脚本编写参考——UserPatch 1.5版

— 翔鹰帝国论坛荣誉出品 —

**发布地址：**[UserPatch 1.5 脚本编写参考](http://www.hawkaoe.net/bbs/thread-146439-1-1.html" \o "【可用于检查本文是否有新版本】)。本文档版本：2020.12.25。最新版和修改日志请前往发布地址获取。

**参考资料：**本文大量参考[Reference.html](http://userpatch.aiscripters.net/reference.html)、[UserPatchConst.per](http://www.hawkaoe.net/bbs/thread-145196-1-1.html)、[airef.github.io](https://airef.github.io/)、[UserPatch 1.3 脚本编写参考](http://www.hawkaoe.net/bbs/thread-112481-1-1.html)。

**（注：要查看目录请到 [视图] 里开启 [导航窗格] ）**

# 编者按：

**如何学习UserPatch 1.5的AI体系——**

UserPatch 1.5 相对于C版有了质的提升，AI能做的事情多了很多（比如[这个贴](https://www.hawkaoe.net/bbs/thread-145510-1-1.html)中的“AI人工智能”部分中所罗列的应用），你不妨先阅读一番[闪闪发光新时代——翔鹰1.4版 新AI效果示例（第二弹）](闪发光新时代——翔鹰1.4版%20新AI效果示例（第二弹）)这一战役AI教程，也可以尝试与[Beginner这个仿人类战术的发展AI](https://www.hawkaoe.net/bbs/thread-144442-1-1.html)对战一番，相信体验之后你会对UserPatch人工智能有更浓厚的兴趣。

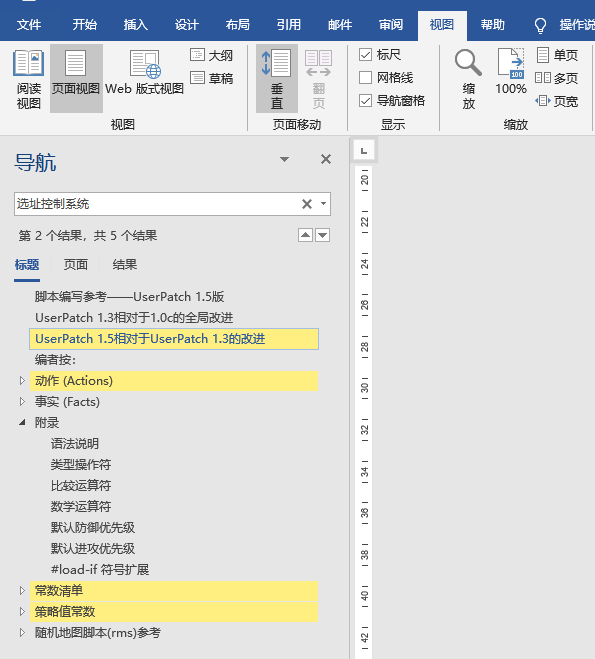
如果你对UserPatch人工智能有了学习的兴趣，我推荐先认真阅读[UserPatch人工智能简单教程](https://www.hawkaoe.net/bbs/thread-118795-1-1.html)，这有助于让你快速入门UserPatch系列的AI体系；[DUC(直接单位控制)机翻教程](https://www.hawkaoe.net/bbs/thread-146280-1-1.html)(暂时只有机翻)则可以带你快速入门仿人类微操的语句知识。

**本UserPatch 1.5 脚本编写参考的定位——**

这并不是一个系统的**教程**，而只是作为一本**词典**，帮助你快速查阅各种AI词汇和语句的作用。所以请不要被“120多页，55000多字”的惊人信息量吓到了，你真正需要的可能只是其中的几页，几千字，相当于阅读一篇短散文。

**那么我们要如何快速查阅这个参考书中的知识呢？——**

1. 首先请到“视图”里打开Word的“导航窗格”。这是一个便携的目录，可以帮助你快速定位到各个标题。
2. 先浏览一番“导航窗格”，看看整个参考的目录结构，大致清楚一下各种词汇的分类和位置。
3. 如果你需要找某个词汇，可以使用Ctrl+F弹出搜索菜单，搜索你想要的关键词，这比肉眼查找高效多了。



# UserPatch 1.3相对于1.0c的全局改进

·所有使用>, >=, <, <=, ==, !=来执行数据比较的事实现在都支持比较运算符。

·字符串表扩充为5000条。

·defconst命令可以定义字符串常数。

·规则数量上限扩充为10000。

·有效目标(goal)编号范围扩充为1~512，其中拓展目标为41~512。

·有效定时器(timer)编号范围扩充为1~50。

# UserPatch 1.5相对于UserPatch 1.3的改进

·更多新的AI指令(action&fact) 、更多新的策略值(sn)。

·新增几个直接单位控制(DUC)的动作(action)。

·更多物件数据常数(可通过load 语句载入[UserPatchConst.per](https://www.hawkaoe.net/bbs/thread-145196-1-1.html)来快捷定义这些常数)。

·AI现在能够精确地放置墙和其他建筑物(可全文搜索：选址控制系统)。

·AI现在有海拔意识，且可以进行格内精细操作。

·最大策略值编号由303提升至511（仅DE版本）。

# 动作 (Actions)

## up-add-cost-data

语法

up-add-cost-data (in)(goal)目标编号 类型操作符 (in)(op)值

功能

在当前成本数据基础上增加或减去某数量的成本。

“当前成本”为通过up-setup-cost-data绑定的连续四个目标。

(in)(op)值 为加到当前成本前，目标编号代表的成本将乘以的倍率，设为负数可以做减法。

示例

1. 向gl-cost-food代表的当前成本数据增加2份军事单位成本数据。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defconst gl-military-cost-food 111)

(defconst gl-military-cost-wood 112)

(defconst gl-military-cost-stone 113)

(defconst gl-military-cost-gold 114)

(defrule

(true)

=>

(up-setup-cost-data 1 gl-military-cost-food)

(up-add-object-cost c: knight-line c: 40)

(up-setup-cost-data 1 gl-cost-food)

(up-add-cost-data gl-military-cost-food c: 2)

(disable-self)

)

## up-add-object-by-id

语法

up-add-object-by-id (in)(const)搜索源 类型操作符 (in)(op)值

功能

把特定地图ID的物件添加到搜索结果中。

此命令可用作事实或行动。

· 用作事实时，如果该id单位不存在或已死亡，则事实不成立；

· 同样，用作动作时若id无效，也不会把该id加入到搜索结果。

有趣的是，这个动作还可以强行将其他玩家的单位加入到本地结果，然后通过up-target-\*指令强行操控。这里有一个应用：[[UP1.5] 让AI强行控制玩家1的单位！](https://www.hawkaoe.net/bbs/thread-146323-1-1.html)。

示例

1. 将gl-stored-id中储存的ID代表的物件添加到本地搜索结果中。

(defconst gl-stored-id 100)

(defrule

(true)

=>

(up-add-object-by-id search-local g: gl-stored-id)

)

## up-add-object-cost

语法

up-add-object-cost 类型操作符 (in)(op)物件编号 类型操作符 (in)(op)值

功能

在当前成本数据基础上增加或减去若干个给定物件的建造成本，并把结果赋值给“当前成本”。

“当前成本”为通过up-setup-cost-data绑定的连续四个目标。

(in)(op)值 为加到当前成本前，物件成本将乘以的倍率，设为负数可以做减法。

示例

1. 向gl-cost-food代表的当前成本数据增加4个长矛兵的成本。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defrule

(true)

=>

(up-setup-cost-data 0 gl-cost-food)

(up-add-object-cost c: spearman c: 4)

(disable-self)

)

2. 增加1个城镇中心的成本（特例）。

(defrule

(true)

=>

(up-add-object-cost c: town-center-foundation c: 1) ; town-center-foundation = 621

;(up-add-object-cost c: town-center c: 1) ; 错误，这样不包括石料消耗

(disable-self)

)

## up-add-point

语法

up-add-point (inout)(goal)点1-x (in)(goal)点2-x 类型操作符 (in)(op)值

功能

加或减两组表示点的目标对，并把结果赋值给点1。

把点2写为0可使用up-set-target-point存储的点。

命令会读取或写入两个连续编号的目标对中，有效目标编号范围为41至510。

(in)(op)值 为加到点1坐标前，点2坐标将乘以的倍率，设为负数可以做减法。

示例

1. 把焦点玩家和目标玩家所在位置相加。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defconst gl-other-x 200)

(defconst gl-other-y 201)

(defrule

(true)

=>

(up-get-point position-focus gl-point-x)

(up-get-point position-target gl-other-x)

(up-add-point gl-point-x gl-other-x c: 1)

)

## up-add-research-cost

语法

up-add-research-cost 类型操作符 (in)(op)科技编号 类型操作符 (in)(op)值

功能

在当前成本数据基础上增加或减去若干个给定科技的研发成本，并把结果赋值给“当前成本”。

“当前成本”为通过up-setup-cost-data绑定的连续四个拓展目标。

(in)(op)值 为加到当前成本前，科技成本将乘以的倍率，设为负数可以做减法。

示例

1. 从gl-cost-food代表的当前成本数据中减去织布机的研发成本。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defrule

(true)

=>

(up-setup-cost-data 0 gl-cost-food)

(up-add-research-cost c: ri-loom c: -1)

(disable-self)

)

## up-assign-builders

语法

up-assign-builders 类型操作符 (in)(op)建筑编号 类型操作符 (in)(op)值

功能

为给定种类的建筑指定给定数量的建筑工。

(in)(op)值 范围为-1~250，设为-1表示禁止指派额外的建筑工。

示例

1. 为所有住房（并非每个住房）指派至少2名建筑工。

(defrule

(true)

=>

(up-assign-builders c: house c: 2)

(disable-self)

)

2. 为城镇中心指派至少4名建筑工。（特例）

(defrule

(true)

=>

(up-assign-builders c: town-center-foundation c: 4) ; town-center-foundation = 621

;(up-assign-builders c: town-center c: 4) ; 错误，这条语句无效

(disable-self)

)

## up-bound-point

语法

up-bound-point (out)(goal)点1-x (in)(goal)点2-x

功能

复制一个代表点的目标对，并将其约束到地图边界内。比如将(-1, -1)转为(0, 0)。

这种约束会把地图外的点移到地图内最近的点。且悬崖和某些障碍物碰撞体积内也被视为地图外，这可以用于避开障碍物。

示例

1. 复制一个点并将其约束到地图边界内。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defconst gl-store-x 200)

(defconst gl-store-y 201)

(defrule

(true)

=>

(up-get-point position-object gl-point-x)

(up-bound-point gl-store-x gl-point-x)

)

## up-bound-precise-point

语法

up-bound-precise-point (inout)(goal)点-x (in)(const)精确模式 类型操作符 (in)(op)边界

功能

在地图中利用附加参数约束一个代表点的目标对。

请确保边界是一个有效的值，且不会导致超出地图边界。

(in)(const)精确模式 写为1，则在约束前地图尺寸将被乘以100，同时边界值也需做同步调整。

(in)(op)边界 取值范围为0~32767，即用于约束点的边界宽度。

示例

1. 使用一个10格的边界在地图中约束一个点。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-object gl-point-x)

(up-bound-precise-point gl-point-x 0 c: 10)

)

## up-build

语法

up-build (in)(const)放置模式 (in)(goal)储备资源状态 类型操作符 (in)(op)建筑编号

功能

使用变量，向建筑序列中增加一个建筑。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 不使用储备资源，建筑一个哨站。

(defrule

(up-can-build 0 c: outpost)

=>

(up-build place-normal 0 c: outpost)

)

2. 使用储备资源，建筑一个哨站。

(defconst gl-escrow-state 0)

(defrule

(true)

=>

(set-goal gl-escrow-state with-escrow)

(disable-self)

)

(defrule

(up-can-build gl-escrow-state c: outpost)

=>

(up-build place-normal gl-escrow-state c: outpost)

)

## up-build-line

语法

up-build-line (in)(goal)点1-x (in)(goal)点2-x 类型操作符 (in)(op)建筑编号

功能

在两个点之间放置一条线的建筑。

若想放置城镇中心和城门，请使用其地基编号，如town-center-foundation 和 gate-ascending，而不要使用town-center 或 gate。

