBridge（卡牌访问桥）

全局变量env

当前Environment卡（与左上角区分，该卡游戏中不可见）

全局变量exp

当前Explorable卡（左上角那张卡）

全局变量weather

当前天气卡

在CardAction中

全局变量receive

接受action的卡

在CardOnCardAction中

全局变量receive

接受action的卡

全局变量given

提供给receive的卡

### CardAccessBridge的属性

CardType：卡牌类型，是字符串，只读

Weight：卡的总重，是单浮点数，只读

Id：卡的uid，是字符串，只读

Spoilage，Usage，Fuel，Progress，Special1，Special2，Special3，Special4

八个耐久度，是浮点数，可读可写

LiquidQuantity

流体容器所含流体的数量，300==一碗，可读可写

### CardAccessBridge的函数

## 索引器

参数 int index

访问的卡为容器时，获取容器内对应槽位的卡 类型CardAccessBridge的列表

参数 string key

从卡的DroppedCollections（存档数据）读写内容

设置的值直接为整数时，直接以key保存整数值

其他情况下以key为zender.luaSupportData.{key}:{value}值为（1，1）保存

#### HasTag：

参数：string tag

返回值：bool

返回卡牌是否包含名称为参数tag的cardtag

#### AddCard：

参数：string id，int amount=1

无返回值

以本卡为基础生成uid为id的卡牌，若id对应卡牌为液体，amout代表流体量，否则amout代表生成次数

额外参数：

TransferedDurabilities部分：

Usage

Fuel

Spoilage

ConsumableCharges

Liquid

Special1

Special2

Special3

Special4

对应同名字段的值

LiquidCard：SimpleUniqueAccess类型，伴随生成什么流体

#### Remove：

参数：bool doDrop

无返回值

删除所访问的卡，doDrop为true时会掉落容器内物品（还有其他用处）

## GameStatAccessBridge（游戏状态访问桥）

### 属性

Value：读取时返回状态当前值（考虑modifier），赋值时修改基础值而非modifier值

Rate：读取时返回状态当前速率（考虑modifier），赋值时修改基础速率而非modifier值

### gameManager（GameManager.Instance）

## 持久化存储辅助内容

使用Load\*函数加载表时返回DataNodeTableAccessBridge（表访问桥）

DataNodeTableAccessBridge有一个索引器：传入字符串的key，读写持久化数据

修改与持久化数据同步

## 全局变量

### Ret 用于返回值

在动态卡牌描述中，Ret[“ret”]为要将描述改为的内容,不填就不变

在lua卡牌action中,Ret[“result”]为lua代码执行后要额外等待多少tp,不填就不等待

## 全局函数

*SaveCurrentSlot* ：参数字符串 key，（数字或字符串） val 将key，val保存到与当前存档槽绑定的数据表

*SaveGlobal* ：参数字符串 key，（数字或字符串） val 将key，val保存到全局数据表

*LoadCurrentSlot* ：参数字符串key 返回 数字或字符串 val 从与当前存档槽绑定的数据表中读取key对应的val

*LoadGlobal* ：参数字符串key 返回 数字或字符串 val 从全局数据表读取key对应的val

GetCard：参数 string id，返回值CardData

返回传入id对应卡的CardData

GetGameCard：返回找到的第一个id对应卡牌，是CardAccessBridge

GetGameCards：返回找到的id对应的所有卡牌，是CardAccessBridge的列表

额外参数：

type：字符串，以下为内容与效果对应

Equipment：只返回装备的卡

Hand：只返回从上到下第三行的卡

Base：只返回从上到下第二行的卡

Location：只返回从上到下第一行的卡

Inventory：只返回容器中的卡

GetStat：返回id对应GameStat

GetGameStat：返回id对应GameStatAccessBridge

CountCardOnBoard：string id, bool \_CountInInventories = true, bool \_CountInBackground = false

\_CountInInventories:是否统计容器内的卡

\_CountInBackground:是否统计在背景中的卡

返回id对应卡的数量

CountCardInBase：string id, bool \_CountInInventories = true

\_CountInInventories:是否统计容器内的卡

返回id对应卡在base上的数量

CountCardInHand：string id, bool \_CountInInventories = true

\_CountInInventories:是否统计容器内的卡

返回id对应卡在手上的数量

CountCardInLocation：string id, bool \_CountInInventories = true

\_CountInInventories:是否统计容器内的卡

返回id对应卡在location上的数量

CountCardEquipped：string id

只统计装备的卡，返回id对应卡数量

# LuaRegister

全局变量Register：LuaRegister类型

函数Reg：string klass, string method, string uid, LuaFunction function

klass：要patch的类的名字，如InGameCardBase

method：要patch的函数的名字，如CanReceiveInInventory

uid：如果patch的函数基于UniqueIDScriptable类型对象执行，那么只有uid匹配，注册的func才会被用到，其他情况另行讨论（暂无）

InGameCardBase：

CanReceiveInInventory：返回是否可以装入某个卡

LuaFunction要求：输入CardAccessBridge this，SimpleUniqueAccess card，SimpleUniqueAccess liquid 返回bool或nil，bool则修改函数结果，nil函数结果不变

this为执行CanReceiveInInventory本身的卡，card为要装入的物品的carddata，liquid为装入流体容器时，流体的carddata

CanReceiveInInventoryInstance：

LuaFunction要求：输入CardAccessBridge this，InGameCardBase \_Card 返回bool或nil

Card为要装入的卡的实例

InventoryWeight：返回重量

LuaFunction要求：输入CardAccessBridge this，float \_\_result 返回数字或nil，为nil则不修改

\_\_result为当前计算得的重量，

CardName：卡牌名称 CardDescription：卡牌描述

LuaFunction要求：输入CardAccessBridge this，bool \_IgnoreLiquid返回字符串或nil

\_IgnoreLiquid参数由游戏本身传入

DismantleActionButton：

Setup：卡牌的按钮初始化函数

LuaFunction要求：输入CardAccessBridge this，DismantleCardAction \_Action，CardAccessBridge \_Card，string \_Action.ActionName.LocalizationKey

\_Action：用来初始化按钮的action实例，\_Card要初始化按钮的卡牌实例，\_Action.ActionName.LocalizationKey，用于区分同个卡牌的不同按钮，action名的翻译key

GameManager：

ChangeStatValue：修改状态值的函数

LuaFunction要求：输入GameManager this，GameStatAccessBridge \_Stat，float \_Value，StatModification \_Modification 无返回值

\_Stat：被修改的状态，\_Value：往状态上加上多少，\_Modification：以什么方式修改，枚举类型，大概会变成数字

可以在这里进行状态修改，生成卡等非即时操作

ChangeStatRate：修改状态速率的函数

LuaFunction要求：输入GameManager this，GameStatAccessBridge \_Stat，float \_Rate，StatModification \_Modification 无返回值

# LuaCodeCardDescription

若卡牌的描述本该显示为如下（包括前后两行）

###luaAction CardDescription  
Ret[“ret”] = "test"  
###

则卡牌的描述会显示为

test

在

###luaAction CardDescription

和  
###

全局表中的receive是对应的卡实体

之间的内容会作为lua代码执行，其第一个返回值就是要显示的内容