把点2写为0可使用up-set-target-point存储的点。

示例

1. 在两点之间建一道木栅栏墙。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defconst gl-other-x 200)

(defconst gl-other-y 201)

(defrule

(true)

=>

(up-build-line gl-point-x gl-other-x c: palisade-wall)

)

## up-buy-commodity

语法

up- buy-commodity 类型操作符 (in)(op)资源编号 类型操作符 (in)(op)值

功能

在市场上买入一定数量的资源。实际买到的资源数量取决于可用的黄金。

(in)(op)值 的范围为1~32767。

示例

1. 试着在市场上买300点食物。

(defrule

(true)

=>

(up-buy-commodity c: food c: 300)

)

## up-cc-add-resource

语法

up-cc-add-resource 类型操作符 (in)(op)资源编号 类型操作符 (in)(op)值

功能

增加（或减少）一定量的资源到当前玩家库存中。

与cc-add-resource相比，这个指令可以增加绝大部分基础和隐藏资源，而且所增加的值也是可变的。

这可以说是一个作弊命令，但并不需要允许作弊也能使用。

(in)(op)值 为较大的负数时，可以导致玩家拥有负数的资源。

示例

1. 增加100点食物到库存。

(defrule

(true)

=>

(up-cc-add-resource c: food c: 100)

)

## up-cc-send-cheat

语法

up-cc-send-cheat (in)(text)作弊码

功能

发送一条作弊代码来执行一次作弊。

必须允许作弊才能生效。作弊码不区分大小写。

示例

1. 使用“to smithereens”作弊代码生产一个英雄破坏者。

(defrule

(true)

=>

(up-cc-send-cheat "to smithereens")

(disable-self)

)

## up-change-name

语法

up-change-name (in)(text)新名称

功能

在游戏进行期间改变AI玩家名称。

文本长度不能超过32个字符，半角占1字符，全角占2字符。超出的部分会压缩显示为“[W]”。

示例

1. 改变AI玩家名称为“William Wallace”。

(defrule

(true)

=>

(up-change-name "William Wallace")

(disable-self)

)

## up-chat-data-to-all

语法

up-chat-data-to-all (in)(text)格式化文本 类型操作符 (in)(op)值

功能

向所有玩家发送一条格式化后的聊天内容。这个指令通常可以用于检查某个目标变量的值。格式化文本中的%d会被替换为值的内容。

示例

1. 告诉所有玩家你的玩家编号。

(defrule

(true)

=>

(up-chat-data-to-all "I am player %d." c: my-player-number)

(disable-self)

)

## up-chat-data-to-player

语法

up-chat-data-to-player (in)(player)玩家 (in)(text)格式化文本 类型操作符 (in)(op)值

功能

向给定玩家发送一条格式化后的聊天内容。格式化文本中的%d会被替换为值的内容。

示例

1. 告诉所有同盟当前的目标玩家编号。

(defrule

(true)

=>

(up-chat-data-to-player every-ally "The target is player %d." s: sn-target-player-number)

(disable-self)

)

## up-chat-data-to-self

语法

up-chat-data-to-self (in)(text)格式化文本 类型操作符 (in)(op)值

功能

向自己发送一条格式化后的聊天内容。格式化文本中的%d会被替换为值的内容。

示例

1. 向自己发送聊天内容“Food: 5.”。

(defrule

(true)

=>

(up-chat-data-to-self "Food: %d." c: 5)

(disable-self)

)

## up-clean-search

语法

up-clean-search (in)(const)搜索源 (in)(const)物件数据 (in)(const)搜索顺序

功能

删除搜索结果中的重复单位，或以物件数据为依据对结果进行排序。

如果(in)(const)物件数据 写为-1，将尝试删除重复项，从而降低结果总数。

如果希望按物件数据排序，最好先删除重复项。

在删除重复项时，使用search-order-none来保存现有顺序的速度可能比使用asc/desc要慢。

受列表中对象的数量影响，此命令可能比较“昂贵”，因此请注意性能。

示例

1. 对远程搜索结果进行去重，并从最旧的对象排序到最新的。之后，根据生命值对列表进行升序排序。

(defrule

(true)

=>

(up-clean-search search-remote -1 search-order-asc)

(up-clean-search search-remote object-data-hitpoints search-order-asc)

)

## up-copy-point

语法

up-copy-point (out)(goal)点1-x (in)(goal)点2-x

功能

把一对表示点2的目标值复制给另一对表示点1的目标。

命令会读取或写入两个连续编号的目标对中，有效目标编号范围为41至510。

示例

1. 复制地图中心点位置。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defconst gl-other-x 200)

(defconst gl-other-y 201)

(defrule

(true)

=>

(up-get-point position-center gl-point-x)

(up-copy-point gl-other-x gl-point-x)

)

## up-create-group

语法

up-create-group (in)(goal)起始索引 (in)(goal)最大单位数量 类型操作符 (in)(op)搜索组编号

功能

从本地搜索结果里，从指定索引开始，尽可能“框选”指定数量的单位，以创建一个ctrl+[0-9]搜索组。

（注意，搜索组并不是人类玩家ctrl+[0-9]的那种编队，想要创建数字编队详见up-modify-group-flag语句）

如果结果列表里没有可用的单位用于编组，那么这个搜索组将被清空，则其效果将与up-reset-group相同。

创建同号搜索组会覆盖已有的同号搜索组，而不是追加到已有结果后面。

(in)(goal)起始索引 为创建搜索组的起始单位位于搜索结果里的索引，首个单位索引为0，若该参数为0则使用默认索引值0，注意，这应该是一个目标变量。

(in)(goal)最大单位数量 若为0则使用默认值40，注意，这应该是一个目标变量。

(in)(op)搜索组编号 只能取0~9，共10个搜索组。

本指令会把搜索结果里索引为 [起始索引 ~ (起始索引+最大单位数量-1) ] 的单位编为搜索组。

这个指令与up-set-group配合时，能让我们把一些保存好的单位直接加入到搜索结果，而不需要再进行一次比较占用性能的搜索。

示例

1. 给最多40个(虽然实际只有10个)村民创建一个0号搜索组。

(defrule

(true)

=>

(up-reset-search 1 1 1 1)

(up-reset-filters)

(up-find-local c: villager-class c: 10)

(up-create-group 0 0 c: 0)

)

## up-cross-tiles

语法

up-cross-tiles (inout)(goal)点1-x (in)(goal)点2-x 类型操作符 (in)(op)网格数

功能

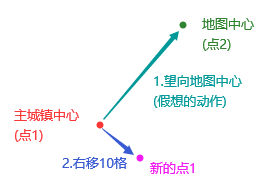
获取站在点1望向点2时，点1的右侧给定格远的点。

(in)(op)网格数 为负值时会得到点1左侧的点。

把点2写为0可使用up-set-target-point存储的点。

示例

1. 获取在面向地图中心时，主城镇中心右侧10格的点。



(defconst gl-self-x 100)

(defconst gl-self-y 101)

(defconst gl-center-x 200)

(defconst gl-center-y 201)

(defrule

(true)

=>

(up-get-point position-self gl-self-x)

(up-get-point position-center gl-center-x)

(up-cross-tiles gl-self-x gl-center-x c: 10)

)

## up-delete-distant-farms

语法

up-delete-distant-farms 类型操作符 (in)(op)距离

功能

删除所有给定资源放置距离以外的农田。(in)(op)距离 的范围为0~255。

示例

1. 删除所有“最小资源放置距离”大于8的农田。

(defrule

(true)

=>

(up-delete-distant-farms c: 8)

(disable-self)

)

## up-delete-idle-units

语法

up-delete-idle-units (in)(const)闲置单位类型

功能

删除所有给定类型的闲置单位。

示例

1. 删除所有闲置村民。

(defrule

(true)

=>

(up-delete-idle-units idle-type-villager)

(disable-self)

)

## up-delete-objects

语法

up-delete-objects 类型操作符 (in)(op)单位编号 类型操作符 (in)(op)生命值

功能

删除所有剩余生命值低于给定值的单位。

示例

1. 删除所有墙系建筑。

(defrule

(true)

=>

(up-delete-objects c: wall-class c: 32767) ; wall-class = 927

(disable-self)

)

## up-disband-group-type

语法

up-disband-group-type (in)(const)编组类型

功能

解散所有给定类型的内部编组。

示例

1. 用一种更直接的方式停止所有陆地探索编组。

(defrule

(true)

=>

(set-strategic-number sn-number-explore-groups 0)

(set-strategic-number sn-total-number-explorers 0)

(up-disband-group-type group-type-land-explore)

(up-reset-unit c: all-units-class)

(disable-self)

)

## up-drop-resources

语法

up-drop-resources (in)(const)资源 类型操作符 (in)(op)值

功能

要求至少携带给定数量资源的村民上交资源。(in)(const)资源可以是四种基本资源，也可以是某种资源物件的种属编号。

示例

1. 命令所有携带至少15食物的食物采集者上交资源。

(defrule

(true)

=>

(up-drop-resources food c: 15)

(disable-self)

)

2. 命令所有携带至少5食物的农民上交资源。

(defrule

(true)

=>

(up-drop-resources farm-class c: 5) ; farm-class = 949

(disable-self)

)

## up-filter-distance

语法

up-filter-distance 类型操作符 (in)(op)最小距离 类型操作符 (in)(op)最大距离

功能

为过滤式搜索系统设置距离参数，只搜索距目标点(target-point) [最小距离] ~ [最大距离] 之间的物件。

(in)(op)最小距离 若设为-1，则取默认值 0。

(in)(op)最大距离 若设为-1，则取默认值 32767。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 获取地图中心的坐标，设为目标点。然后配置搜索系统，使其只会选择目标点周围20格以内的物件。最后搜索自己的单位，不限定物件。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

(up-set-target-point gl-center-x)

(up-filter-distance c: -1 c: 20)

(up-find-local c: -1 c: 240)

)

## up-filter-exclude

语法

up-filter-exclude (in)(const)命令编号 (in)(const)行动编号 (in)(const)执行编号 (in)(const)种属编号

功能

为过滤式搜索系统设置排除参数。尚未探明上面的几个参数之间是或关系还是且关系（即：是物件要所有排除参数都符合才会被搜索系统忽略，还是只需符合任一排除参数就会被忽略）。

若某个参数设为-1，则过滤搜索会忽略此条件。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 配置搜索系统，使其会排除(即不选择)具有贸易命令编号的单位，然后对玩家1进行远程搜索，不限定物件。

(defrule

(true)

=>

(up-filter-exclude 5 -1 -1 -1)

(set-strategic-number sn-focus-player-number 1)

(up-find-remote c: -1 c: 40)

)

## up-filter-garrison

语法

up-filter-garrison 类型操作符 (in)(op)最小驻扎数 类型操作符 (in)(op)最大驻扎数

功能

为过滤式搜索系统设置驻扎数参数，只搜索驻扎有 [最小驻扎数] ~ [最大驻扎数] 个单位的物件。

(in)(op)最小驻扎数 若设为-1，则取默认值 0。

(in)(op)最大驻扎数 若设为-1，则取默认值 32767。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 配置搜索系统，使其只会选择驻扎有至少2个单位的物件，然后搜索自己的至多10辆冲车。

(defrule

(true)

=>

(up-filter-garrison c: 2 c: -1)

(up-find-local c: battering-ram-line c: 10)

)

## up-filter-include

语法

up-filter-include (in)(const)命令编号 (in)(const)行动编号 (in)(const)执行编号 (in)(const)位于主大陆

功能

为过滤式搜索系统设置包括参数。这些参数之间是且关系（即：必须全部都符合才会被搜索）。

若某个参数设为-1，则过滤搜索会忽略此条件。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 配置搜索系统，使其只会搜索具有军事命令编号(亦即军事部队)、且正在进行攻击的物件，然后对玩家1进行远程搜索，不限定物件。

(defrule

(true)

=>

(up-filter-include cmdid-military actionid-attack -1 -1)

(set-strategic-number sn-focus-player-number 1)

(up-find-remote c: -1 c: 40)

)

## up-filter-range

语法

up-filter-range (in)(const)最小驻扎数 (in)(const)最大驻扎数 (in)(const)最小距离 (in)(const)最大距离

功能

为过滤式搜索系统设置范围参数。

这条语句是up-filter-distance和up-filter-garrison的简单组合，不过相比那两条语句，该语句的参数只支持常数，并不支持使用变量或策略值。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 配置搜索系统，使其只会选择目标点周围10格以内的单位，然后对玩家1进行远程搜索，不限定物件。

(defrule

(true)

=>

(up-filter-range -1 -1 -1 10)

(set-strategic-number sn-focus-player-number 1)

(up-find-remote c: -1 c: 40)

)

## up-filter-status

语法

up-filter-status 类型操作符 (in)(op)物件状态 类型操作符 (in)(op)物件清单

功能

为过滤式搜索系统设置物件状态参数，从而与up-find-status-\*配合使用。

(in)(op)物件状态 的默认值是2 (开局及up-reset-filters后)，这会匹配大多数活动物件。未建成建筑的状态为0，而某些资源的状态为3，更多细节详见参数说明。

对于普通的远程搜索（即up-find-local和up-find-remote），如果你使用的是单位编号而不是用种属编号来搜索，则可以匹配到状态为0~3的物件。

多个不同的up-filter-\*语句的筛选条件可以互相叠加 (是且关系)，而执行相同的up-filter-\*语句将覆盖之前该语句设定的参数。

一般是先设好足够的up-filter-\*，再执行up-find-\*来依据筛选条件进行搜索。

示例

1. 配置搜索系统，使其只会选择未完成的城堡。

(defrule

(true)

=>

(up-filter-status c: status-pending c: list-active)

(up-find-status-local c: castle c: 5)

)

## up-find-flare

语法

up-find-flare (out)(goal)点

功能

获取同盟发送的闪光信号坐标，写入表示点的目标对中。

命令会写入两个连续编号的目标对中，有效目标编号范围为41至510。

注意，如果获取失败，会返回(-1,-1)。

命令等效于up-find-player-flare 以any-ally。

DE版Build 42848后能通过这个语句正常搜索信号。

示例

1. 获取最新的同盟闪光信号位置。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(unit-type-count flare > 0) ; flare = 274

=>

(up-find-flare gl-point-x)

)

## up-find-local

语法

up-find-local 类型操作符 (in)(op)单位编号 类型操作符 (in)(op)数量

功能

寻找本地玩家的物件，供直接单位控制(DUC)使用。

搜索的起始点是玩家单位列表的首位，终点是单位列表的末尾，如果中途搜索到了给定数量的单位，则暂停搜索。

如果 (in)(op)单位编号 改变，那么搜索索引偏移会重置（即回到玩家单位列表首位重新开始搜索）。反之，搜索会从上次搜索结束处继续（一般是一次搜索后未搜索完所有满足条件的单位）。

(in)(op)数量 取值应为0~240，换句话说，本地搜索结果列表最多只能容纳240个单位。

对于普通的远程搜索（即up-find-local和up-find-remote），如果你使用的是单位编号而不是用种属编号来搜索，则可以匹配到状态为0~3的物件。

本命令可以当做事实或动作使用。

示例

1. 寻找最多20个自己拥有的步兵单位，然后从头寻找最多3个弓兵单位，再从第4个弓兵继续寻找最多5个弓兵单位。

(defrule

(true)

=>

(up-find-local c: infantry-class c: 20) ; infantry-class = 906

(up-find-local c: archery-class c: 3) ; archery-class = 900

(up-find-local c: archery-class c: 5) ; archery-class = 900

)

## up-find-next-player

语法

up-find-next-player (in)(const)玩家外交状态 (in)(const)寻找玩家方法 (inout)(goal)玩家编号

功能

寻找下一个符合给定信息的活动玩家。(inout)(goal)玩家编号 除了储存匹配结果，也决定了搜索的起始玩家。

示例

1. 从gl-player储存的玩家起始，寻找距离自己第二近的敌对玩家，存储在gl-player中。

(defconst gl-player 101)

(defrule

(true)

=>

(up-find-player enemy find-closest gl-player)

(up-find-next-player enemy find-closest gl-player)

)

## up-find-player

语法

up-find-player (in)(const)玩家外交状态 (in)(const)寻找玩家方法 (out)(goal)玩家编号

功能

寻找第一个符合给定条件的活动玩家。

示例

1. 寻找距离自己最近的敌对玩家，存储在gl-player中。

(defconst gl-player 101)

(defrule

(true)

=>

(up-find-player enemy find-closest gl-player)

)

## up-find-player-flare

语法

up-find-player-flare (in)(player)任一玩家 (out)(goal)点

功能

获取任意可见的单个玩家的闪光信号，读取其位置到表示点的目标对中。

命令会写入两个连续编号的目标对中，有效目标编号范围为41至510。

如果获取失败，会返回(-1,-1)。

示例

1. 获取任意敌对玩家闪光信号位置。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(unit-type-count flare > 0) ; flare = 274

=>

(up-find-player-flare any-enemy gl-point-x)

)

## up-find-remote

语法

up-find-remote 类型操作符 (in)(op)单位编号 类型操作符 (in)(op)数量

功能

寻找焦点玩家的物件，供直接单位控制(DUC)使用。

搜索的起始点是玩家单位列表的首位，终点是单位列表的末尾，如果中途搜索到了给定数量的单位，则暂停搜索。

使用本命令前，要设置sn-focus-player-number。

如果 (in)(op)单位编号 改变，那么搜索索引偏移会重置（即回到玩家单位列表首位重新开始搜索）。反之，搜索会从上次搜索结束处继续（一般是一次搜索后未搜索完所有满足条件的单位）。

(in)(op)数量 取值应为0~40，换句话说，远程搜索结果列表最多只能容纳40个单位。

对于普通的远程搜索（即up-find-local和up-find-remote），如果你使用的是单位编号而不是用种属编号来搜索，则可以匹配到状态为0~3的物件。

本命令可以当做事实或动作使用。

示例

1. 寻找敌对玩家2拥有的至多4个城堡。

(defrule

(true)

=>

(set-strategic-number sn-focus-player-number 2)

(up-find-remote c: castle c: 4)

)

## up-find-resource

语法

up-find-resource 类型操作符 (in)(op)资源 类型操作符 (in)(op)数量

功能

寻找可采集的资源物件，供直接单位控制(DUC)使用。

(in)(op)资源 只支持四种基础资源和食物类的物件编号。

本命令会把数据储存到远程列表中，且会考虑由up-filter-status设定的状态值。在搜索之前，请确保设置了正确的状态，详情说明请看物件状态。

当在此命令与其他远程搜索命令(如up-find-remote)之间切换时，远程索引偏移将自动重置；如果资源改变，那么搜索索引偏移也会重置。反之，搜索会从上次搜索结束处继续。

本命令可以当做事实或动作使用。

示例

1. 寻找2块可见的金矿，储存在远程搜索结果中。

(defrule

(true)

=>

(up-full-reset-search)

(up-filter-status c: status-resource c: list-active)

(up-find-resource c: gold c: 2)

)

## up-find-status-local

语法

up-find-status-local 类型操作符 (in)(op)单位编号 类型操作符 (in)(op)数量

功能

寻找以状态过滤后的本地玩家的物件。

这个与up-find-local基本相同，只是会多考虑由up-filter-status设定的状态值。

对于普通的远程搜索（即up-find-local和up-find-remote），如果你使用的是单位编号而不是用种属编号来搜索，则可以匹配到状态为0~3的物件。

如果单位编号改变，那么搜索索引偏移会重置。反之，搜索会从上次搜索结束处继续。

数量取值应为0~240。

本命令可以当做事实或动作使用。

示例

1. 寻找本地玩家拥有的至多20个步兵单位。

(defrule

(true)

=>

(up-filter-status c: status-ready c: list-active)

(up-find-status-local c: infantry-class c: 20) ; infantry-class = 906

)

## up-find-status-remote

语法

up-find-status-remote 类型操作符 (in)(op)单位编号 类型操作符 (in)(op)数量

功能

寻找以状态过滤后的焦点玩家的物件，供直接单位控制(DUC)使用。

使用本命令前，要设置sn-focus-player-number。

这个与up-find-remote基本相同，只是会多考虑由up-filter-status设定的状态值。

对于普通的远程搜索（即up-find-local和up-find-remote），如果你使用的是单位编号而不是用种属编号来搜索，则可以匹配到状态为0~3的物件。

如果焦点玩家或单位编号改变，那么搜索索引偏移会重置。反之，搜索会从上次搜索结束处继续。

本命令可以当做事实或动作使用。

示例

1. 寻找敌对玩家P2拥有的至多4座城堡。

(defrule

(true)

=>

(set-strategic-number sn-focus-player-number 2)

(up-find-status-remote c: castle c: 4)

)

## up-full-reset-search

语法

up-full-reset-search

功能

给搜索系统重置所有搜索和筛选状态。此命令是(up-reset-search 1 1 1 1)和(up-reset-filters)的简单组合，用于简化规则。

示例

1. 清除所有搜索和筛选状态。

(defrule

(true)

=>

(up-full-reset-search)

)

## up-garrison

语法

up-garrison (in)(const)物件编号 类型操作符 (in)(op)单位编号

功能

命令所有给定类型的单位驻扎到给定物件中。

(in)(const)物件编号 不可以是一个种属或者兵系。它必须是能够驻扎其他单位的有效根物件(如应为battering-ram而不能是battering-ram-line)。

DE版Build 42848后支持兵系（尚未探明是支持上面哪个参数）。

示例

1. 命令所有在室外的步兵单位驻扎到冲车中。

(defrule

(true)

=>

(up-garrison battering-ram c: infantry-class) ; infantry-class = 906

(disable-self)

)

2. 命令所有在室外的弓箭手驻扎到塔中。

(defrule

(true)

=>

(up-garrison watch-tower c: archer-line)

(disable-self)

)

## up-gather-inside

语法

up-gather-inside 类型操作符 (in)(op)建筑编号 类型操作符 (in)(op)状态

功能

命令所有给定类型的建筑生产集结点为建筑本身。

(in)(op)状态 如果设为1，所生产和已驻扎的单位都会维持在建筑里；  
如果设为-1，建筑不会释放已驻扎单位，新生产的单位则正常释放；  
如果设为0，则任何单位都会像平常一样释放出去。

示例

1. 命令船坞训练的船只驻扎在船坞内，直到总共拥有了10艘船再释放。

(defrule

(true)

=>

(up-gather-inside c: dock c: 1)

(disable-self)

)

(defrule

(unit-type-count warship-class >= 10) ; warship-class = 922

=>

(up-gather-inside c: dock c: 0)

)

2. 阻止城镇中心释放驻扎单位，但同时允许继续训练村民。

(defrule

(true)

=>

(up-gather-inside c: town-center c: -1)

(disable-self)

)

## up-get-attacker-class

语法

up-get-attacker-class (out)(goal)攻击来源种属

功能

获取最后触发town-under-attack的敌对单位的种属。

示例

1. 获取正在进攻的敌对单位的种属，存储在gl-class中。

(defconst gl-class 101)

(defrule

(true)

=>

(up-get-attacker-class gl-class)

)

## up-get-cost-delta

语法

up-get-cost-delta (out)(goal)资源差值

功能

计算现有四种资源（依次为食物、木材、石料、黄金）与当前存储的成本数据的差值，储存到以 (out)(goal)资源差值 为首、连续编号的4个目标中。首个目标必须是41~508号的拓展目标。

示例

1. 计算上述差值，存储在以gl-delta-food为首的4个目标中。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defconst gl-delta-food 121)

(defconst gl-delta-wood 122)

(defconst gl-delta-stone 123)

(defconst gl-delta-gold 124)

(defrule

(true)

=>

(up-setup-cost-data 1 gl-cost-food)

(up-add-object-cost c: archer-line c: 10)

(up-get-cost-delta gl-delta-food)

(disable-self)

)

## up-get-event

语法

up-get-event 类型操作符 (in)(op)事件编号 (out)(goal)值

功能

获取触发AI事件的值。

AI事件应用于触发效果“发送AI信号(事件/触发器)”和AI事实“event-detected”。

示例

1. 获取0号事件的值，储存在gl-value中。

(defconst gl-value 100)

(defrule

(true)

=>

(up-get-event c: 0 gl-value)

)

## up-get-fact

语法

up-get-fact (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取my-player-number代表的玩家的一个事实，存储在一个目标中。

有些事实可以搭配参数，有些事实没有参数而需要写为0。

本命令可作为事实或动作使用。

示例

1. 读取military-population值，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-fact military-population 0 gl-data)

)

## up-get-fact-max

语法

up-get-fact-max (in)(player)玩家 (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取多个玩家的给定事实中的最大值，存储在一个目标中。

本命令可作为事实或动作使用。

（用作事实时）符合条件的玩家编号会被设置为this-any-\*百搭常数，供规则的动作部分调用以保证玩家编号一致。

(in)(player)玩家 只能是any-\*百搭常数。

有些事实可以搭配 (in)(const)参数，有些事实没有参数而需要写为0。

示例

1. 获取同盟中分数最高者，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-fact-max any-ally current-score 0 gl-data)

)

## up-get-fact-min

语法

up-get-fact-min (in)(player)玩家 (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取多个玩家的给定事实中的最小值，存储在一个目标中。

本命令可作为事实或动作使用。

（用作事实时）符合条件的玩家编号会被设置为this-any-\*百搭常数，供规则的动作部分调用以保证玩家编号一致。

(in)(player)玩家 只能是any-\*百搭常数。

有些事实可以搭配 (in)(const)参数，有些事实没有参数而需要写为0。

示例

1. 获取同盟中分数最低者，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-fact-min any-ally current-score 0 gl-data)

)

## up-get-fact-sum

语法

up-get-fact-sum (in)(player)玩家 (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

计算所有给定玩家的某项事实值的和，存储在一个目标中。

(in)(player)玩家 只能是any-\*百搭常数。

有些事实可以搭配 (in)(const)参数，有些事实没有参数而需要写为0。

本命令可作为事实或动作使用。

示例

1. 计算所有敌对玩家军事人口值的和，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-fact-sum any-enemy military-population 0 gl-data)

)

## up-get-focus-fact

语法

up-get-focus-fact (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取焦点玩家的给定事实的值，存储在一个目标中。

使用本命令前，要设置sn-focus-player-number。

有些事实可以搭配 (in)(const)参数，有些事实没有参数而需要写为0。

本命令可作为事实或动作使用。

示例

1. 获取焦点玩家人口值，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-focus-fact population 0 gl-data)

)

## up-get-group-size

语法

up-get-group-size 类型操作符 (in)(op)搜索组编号 (out)(goal)规模

功能

读取一个搜索组里的当前单位数量，存储在一个目标中。

示例

1. 获取第一个搜索组的规模，存储到gl-size中。

(defconst gl-size 100)

(defrule

(true)

=>

(up-get-group-size c: 0 gl-size)

)

## up-get-guard-state

语法

up-get-guard-state (out)(goal)状态

功能

获取保卫状态储存到以 (out)(goal)状态 为首的连续编号的4个拓展目标中（目标编号范围41~512）。

命令会以以下顺序给这些目标填充数据：[(保卫对象)单位编号]、[资源编号]、[资源增量]、[保卫标志位]。

这4种数据与RMS脚本里guard\_state的相同。

请使用up-compare-flag来检查 [保卫标志位] 的值。

随机地图脚本中：

如果附加了guard-flag-resource，且 [单位编号] 对应的物件仍留存，那么 [资源增量] / 100的结果将缓慢添加到 [资源编号] 里。

如果附加了guard-flag-resource和guard-flag-inverse，那么仅当已没有[单位编号]对应的物件留存时，资源才会被增加。

如果附加了guard-flag-victory，那么当已经没有[单位编号]对应的物件留存后，AI将被打败。

示例

1. 检查当失去保卫对象单位后是否AI会被打败。

(defconst gl-guard-type 100)

(defconst gl-guard-resource 101)

(defconst gl-guard-delta 102)

(defconst gl-guard-flags 103)

(defrule

(true)

=>

(up-get-guard-state gl-guard-type)

)

(defrule

(up-compare-flag gl-guard-flags == guard-flag-victory)

=>

(do-nothing)

)

2. 检查在保护保卫对象单位(TypeId)期间AI是否能获得资源。

(defconst gl-guard-type 100)

(defconst gl-guard-resource 101)

(defconst gl-guard-delta 102)

(defconst gl-guard-flags 103)

(defrule

(true)

=>

(up-get-guard-state gl-guard-type)

)

(defrule

(up-compare-goal gl-guard-delta > 0)

(up-compare-flag gl-guard-flags == guard-flag-resource)

(up-compare-flag gl-guard-flags != guard-flag-inverse)

=>

(do-nothing)

)

## up-get-indirect-goal

语法

up-get-indirect-goal 类型操作符 (in)(op)待读取目标编号 (out)(goal)值

功能

以给定数值作为目标的编号，从而间接读取该编号所对应的目标变量的值。

这个语句的一个应用是 [[UP1.5]变量存取的最佳方式 指标、指针、阵列的使用](https://www.hawkaoe.net/bbs/thread-146465-1-1.html)。

示例

1. 通过使用100号目标间接获取1号目标的值，储存到gl-value中。

(defconst gl-goal-id 100)

(defconst gl-value 101)

(defrule

(true)

=>

(set-goal gl-goal-id 1)

(up-get-indirect-goal g: gl-goal-id gl-value)

)

## up-get-object-data

语法

up-get-object-data (in)(const)物件数据 (out)(goal)数据

功能

获取目标物件的特定信息。

在使用这条命令前，请先设定目标物件(up-set-target-by-id或up-set-target-object)。

如果数据获取失败，则返回-2。

本命令可作为事实或动作使用。

示例

1. 获取目标物件的种属编号，储存到gl-data中。

(defconst gl-data 100)

(defrule

(true)

=>

(up-get-object-data object-data-class gl-data)

)

## up-get-object-target-data

语法

up-get-object-target-data (in)(const)物件数据 (out)(goal)数据

功能

获取目标物件的目标单位的特定信息。

在使用这条命令前，请先设定目标物件(up-set-target-by-id或up-set-target-object)。

如果数据获取失败，则返回-2。

本命令可作为事实或动作使用。

示例

1. 获取目标物件的目标单位的种属信息。

(defconst gl-data 100)

(defrule

(true)

=>

(up-get-object-target-data object-data-class gl-data)

)

## up-get-object-type-data

语法

up-get-object-type-data 类型操作符 (in)(op)物件编号 (in)(const)物件数据 (out)(goal)数据

功能

获取某种特定物件的特定原始数据。

“原始数据”是指在dat文件里设定的数据，不会在游戏期间被科技树或修改属性改变。

(in)(op)物件编号 可以是单位系或特定的单位编号，但不能是种属。

这条指令可能会很“昂贵”，所以请考虑性能。

DE在Build 37650起开始支持未启用/不可用的单位。

示例

1. 获取掷矛系的训练场所。

(defconst gl-data 100)

(defrule

(true)

=>

(up-get-object-type-data c: skirmisher-line object-data-train-site gl-data)

)

## up-get-path-distance

语法

up-get-path-distance (in)(goal)点-x (in)(const)严格 (out)(goal)路程

功能

获取从目标物件(target-object) 到 指定点 之间的路程 (向下取整)。

注意，由于点坐标只能是整数，因此实际计算时是格子左角之间的路程。

与直线距离不同，路程是考虑障碍物的。

· 如果点是无法抵达的，则路程会返回65535；

· 如果无法获取，则路程为-2。

(in)(const)严格 设为1表示不允许为了抵达而对点做偏移，这时若格子里有障碍物，可能会判定为无法抵达；

设为0表示　允许为了能抵达而对点做少量偏移（比如从树林内移到最近的树林边缘）。

把点写为0，可使用up-set-target-point存储的点位置。

示例

1. 获取从目标物件到地图中心的路程。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defconst gl-distance 200)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

(up-get-path-distance gl-center-x 0 gl-distance)

)

## up-get-player-color

语法

up-get-player-color (in)(player)任一玩家 (out)(goal)颜色编号

功能

读取单个玩家的颜色编号，同时将颜色名称存储在缓冲区中。

颜色编号范围为1~8，依次代表蓝、红、绿、黄、青、紫、灰、橙。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 获取目标玩家(target-player)的颜色编号，并将该颜色名称发送出来。

(defconst gl-value 100)

(defrule

(true)

=>

(up-get-player-color target-player gl-value)

(up-chat-data-to-all "Target: %s" c: 7031232)

)

## up-get-player-fact

语法

up-get-player-fact (in)(player)任一玩家 (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取单个玩家的给定事实的值，存储在一个目标中。

为更好的性能考虑，请尽可能使用up-get-fact系列里更直接的命令。

有些事实可以搭配 (in)(const)参数，如果不需要可以设为0。

本命令可作为事实或动作使用。

似乎在DE中，(in)(player)任一玩家 写为 focus-player 或者 target-player 时并无法正常获取数据。

示例

1. 获取玩家2的村民人口值，存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-player-fact 2 civilian-population 0 gl-data)

)

2. 获取玩家2的3号资源值（即黄金库存），存储到gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-player-fact 2 resource-amount 3 gl-data)

)

## up-get-point

语法

up-get-point (in)(const)位置类型 (out)(goal)点

功能

读取特定的点坐标信息，存储到表示点的目标对中。

命令会写入两个连续编号的目标对中，有效目标编号范围为41至510。

如果获取失败，会返回(-1,-1)。

示例

1. 获取地图中心的位置。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-point-x)

)

2. 获取目标玩家的位置。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-target gl-point-x)

)

## up-get-point-contains

语法

up-get-point-contains (in)(goal)点-x (out)(goal)地图ID 类型操作符 (in)(op)物件编号

功能

如果给定物件存在于给定点的格子上，则获取其地图ID。

把点写为0，可使用up-set-target-point存储的点位置。

(in)(op)物件编号 可以是单位编号、建筑编号、兵系、种属。请注意，如果使用all-units-class (-1)，这可能会获取到出乎意料的物件，比如刚好飞过格子的鸟、各种地形植物等。

多个物件在同一个格子上时，获取的优先级为：①先抵达格子上的物件，优先读取；②若同时抵达格子(如地编预先放置的)，则地图ID靠前的优先读取。

本命令可作为事实或动作使用。

示例

1. 如果一棵树存在于地图中心处，则获取它的地图ID。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defconst gl-tree-id 102)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-get-point-contains gl-center-x gl-tree-id c: tree-class) ; tree-class = 915

=>

(do-nothing)

)

## up-get-point-distance

语法

up-get-point-distance (in)(goal)点1-x (in)(goal)点2-x (out)(goal)距离

功能

计算两点直线距离。

把点2写为0，可使用up-set-target-point存储的点位置。

示例

1. 计算主城镇中心到地图中心的距离。

(defconst gl-self-x 100)

(defconst gl-self-y 101)

(defconst gl-center-x 200)

(defconst gl-center-y 201)

(defconst gl-distance 300)

(defrule

(true)

=>

(up-get-point position-self gl-self-x)

(up-get-point position-center gl-center-x)

(up-get-point-distance gl-self-x gl-center-x gl-distance)

)

## up-get-point-elevation

语法

up-get-point-elevation (in)(goal)点-x (out)(goal)数据

功能

获取给定点处的海拔高度。

把点写为0，可使用up-set-target-point存储的点位置。

如果获取失败，则返回-2。

示例

1. 获取地图中心处的海拔高度。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defconst gl-center-z 200)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

(up-get-point-elevation gl-center-x gl-center-z)

)

## up-get-point-terrain

语法

up-get-point-terrain (in)(goal)点-x (out)(goal)地形

功能

获取给定点的地形编号。

把点写为0，可使用up-set-target-point存储的点位置。

示例

1. 获取地图中心的地形编号。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defconst gl-terrain 200)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

(up-get-point-terrain gl-center-x gl-terrain)

)

## up-get-point-zone

语法

up-get-point-zone (in)(goal)点-x (out)(goal)数据

功能

获取给定点的地图区域编号。

每片陆地/水域都分配有不同的区域编号，这可以用于判断物件是否处于不同的陆地/水域。更多信息详见位于主大陆。

如果你没有村民（或城镇中心？），地图区域编号可能会有所不同。

把点写为0，可使用up-set-target-point存储的点位置。

示例

1. 获取地图中心的地图区域编号。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defconst gl-center-zone 200)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

(up-get-point-zone gl-center-x gl-center-zone)

)

## up-get-precise-time

语法

up-get-precise-time (in)(goal)起始时刻 (out)(goal)时间

功能

获取系统时间戳或时间戳之差（单位：毫秒。与现实时间同步）。

(in)(goal)起始时刻 决定了获取方式，写为0时会获取游戏系统时间戳；写为一个有效的目标变量时，将用当前时间戳减去目标变量储存的时间戳，得到从 [起始时刻] 到当前时间经过的时长。

帝国的时间戳与[Unix时间戳](https://baike.baidu.com/item/unix时间戳)数值并不相同，但本质上都是将现实时间转化为一串长数字。

这个语句的一个应用是 [[UP1.5] 利用AI检测玩家/游戏暂停了多长时间](https://www.hawkaoe.net/bbs/thread-146522-1-1.html)（例图如下）。



示例

1. 获取当前的游戏系统时间戳。

(defconst gl-start-time 100)

(defrule

(true)

=>

(up-get-precise-time 0 gl-start-time)

(disable-self)

)

2. 获取从开始的时间戳起到现在经过的时间。

(defconst gl-start-time 100)

(defconst gl-elapsed-time 101)

(defrule

(true)

=>

(up-get-precise-time gl-start-time gl-elapsed-time)

(disable-self)

)

## up-get-projectile-player

语法

up-get-projectile-player (in)(const)抛射物类型 (out)(goal)玩家编号

功能

获取最后一名使用给定类型的抛射物攻击的敌对玩家编号。

示例

1. 获取最后一名使用城堡箭攻击的敌对玩家编号，存储在gl-player中。

(defconst gl-player 101)

(defrule

(true)

=>

(up-get-projectile-player projectile-castle gl-player)

)

## up-get-rule-id

语法

up-get-rule-id (out)(goal)规则编号

功能

获取当前规则在所有规则集合中的从零开始的编号。

这个“规则编号”可以配合up-jump-direct来精准地跳转到指定规则。

这个语句的一个应用是 [[UP1.5] 函数的呼叫与使用](https://www.hawkaoe.net/bbs/thread-146485-1-1.html)。

示例

1. 获取当前规则的编号。

(defconst gl-current-rule-id 100)

(defrule

(true)

=>

(up-get-rule-id gl-current-rule-id)

)

## up-get-search-state

语法

up-get-search-state (out)(goal)状态

功能

获取一次搜索状态，存储到以“(out)(goal)状态”为首的连续4个拓展目标中。

语句会按以下顺序存储数据：①当前本地搜索结果总数，②最近一次本地搜索新增的结果数量，③当前远程搜索结果总数，④最近一次远程搜索新增的结果数量。

这个语句通常用于判断是否找到了结果、找到了多少个结果。

示例

1. 获取搜索状态，并报告出来。

(defconst gl-local-total 100)

(defconst gl-local-last 101)

(defconst gl-remote-total 102)

(defconst gl-remote-last 103)

(defrule

(true)

=>

(up-get-search-state gl-local-total)

(upchat-data-to-all "本地结果总数 = %d" g: gl-local-total)

(upchat-data-to-all "远程结果总数 = %d" g: gl-local-last)

(upchat-data-to-all "本地结果新增数 = %d" g: gl-remote-total)

(upchat-data-to-all "远程结果新增数 = %d" g: gl-remote-last)

)

## up-get-shared-goal

语法

up-get-shared-goal 类型操作符 (in)(op)共享目标编号 (out)(goal)值

功能

获取一个共享目标变量的值。

示例

1. 获取0号共享目标变量的值。

(defconst gl-value 100)

(defrule

(true)

=>

(up-get-shared-goal c: 0 gl-value)

)

## up-get-signal

语法

up-get-signal 类型操作符 (in)(op)信号编号 (out)(goal)值

功能

获取指定AI信号的值。

示例

1. 获取0号信号的值，储存在gl-value中。

(defconst gl-value 100)

(defrule

(true)

=>

(up-get-signal c: 0 gl-value)

)

## up-get-target-fact

语法

up-get-target-fact (in)(const)事实编号 (in)(const)参数 (out)(goal)数据

功能

读取目标玩家给定事实的值。

使用本命令前，请先设置sn-target-player-number。

有些事实可以搭配 (in)(const)参数，有些事实没有参数而需要写为0。

本命令可作为事实或动作使用。

示例

1. 读取目标玩家军事人口值，存储在gl-data中。

(defconst gl-data 101)

(defrule

(true)

=>

(up-get-target-fact military-population 0 gl-data)

)

## up-get-threat-data

语法

up-get-threat-data (out)(goal)过去时间 (out)(goal)玩家编号 (out)(goal)威胁种属 (out)(goal)受击种属

功能

获取时间上最近的一次威胁 离上次威胁过去的时间(ElapsedTime，单位: 游戏ms)、来源玩家、来源物件的种属(SourceClass)、遭受攻击的物件的种属(TargetClass)。

对自己任何单位或建筑造成伤害的行为都会被视为威胁。尤其是投石车之类，其强制攻击造成的溅射伤害也会被视为威胁。

盖亚的玩家编号为0。

示例

1. 获取上一次威胁的信息。

(defconst gl-threat-time 101)

(defconst gl-threat-player 102)

(defconst gl-threat-source 103)

(defconst gl-threat-target 104)

(defrule

(true)

=>

(up-get-threat-data gl-threat-time gl-threat-player gl-threat-source gl-threat-target)

)

## up-get-timer

语法

up-get-timer 类型操作符 (in)(op)定时器编号 (out)(goal)值

功能

计算一个定时器将/已在何时(游戏时间)激活，单位为ms。

而且，即使定时器已经处于激活状态(即倒计时已经到了0)，这个计算值不会改变，仍然是原值。

如果定时器被关闭，则该值为-1。

比如，在游戏时间00:10处(enable-timer 1 5)，则激活时间为00:15，那么计算得到的值为15000 ms。在游戏时间00:45处，这个计算值仍然会是15000 ms。若通过(disable-timer 1)关闭了定时器，则计算值为-1。

示例

1. 读取20号定时器的激活时间。

(defconst gl-value 100)

(defrule

(true)

=>

(up-get-timer c: 20 gl-value)

)

## up-get-upgrade-id

语法

up-get-upgrade-id (in)(player)任一玩家 (in)(const)计数(?) (in)(goal)单位编号 (out)(goal)升级单位编号

功能

获取单个玩家某种单位的升级单位编号。

(out)(goal)升级单位编号，是整个单位系里的当前单位。比如研究弩手后弓兵系的 [升级单位编号] 为24-弩手。

(in)(const)计数(? Count)，设为1可获取当前单位编号用于计数(?)。

(in)(goal)单位编号，可以是一个单位编号或单位系，为建筑编号时只会返回该编号本身。

示例

1. 获取当前民兵的升级单位编号。

(defconst gl-type 100)

(defconst gl-upgrade 101)

(defrule

(true)

=>

(set-goal gl-type militiaman-line)

(up-get-upgrade-id my-player-number 0 gl-type gl-upgrade)

(up-store-type-name g: gl-upgrade)

(up-chat-data-to-all "Upgrade: %s" c: 7031232)

)

## up-get-victory-data

语法

up-get-victory-data (out)(goal)玩家编号 (out)(goal)类型 (out)(goal)时间

功能

获取标准游戏的胜利状态信息，存储到给定目标中。

(out)(goal)玩家编号，会储存即将胜利的玩家编号，若无效则返回0。

(out)(goal)类型，会获取以下类型之一：圣物、奇观、纪念碑，若无效则返回0。

(out)(goal)时间，会储存距离胜利的剩余游戏年的10倍，若无效则返回-1。

示例

1. 获取当前胜利状态信息，存储到给定目标中。

(defconst gl-victory-player 101)

(defconst gl-victory-type 102)

(defconst gl-victory-time 103)

(defrule

(true)

=>

(up-get-victory-data gl-victory-player gl-victory-type gl-victory-time)

)

## up-get-victory-limit

语法

up-get-victory-limit (out)(goal)限制

功能

获取时间或得分的胜利限制。

如果是时限胜利，则获取剩余游戏年的10倍；

如果是得分胜利，则获取胜利所需目标分数；如果无效则返回-1。

示例

1. 获取当前胜利状态信息，存储到给定目标中。

(defconst gl-victory-limit 101)

(defrule

(true)

=>

(up-get-victory-limit gl-victory-limit)

)

## up-guard-unit

语法

up-guard-unit (in)(const)物件编号 类型操作符 (in)(op)单位编号

功能

命令单个给定的单位去保护一种给定物件中随机的一个。

(in)(const)物件编号，为被保护的对象。

示例

1. 命令一个长矛兵随机保护一个僧侣。

(defrule

(true)

=>

(up-guard-unit monk c: spearman-line)

)

## up-jump-direct

语法

up-jump-direct 类型操作符 (in)(op)规则编号

功能

在当前规则集合中直接跳转到某条规则。

当#load块有可能使跳转目标不可靠时，绝对不要使用本命令。

你可以使用up-get-rule-id来获取一个有效的规则编号用于跳转。

请确保跳转到的规则确实存在。

使用本动作，你可以借助智能的跳过从而减少每次遍历的规则量，也可以借助循环增加规则量。

成百上千次跳转可能会造成机器负担，所以请考虑游戏性能。

这个语句的一个应用是 [[UP1.5] 函数的呼叫与使用](https://www.hawkaoe.net/bbs/thread-146485-1-1.html)。

示例

1. 如果条件为真，则跳转到第1条规则。

(defconst g-rule-id 100)

(defrule

(true)

=>

(up-get-rule-id g-rule-id)

)

(defrule

(true)

=>

(up-jump-direct g: g-rule-id)

)

## up-jump-dynamic

语法

up-jump-dynamic 类型操作符 (in)(op)规则偏移量

功能

在当前规则集合中动态地跳转。

相比于只能跳转固定值的up-jump-rule，这个指令更加灵活可变。

当#load块有可能使跳转目标不可靠时，绝对不要使用本命令。

请确保跳转到的规则确实存在。

使用本动作，你可以借助跳过减少每次遍历的规则量，也可以借助循环增加规则量。

成百上千次跳转可能会造成机器负担，所以请考虑游戏性能。

(in)(op)规则偏移量，为负数时，-1表示跳转到规则自身开头，-2表示跳转到上条规则开头，1表示跳转到下一条规则开头，以此类推。

示例

1. 如果条件为真，那么跳过1条规则。

(defconst g-jump 100)

(defrule

(true)

=>

(up-modify-goal g-jump c:= 1)

)

(defrule

(true)

=>

(up-jump-dynamic g: g-jump)

)

(defrule ; 这条规则被跳过了

(true)

=>

(do-nothing)

)

## up-jump-rule

语法

up-jump-rule (in)(const)规则偏移量

功能

在当前规则集合中前后跳转。

请确保跳转到的规则确实存在。

使用本动作，你可以借助智能的跳过从而减少每次遍历的规则量，也可以借助循环增加规则量。

成百上千次跳转可能会造成机器负担，所以请考虑游戏性能。

(in)(op)规则偏移量，为负数时，-1表示跳转到规则自身开头，-2表示跳转到上条规则开头，1表示跳转到下一条规则开头，以此类推。

示例

1. 如果条件为真，那么跳过1条规则。

(defrule

(true)

=>

(up-jump-rule 1)

)

(defrule ; 这条规则被跳过了

(true)

=>

(do-nothing)

)

2. 循环执行一条规则3次。

(defconst gl-value 100)

(defrule

(true)

=>

(chat-local-to-self "开始")

(set-goal gl-value 0)

)

(defrule ; 这条规则会被重复执行3次

(up-compare-goal gl-value < 3)

=>

(up-modify-goal gl-value c:+ 1)

(up-jump-rule -1)

)

## up-lerp-percent

语法

up-lerp-percent (inout)(goal)点1-x (in)(goal)点2-x 类型操作符 (in)(op)百分数

功能

计算从点1到点2的某个百分位点。

(in)(op)百分数，为负数会导致反向，设为大于100的百分数会导致延长。

设置点2为0可使用up-set-target-point存储的点。

示例

1. 计算从主城镇中心到地图中心的25%分位点。

(defconst gl-self-x 100)

(defconst gl-self-y 101)

(defconst gl-center-x 200)

(defconst gl-center-y 201)

(defrule

(true)

=>

(up-get-point position-self gl-self-x)

(up-get-point position-center gl-center-x)

(up-lerp-percent gl-self-x gl-center-x c: 25)

)

## up-lerp-tiles

语法

up-lerp-tiles (inout)(goal)点1-x (in)(goal)点2-x 类型操作符 (in)(op)网格数

功能

获取以点1为基准，沿点2方向偏移给定网格数后的点。

(in)(op)网格数，为负数会导致反向。

设置点2为0可使用up-set-target-point存储的点。

示例

1. 获取主城镇中心向地图中心方向偏移10格后的点。

(defconst gl-self-x 100)

(defconst gl-self-y 101)

(defconst gl-center-x 200)

(defconst gl-center-y 201)

(defrule

(true)

=>

(up-get-point position-self gl-self-x)

(up-get-point position-center gl-center-x)

(up-lerp-tiles gl-self-x gl-center-x c: 10)

)

## up-log-data

语法

up-log-data (in)(const)文本模式 (in)(text)格式化文本 类型操作符 (in)(op)值

功能

写入一行格式化文本到aoelog.txt文件里。

(in)(text)格式化文本 中的%d会被替换为 (in)(op)值 的内容。

(in)(const)文本模式 为0，写入正常的格式化文本；为1，可以写入纯文本。

必须在关闭游戏后才能打开aoelog.txt。这个txt文件位于游戏目录里。

写入数据时请考虑游戏性能。

示例

1. 记录自己的玩家编号。

(defrule

(true)

=>

(up-log-data 0 "Player number: %d" c: my-player-number)

(disable-self)

)

## up-modify-escrow

语法

up-modify-escrow (in)(const)资源 数学运算符 (in)(op)值

功能

为储备资源执行数学运算。(in)(const)资源 只能是4种基础资源。

示例

1. 设置食物储备资源为100。

(defrule

(true)

=>

(up-modify-escrow food c:= 100)

(disable-self)

)

## up-modify-flag

语法

up-modify-flag (inout)(goal)目标编号 数学运算符 (in)(op)标志位

功能

修改储存在目标变量值中的标志位。

每个标志位只有两种状态：[已附加] 和 [已移除]。

数学运算符 只能是 [c/g/s]:+（用于附加标志位），以及 [c/g/s]:-（用于移除标志位）。

标志位允许我们通过2的次幂把多个 [0-1状态值] 集中储存到一个目标变量中（如1，2，4，8，16……），这可以大大节省变量个数。

实测在21 ~ 224之间的标志位可稳定存在，更高位可能会因为 [已附加] 的标志位过多而失效（整个变量溢出变成负数）。

相比于通过up-modify-goal来实现二进制位信息储存，使用这个语句会更可靠，因为：

假设gl-settings已经为8，这时若重复(up-modify-goal gl-settings c:+ 8)，就会变成16，导致设定错乱；

而(up-modify-flag gl-settings c:+ 8) 无论重复多少次，都只会是8。

标志位与十进制数的关系：

比如，如果目标变量里存的十进制数为15，那其二进制表示为0000 0000 … 1111，那标志位8、4、2、1都将为 [已附加] 状态。这时如果(up-modify-flag <goal\_id> c:- 4)，目标变量就会变成11，即0000 0000 … 1011。

再如，如果目标变量里存的十进制数为 -3，那其二进制表示为1111 1111 … 1101，那标志位16777215、8388607……8、4、1都将为 [已附加] 状态，而标志位2为 [已移除] 状态。这时如果(up-modify-flag <goal\_id> c:- 4)，目标变量就会变成 -7，即1111 1111 … 1001。

示例

1. 把一个标志位附加给储存于gl-settings的值。

(defconst gl-settings 101)

(defconst first-state 1)

(defconst second-state 2)

(defconst third-state 4)

(defrule

(true)

=>

(up-modify-flag gl-settings c:+ second-state)

(disable-self)

)

2. 利用目标变量作为媒介，实现对sn-object-repair-level更可靠的修改（单独判断各种建筑是否需要维修，从而单独附加/移除它们对应的标志位）。

(defconst temp-flag 100) ;临时变量，用于flag操作。

(defrule

(true)

=>

(up-modify-goal temp-flag s:= sn-object-repair-level) ;获取sn到goal。

)

(defrule

(true) ;假设这是“城镇中心需要维修”的判断语句。

=>

(up-modify-flag temp-flag c:+ 2) ;给goal附加标志位[2]。

)

;（其他建筑的维修同理。故此处省略其他“XXXX需要维修”的规则。）

(defrule

(true)

=>

(up-modify-sn sn-object-repair-level g:= temp-flag) ;更新sn。

)

## up-modify-goal

语法

up-modify-goal (inout)(goal)目标编号 数学运算符 (in)(op)值

功能

为给定目标储存的值执行数学运算。

本命令可作为事实或动作使用。

示例

1. 把gl-sheep-total存储的值加1。

(defconst gl-sheep-total 101)

(defrule

(true)

=>

(up-modify-goal gl-sheep-total c:+ 1)

(disable-self)

)

## up-modify-group-flag

语法

up-modify-group-flag (in)(const)修改方式 类型操作符 (in)(op)搜索组编号

功能

对给定搜索组里的单位的“ctrl编队编号标志”执行附加或移除(类似于shift + 0~9)。

使用本命令前，最好先通过up-create-group创建相应的搜索组。

这个命令并不会像ctrl + 0~9那样粗暴地覆盖整个编队，而是把新单位添加到编队或从编队移除。

~~（你最好小心管理编队编号以避免意外情况。比如，在把其他搜索组附加到一个已使用的编队前，最好先移除这个已用编队里单位的编号，避免新旧单位共存。）~~

你可以利用object-data-group-flag快速找到地图上的已有编队编号的单位，object-data-group-flag就是他们身上的编队编号。

(in)(const)修改方式 为0时表示移除，为1时表示附加。

(in)(op)搜索组编号 只能取0~9，共10个搜索组。

示例

1. 设置位于2号搜索组里的单位的编队编号为2，相当于用“shift + 2”把2号搜索组里的单位附加到2号编队里。

(defrule

(true)

=>

(up-modify-group-flag 1 c: 2) ; object-data-group-flag 将为 2

)

## up-modify-sn

语法

up-modify-sn (inout)(sn)策略值编号 数学运算符 (in)(op)值

功能

为给定策略值值执行数学运算。

示例

1. 把sn-maximum-town-size存储的值减2。

(defrule

(true)

=>

(up-modify-sn sn-maximum-town-size c:- 2)

(disable-self)

)

## up-release-escrow

语法

up-release-escrow

功能

把所有储备资源值设为0。

示例

1. 清空所有储备资源值。

(defrule

(true)

=>

(up-release-escrow)

(disable-self)

)

## up-remove-objects

语法

up-remove-objects (in)(const)搜索源 (in)(const)物件数据 类型操作符 (in)(op)值

功能

把满足指定物件数据条件的单位从搜索结果中剔除。

(in)(const)物件数据 若写为-1，则会根据物体在搜索结果列表中的索引值来判断。

示例

1. 将生命值低于20的单位从本地搜索结果中剔除。

(defrule

(true)

=>

(up-remove-objects search-local object-data-hitpoints < 20)

)

2. 先根据生命值从低到高排序，然后将索引大于0的结果从远程结果中剔除，只保留第一个结果（也就是生命值最低的那个）。

(defrule

(true)

=>

(up-clean-search search-remote object-data-hitpoints search-order-asc)

(up-remove-objects search-remote object-data-index > 0)

)

## up-request-hunters

语法

up-request-hunters 类型操作符 (in)(op)值

功能

尝试请求给定数量的猎人加入引猪的队伍。

这只会对通过战略常数(sn) 而非DUC(直接单位控制) 引诱的野猪生效。

这并不能保证实际的猎人们能到达所需总数。

示例

1. 请求4个猎人加入当前引猪的队伍。

(defrule

(true)

=>

(up-request-hunters c: 4)

(disable-self)

)

## up-research

语法

up-research (in)(goal)储备资源状态 类型操作符 (in)(op)科技编号

功能

使用变量，向研究队列中添加一项科技。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 不使用储备资源，研究箭羽。

(defrule

(up-can-research 0 c: ri-fletching)

=>

(up-research 0 c: ri-fletching)

)

## up-reset-attack-now

语法

up-reset-attack-now

功能

重置由attack-now设置的无限指派循环。

示例

1. 重置活动编队的attack-now标志。

(defrule

(true)

=>

(up-reset-attack-now)

(disable-self)

)

## up-reset-building

语法

up-reset-building (in)(const)保留当前研究 类型操作符 (in)(op)建筑编号

功能

停止所有给定类型建筑的活动和研究进程。

(in)(const)保留当前研究 若设为1，则正在研究科技的建筑不会被停止。

示例

1. 停止所有城镇中心的当前活动，正在执行研究的除外。

(defrule

(true)

=>

(up-reset-building 1 c: town-center)

)

## up-reset-cost-data

语法

up-reset-cost-data (out)(goal)目标编号

功能

把连续4个储存成本的目标变量值设为0。

(out)(goal)目标编号 应为连续4个拓展目标的首个，有效目标编号为41~508。

示例

1. 设置gl-cost-food, gl-cost-wood, gl-cost-stone, gl-cost-gold为0。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defrule

(true)

=>

(up-reset-cost-data gl-cost-food)

(disable-self)

)

## up-reset-filters

语法

up-reset-filters

功能

重置过滤式搜索系统的搜索索引和筛选状态。

搜索索引将回拨到全图物件列表的起始处。

所有筛选状态将被设为-1。

此外，可使用up-reset-search清空搜索结果。

示例

1. 清空搜索索引和筛选状态。

(defrule

(true)

=>

(up-reset-filters)

)

## up-reset-group

语法

up-reset-group 类型操作符 (in)(op)搜索组编号

功能

清空一个搜索组里的所有单位。

示例

1. 清空第一个(编号0)搜索组。

(defrule

(true)

=>

(up-reset-group c: 0)

)

## up-reset-placement

语法

up-reset-placement 类型操作符 (in)(op)建筑编号

功能

受阻碍时，清除给定类型建筑的放置队列。

请谨慎使用。

示例

1. 清除所有还未放置地基的待建兵营。

(defrule

(true)

=>

(up-reset-placement c: barracks)

(disable-self)

)

## up-reset-scouts

语法

up-reset-scouts

功能

停止并解散所有陆地探索单位编组。

示例

1. 配置必需的策略值之后，停止所有陆地探索单位。

(defrule

(true)

=>

(set-strategic-number sn-number-explore-groups 0)

(set-strategic-number sn-total-number-explorers 0)

(up-reset-scouts)

(disable-self)

)

## up-reset-search

语法

up-reset-search (in)(const)本地索引 (in)(const)本地列表 (in)(const)远程索引 (in)(const)远程列表

功能

为搜索系统重置搜索状态。

四个参数分别控制：①本地物件列表搜索索引偏移量，②当前搜索的本地物件列表，③远程物件列表搜索索引偏移量，④当前搜索的远程物件列表。

参数设为1代表重置，0代表保留。

其中“搜索索引偏移量”是指：搜索暂停或结束时所处的单位与物件列表的首个单位之间的索引差值，继续搜索时会跳转到(首个单位索引+偏移量)对应的单位处启动搜索，重置后偏移量为0，搜索到最后一个单位时为最大。

这个语句能让我们选择性地清空某个搜索列表，或者重头开始搜索。比如保留本地结果，只清空远程结果。

示例

1. 清空所有搜索状态。

(defrule

(true)

=>

(up-reset-search 1 1 1 1)

)

2. 保留本地结果，只清空远程结果，索引偏移也都保留。

(defrule

(true)

=>

(up-reset-search 0 0 1 0)

)

## up-reset-target-priorities

语法

up-reset-target-priorities (in)(const)优先级类型 (in)(const)模式

功能

重置或清空 进攻性/防御性优先级。

(in)(const)模式 为0会恢复为默认优先级。

对于防御性优先级，设置 (in)(const)模式 为1会重置所有防御性优先级为-1。

对于进攻性优先级，设置 (in)(const)模式 为1会重置单位类型为0，同时种属重置为-1。

示例

1. 重置防御性优先级为默认值。

(defrule

(true)

=>

(up-reset-target-priorities priority-defense 0)

(disable-self)

)

2. 清除所有进攻性优先级（没有目标具有优先级）。

(defrule

(true)

=>

(up-reset-target-priorities priority-offense 1)

(disable-self)

)

## up-reset-unit

语法

up-reset-unit 类型操作符 (in)(op)单位编号

功能

停止给定类型单位的行动。

这等效于在游戏中选择该种的所有单位，按下“停止”按钮。

示例

1. 停止地图上的所有长矛兵。

(defrule

(true)

=>

(up-reset-unit c: spearman-line)

)

## up-retask-gatherers

语法

up-retask-gatherers (in)(const)资源 类型操作符 (in)(op)值

功能

重新指派给定数量的村民收集某种资源。

(in)(const)资源 可以是四种基本资源，也可以是某种资源物件的种属编号。

示例

1. 尝试重新指派3个伐木工。

(defrule

(true)

=>

(up-retask-gatherers wood c: 3)

(disable-self)

)

## up-retreat-now

语法

up-retreat-now

功能

撤退所有军事单位到主城镇中心。

示例

1. 全军撤退。

(defrule

(true)

=>

(up-retreat-now)

(disable-self)

)

## up-retreat-to

语法

up-retreat-to (in)(const)物件编号 类型操作符 (in)(op)单位编号

功能

命令给定类型的所有单位撤退到一个随机物件。

(in)(const)物件编号 为集合处，可以是单位或建筑。

(in)(op)单位编号 为要撤退的单位，支持单位编号、种属、兵系。写为-1并不能表示所有单位。DE版Build 37650后支持使用兵系(xxx-line)。

这个语句可以很方便地实现军队的聚集。先用up-send-flare发送闪光到前线某处，再让大部队“撤退”到闪光处以形成攻势的攻击手法，被称为“RTF(retreat-to-flare)”。

不过DE版截至Build 43210，仍然无法撤退到flare，所以RTF手法暂时不能用。

示例

1. 如果可能的话，命令所有骑士撤退到一个随机城堡。

(defrule

(true)

=>

(up-retreat-to castle c: knight-line)

(disable-self)

)

## up-sell-commodity

语法

up-sell-commodity 类型操作符 (in)(op)资源编号 类型操作符 (in)(op)值

功能

在市场里卖出一定数量的资源。

实际卖出的数量取决于实际可用的资源量。

示例

1. 尝试在市场里卖出200食物。

(defrule

(true)

=>

(up-sell-commodity c: wood c: 200)

)

## up-send-flare

语法

up-send-flare (in)(goal)点

功能

在给定点处向盟友发送一个闪光信号。

示例

1. 在目标玩家(sn-target-player-number)处向盟友发送一个闪光信号。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-target gl-point-x)

(up-send-flare gl-point-x)

)

## up-send-scout

语法

up-send-scout (in)(const)编组类型 (in)(const)侦查方法

功能

命令给定编组类型的陆地或海上侦察单位到给定的位置去。

示例

1. 命令陆地侦察单位到地图对面去。

(defrule

(true)

=>

(up-send-scout group-type-land-explore scout-opposite)

(disable-self)

)

## up-set-attack-stance

语法

up-set-attack-stance (in)(const)单位编号 类型操作符 (in)(op)攻击姿态

功能

设置给定类型的所有单位的攻击姿态。

(in)(const)单位编号 支持单位编号、种属、兵系。写为-1时，表示不作限制，对任何种类的单位都设置。

示例

1. 设置所有长矛兵为“原地坚守”姿态。

(defrule

(true)

=>

(up-set-attack-stance spearman-line c: stance-stand-ground)

(disable-self)

)

2. 设置所有种类单位的攻击姿态为“防御”姿态。

(defrule

(true)

=>

(up-set-attack-stance -1 c: stance-defensive)

)

## up-set-defense-priority

语法

up-set-defense-priority 类型操作符 (in)(op)建筑编号 类型操作符 (in)(op)值

功能

给一种建筑设置防御性（TSA Town-Size-Attack）寻敌(Targeting)优先级。

对单位设置优先级是没有效果的。

(in)(op)值 范围为-1~32767，写为-1时为最低优先级。

AI会依据这个优先级和sn-maximum-town-size之类的值，控制单位去攻击城镇内的敌方建筑物。

默认优先级见：默认防御优先级。

示例

1. 将伐木场的防御性寻敌优先级提高到1000。

(defrule

(true)

=>

(up-set-defense-priority c: lumber-camp c: 1000)

(disable-self)

)

## up-set-event

语法

up-set-event 类型操作符 (in)(op)事件编号 类型操作符 (in)(op)值

功能

设置触发AI事件的值，用于向自己或其他AI传递信息。

(in)(op)值 可取-32768~32767。非0为激活，0为注销。

AI事件应用于触发效果“发送AI信号(事件/触发器)”和AI事实“event-detected”。

示例

1. 设置1号事件值为1，以激活该事件。

(defrule

(true)

=>

(up-set-event c: 1 c: 1)

)

## up-set-group

语法

up-set-group (in)(const)搜索源 类型操作符 (in)(op)搜索组编号

功能

拷贝某个搜索组里的全部单位，放到本地/远程搜索结果中。

注意，搜索组里的单位会覆盖搜索结果列表，而不是追加到已有结果后面。

(in)(op)搜索组编号 可为0~9，共10个搜索组。

这个指令与up-create-group配合时，能让我们把一组保存好的单位直接加入到搜索结果，而不需要再进行一次比较占用性能的搜索。

此外，再配合up-remove-objects、up-clean-search、up-find-\*等语句，还可以实现对搜索组的剔除、去重/排序、追加并更新等操作。

示例

1. 把搜索组#0里的单位拷贝给本地搜索结果。

(defrule

(true)

=>

(up-set-group search-local c: 0)

)

2. 载入搜索组#0、剔除非空闲者，搜索自己的步弓系单位并追加到结果后面，再对结果列表去重、剔除非空闲者后，更新搜索组。（这样搜索组#0就储存有一些空闲的步弓系、且实时更新啦）

(*defrule*

    (true)

*=>*

    (up-set-group search-local c: 0)

    (up-remove-objects search-local object-data-idling c:== 0)

    (up-find-local c: archer-line c: 40)

    (up-clean-search search-local -1 search-order-asc)

    (up-remove-objects search-local object-data-idling c:== 0)

    (up-create-group 0 0 c: 0)

)

## up-set-indirect-goal

语法

up-set-indirect-goal 类型操作符 (in)(op)目标编号 类型操作符 (in)(op)值

功能

以给定数值作为目标的编号，从而间接设置该编号所对应的目标变量的值。

这个语句的一个应用是 [[UP1.5]变量存取的最佳方式 指标、指针、阵列的使用](https://www.hawkaoe.net/bbs/thread-146465-1-1.html)。

示例

1. 通过使用100号目标间接设置2号目标的值为128。

(defconst gl-goal-id 100)

(defrule

(true)

=>

(set-goal gl-goal-id 2)

(up-set-indirect-goal g: gl-goal-id c: 128)

)

## up-set-offense-priority

语法

up-set-offense-priority 类型操作符 (in)(op)物件编号 类型操作符 (in)(op)值

功能

设置给定物件的进攻性优先级，用于影响部队的进攻性寻敌策略。

(in)(op)物件编号 可以为单位或建筑。

(in)(op)值 范围为-1~11，写为-1时为最低优先级。

或许本语句设置的优先级，需要将sn-enable-offensive-priority设为1后才能正常生效。

默认优先级见：默认进攻优先级。

示例

1. 将矿场的进攻优先级提高到11。

(defrule

(true)

=>

(up-set-offense-priority c: mining-camp c: 11)

(disable-self)

)

## up-set-placement-data

语法

up-set-placement-data (in)(player)参考玩家 (in)(const)物件编号 类型操作符 (in)(op)选址距离

功能

为建筑的 [选址控制系统] 设置放置信息，以供up-build + place-control选址。

(in)(const)物件编号 是放置时的参考物(focus)。如果该参考玩家没有所给定的物件编号，请确保该玩家有至少一座城镇中心用于参考。如果该参考玩家已没有任何物件，那么放置信息将不会如期生效。

(in)(op)选址距离 范围为-254~254，代表的是使用up-build + place-control选址控制系统建造时，放置位置与该玩家物件的距离。

如果找到了一个目标敌人，则距离会以离该目标玩家最近的建筑为零点向前或向后偏移。否则，将使用地图中心作为参考点。

这个语句需要与配合sn-placement-zone-size、sn-placement-fail-delta、sn-placement-to-center使用。这里有个帖子讨论各个影响因素和生效机理：[place-control（选址控制系统）的影响参数的探讨交流](https://www.hawkaoe.net/bbs/thread-146375-1-1.html)。

示例

1. 把房屋建在距离自己主城镇中心后方25格处。

(defrule

(up-can-build 0 c: house)

=>

(up-set-placement-data my-player-number -1 c: -25) ; 主城镇中心 = -1

(up-build place-control 0 c: house)

)

2. 把房屋建在距离焦点玩家最近(latest)的一个伐木场前方2格处。

(defrule

(up-can-build 0 c: house)

=>

(up-set-placement-data focus-player lumber-camp c: 2)

(up-build place-control 0 c: house)

)

## up-set-precise-target-point

语法

up-set-precise-target-point (in)(goal)点

功能

设置不受地图边界约束的精确目标点，供其他命令使用。

此命令与up-set-target-point几乎等效，区别在于这个命令不会自动将点限制到地图内。

请确保该点对于up-bound-precise-point是有效的。

一个精确点的坐标应该是普通点保留2位小数再×100的结果。请务必注意，不要用普通点作为精确目标点，否则实际位置会落在地图左角，造成不必要的麻烦。

示例

1. 设置目标点为精确的地图中心。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-point-x)

(up-modify-goal gl-point-x c:\* 100)

(up-modify-goal gl-point-y c:\* 100)

(up-set-precise-target-point gl-point-x)

)

## up-set-shared-goal

语法

up-set-shared-goal 类型操作符 (in)(op)共享目标编号 类型操作符 (in)(op)值

功能

设置共享目标变量的值。

示例

1. 设置1号共享目标变量的值为0。

(defrule

(true)

=>

(up-set-shared-goal c: 1 c: 0)

)

## up-set-signal

语法

up-set-signal 类型操作符 (in)(op)信号编号 类型操作符 (in)(op)值

功能

设置AI信号的值，用于向触发系统传递信息。

(in)(op)值 可取-32768~32767。0为注销，非零为激活。

示例

1. 设置1号信号值为0，以注销该信号。

(defrule

(true)

=>

(up-set-signal c: 1 c: 0)

)

## up-set-target-by-id

语法

up-set-target-by-id 类型操作符 (in)(op)地图ID

功能

通过地图ID设置目标物件，供其他命令使用。

可以使用up-get-point和position-object等命令来引用目标物件。

若设置时，所给的(in)(op)地图ID无效或单位已死亡，则当前目标物件不会改变（例如，如果开局第一次使用该语句时，给地图ID=10000，而它是无效的，那么目标物件仍然是0）。

特别注意：目标物件的地图ID初始为0。

本命令可当做事实或动作使用。

作为事实时，如果(in)(op)地图ID无效或单位已死亡，则事实不成立。

示例

1. 设置目标物件为up-get-object-data获取的物件地图ID。

(defconst gl-stored-id 100)

(defrule

(true)

=>

(up-get-object-data object-data-id gl-stored-id)

(up-set-target-by-id g: gl-stored-id)

)

## up-set-target-object

语法

up-set-target-object (in)(const)搜索源 类型操作符 (in)(op)索引

功能

从搜索结果中设置目标物件，供其他命令使用。

可以使用up-get-point和position-object等命令来引用目标物件。

(in)(op)索引 代表搜索结果的第几个单位，索引为0的是第一个。

注意，若索引无效（即超出了 [搜索结果总数 - 1] 或搜索结果列表里没有任何单位），则当前的目标物件不会改变。

特别注意：目标物件的地图ID初始为0。也就是说，如果开局第一次使用该语句时，没有搜到任何单位的情况下指定了索引=0，但它又是无效的，那么目标物件仍然是0。

本命令可当做事实或动作使用。

作为事实时，如果索引无效，则事实不成立。

示例

1. 设置目标物件为本地搜索中第二个匹配结果。

(defrule

(true)

=>

(up-find-local c: villager-class c: 3)

(up-set-target-object search-local c: 1)

)

## up-set-target-point

语法

up-set-target-point (in)(goal)点

功能

设置目标点，供其他命令使用。

这个语句会把超出地图边界的点约束到地图内。（比如地图尺寸为120x120，如果给一个(500, 0)的点，那就会进行约束，结果目标点会变成(120, 0)。）

(in)(goal)点-x 应为连续2个拓展目标的首个，目标编号应为41~511。

示例

1. 设置目标点为地图中心。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-point-x)

(up-set-target-point gl-point-x)

)

## up-set-timer

语法

up-set-timer 类型操作符 (in)(op)定时器编号 类型操作符 (in)(op)值

功能

关闭或启用一个定时器。

(in)(op)值 写为-1可以关闭定时器。

如果写为>=0的数，则这个指令将类似enable-timer这个动作。

特别地，如果值为0，则会定时器会立刻触发。

示例

1. 使20号定时器在10秒后触发。

(defconst gl-value 100)

(defrule

(true)

=>

(up-set-timer c: 20 c: 10)

)

## up-setup-cost-data

语法

up-setup-cost-data (in)(const)重置成本 (inout)(goal)目标编号

功能

设定(绑定)四个目标变量，用于依次储存食物、木材、石料和黄金成本，供其他命令使用。

(in)(const)重置成本 若为1则四个目标的值将被重置为0，若为0则保留目标的原数据。

(inout)(goal)目标编号 应为四个连续的拓展目标的首个，有效目标编号为41~508。

示例

1. 设置(绑定) gl cost food、gl cost wood、gl cost stone和gl cost gold四个目标以存储成本数据并重设值为0。

(defconst gl-cost-food 101)

(defconst gl-cost-wood 102)

(defconst gl-cost-stone 103)

(defconst gl-cost-gold 104)

(defrule

(true)

=>

(up-setup-cost-data 1 gl-cost-food)

(disable-self)

)

## up-store-map-name

语法

up-store-map-name (in)(const)扩展名

功能

把当前地图名称储存到内部缓冲区中。

对于rms，这是地图的文件名。不过，如果地图是一个动态的加载器(loader)，比如Full Random、Random Land Map或 Blind Random，则这将是加载器(loader)名称而不是实际地图名。

对于场景地图，这将是原来保存的文件名而不是当前的文件名。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

(in)(const)扩展 设为1时会包含地图文件扩展名。

示例

1. 储存当前地图名称到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-map-name 0)

(up-chat-data-to-all "Map: %s" c: 7031232)

)

## up-store-object-name

语法

up-store-object-name

功能

把目标物件的种类(type)名称储存到内部缓冲区中。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存地图上编号为0（第一个放置的）物件的种类名称到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-set-target-by-id c: 0)

(up-store-object-name)

(up-chat-data-to-all "Object: %s" c: 7031232)

)

## up-store-player-chat

语法

up-store-player-chat (in)(player)任一玩家

功能

把单个玩家的聊天信息储存到内部缓冲区中。

注意，只有最后一条信息的最后一个单词会被储存（空格为单词之间的分隔符，如“go to”有两个单词，“威廉·华莱士”是单个单词），

而且这条信息必须出现在当前AI玩家的聊天记录里（可通过PageUP按键翻页找到）

且必须是由指定玩家向该AI玩家发送的信息（触发发送的信息不属于任何玩家）。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存焦点玩家的聊天信息到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-player-chat focus-player)

(up-chat-data-to-all "Message: %s" c: 7031232)

)

## up-store-player-name

语法

up-store-player-name (in)(player)任一玩家

功能

把单个玩家的名称储存到内部缓冲区中。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存焦点玩家的名称到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-player-name focus-player)

(up-chat-data-to-all "Player: %s" c: 7031232)

)

## up-store-tech-name

语法

up-store-tech-name 类型操作符 (in)(op)科技编号

功能

把一个科技的名称储存到内部缓冲区中。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存自己的特色科技名称到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-tech-name c: my-unique-research)

(up-chat-data-to-all "Tech: %s" c: 7031232)

)

## up-store-text

语法

up-store-text 类型操作符 (in)(op)字符串编号

功能

把语言文件（字符串表）里的一条字符串储存到内部缓冲区中。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存“Random”到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-text c: 10230) ; 10230 为“Random”在语言文件里的字符串ID

(up-chat-data-to-all "Text: %s" c: 7031232)

)

## up-store-type-name

语法

up-store-type-name 类型操作符 (in)(op)单位编号

功能

把一种物件的种类(type)名称储存到内部缓冲区中。

缓冲区可以通过 %s 和c: 7031232（而非%d）在up-chat-data-\*指令中调用（注意7031232不能储存在defconst里）。

这个缓冲区由所有AI玩家共享，所以在规则结束前(in a rule pass)请先储存数据。

示例

1. 储存兵营的种类名称“兵营”到内部缓冲区，并告知所有玩家。

(defrule

(true)

=>

(up-store-type-name c: barracks)

(up-chat-data-to-all "Type: %s" c: 7031232)

)

## up-target-objects

语法

up-target-objects (in)(const)目标来源 (in)(const)指派行动 (in)(const)阵型 (in)(const)攻击姿态

功能

“框选”本地搜索结果里的单位，以给定行动方式、攻击姿态、阵型向指定单位指派。

这个指令只能执行以下行动：action-default、action-move、~patrol、~guard、~follow、~stop、~ground、~garrison、~delete、~gather、和~none，其中action-default等效于鼠标右键。

如果本地结果数量 > 远程结果数量，则默认模式下，该语句会将本地搜索结果平均分配给远程结果，再分别行动（其他分配模式由sn-number-tasked-units控制）。

注意，如果远程搜索结果中没有单位，则有些指派行动可能无法正常执行。对于不需要其他单位作为行动的对象的情况，如action-stop和action-delete，推荐使用up-target-point。

(in)(const)目标来源 设为0时使用远程搜索结果作为目标，设为1时只把up-set-target-object设定的目标物件作为唯一目标。

示例

1. 指派通过up-find-local选中的单位向通过up-find-remote选中的目标单位巡逻。

(defrule

(true)

=>

(up-target-objects 0 action-patrol -1 -1)

)

## up-target-point

语法

up-target-point (in)(goal)点-x (in)(const)指派行动 (in)(const)阵型 (in)(const)攻击姿态

功能

“框选”本地搜索结果里的单位，以给定行动方式、攻击姿态、阵型向某个点指派。

这个命令可以执行“指派行动”里列举的所有行动方式。不过，action-default、~guard、~follow和~garrison会表现为action-move。

除了自动清除完敌人后会自动重整，action-move到点(-1,-1)也可以重整阵型（即向原地指派）。

默认模式下，该语句会让本地搜索结果均匀分成2组后，再各自行动，若想分更多组，需通过修改sn-number-tasked-units控制）。

把(in)(goal)点-x 写为0，可使用up-set-target-point存储的目标点。

特别地：使用action-train时，需改用这个语法：[up-target-point (in)(goal)储备资源状态 action-train 类型操作符 (in)(op)单位编号]。

其中(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 把通过up-find-local选中的单位指派到地图中心。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-point-x)

(up-target-point gl-point-x action-default -1 -1)

)

2. 如果找到了一座兵营，则使用action-train命令选中的建筑训练一个长枪兵。

(defrule

(up-find-local c: barracks c: 1)

=>

(up-target-point 0 action-train c: spearman)

)

## up-train

语法

up-train (in)(goal)储备资源状态 类型操作符 (in)(op)单位编号

功能

使用变量，向训练队列增加一个单位。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 使用储备资源，训练长矛兵系。

(defconst gl-escrow-state 101)

(defrule

(true)

=>

(set-goal gl-escrow-state with-escrow)

(disable-self)

)

(defrule

(up-can-train gl-escrow-state c: spearman-line)

=>

(up-train gl-escrow-state c: spearman-line)

)

## up-tribute-to-player

语法

up-tribute-to-player (in)(player)任一玩家 (in)(const)资源编号 类型操作符 (in)(op)值

功能

向另一个玩家进贡可变数量的资源。

示例

1. 向焦点玩家进贡100食物。

(defrule

(true)

=>

(set-strategic-number sn-focus-player-number 2)

(up-tribute-to-player focus-player food c: 100)

)

## up-ungarrison

语法

up-ungarrison 类型操作符 (in)(op)物件编号

功能

命令所有给定种类的物件，原地卸载其内驻扎的单位。

示例

1. 释放船坞驻扎的所有单位。

(defrule

(true)

=>

(up-ungarrison c: dock)

(disable-self)

)

2. 释放冲车驻扎的所有单位（但不知为何只对轻型冲车有效）。

(defrule

(true)

=>

(up-ungarrison c: battering-ram-line)

(disable-self)

)

## up-update-targets

语法

up-update-targets

功能

让在城镇范围内的物件立即执行更新。

示例

1. 缩小城镇范围后，立刻撤退到主城镇中心。

(defrule

(true)

=>

(set-strategic-number sn-maximum-town-size 15)

(up-update-targets)

(up-retreat-now)

(disable-self)

)

# 事实 (Facts)

## up-allied-goal

语法

up-allied-goal (in)(player)电脑同盟 (in)(goal)目标编号 比较运算符 (in)(op)值

功能

为一个同盟的目标变量执行比较操作。

由于只有电脑玩家才有目标变量，所以只能检查电脑同盟。

(in)(player)电脑同盟 可以是any-ally、any-computer-ally，也可以是自己或同盟玩家ID，不接受every-\*百搭参数。

示例

1. 检查是否有同盟的编号为gl-sheep-total的目标变量值大于6。

(defconst gl-sheep-total 101)

(defrule

(up-allied-goal any-ally gl-sheep-total > 6)

=>

(do-nothing)

)

## up-allied-resource-amount

语法

up-allied-resource-amount (in)(player)同盟 (in)(const)资源编号 比较运算符 (in)(op)值

功能

为一个同盟的某项资源的数量执行比较操作。

(in)(player)同盟 可以是any-ally、any-computer-ally、any-human-ally，也可以是自己或同盟玩家ID，不接受every-\*百搭参数。

示例

1. 检查是否有同盟的食物数量小于50。

(defrule

(up-allied-resource-amount any-ally food < 50)

=>

(do-nothing)

)

## up-allied-resource-percent

语法

up-allied-resource-percent (in)(player)同盟 (in)(const)资源编号 比较运算符 (in)(op)值

功能

为一个同盟的某项资源值的100倍执行比较操作。

(in)(player)同盟 可以是any-ally、any-computer-ally、any-human-ally，也可以是自己或同盟玩家ID，不接受every-\*百搭参数。

示例

1. 通过检查进贡费用变量，看同盟是否研究了铸币。

(defrule

(up-allied-resource-percent any-ally amount-tribute-inefficiency < 30)

=>

(chat-to-all "至少有一个同盟研究了铸币。")

(disable-self)

)

## up-allied-sn

语法

up-allied-sn (in)(player)电脑同盟 (in)(sn)策略值编号 比较运算符 (in)(op)值

功能

为一个电脑同盟的策略值执行比较操作。由于只有电脑玩家才有策略值，所以只能检查电脑同盟。

(in)(player)电脑同盟 可以是any-ally、any-computer-ally，也可以是自己或同盟玩家ID，不接受every-\*百搭参数。

示例

1. 检查是否有同盟的sn-maximum-town-size策略值大于等于30。

(defrule

(up-allied-sn any-ally sn-maximum-town-size >= 30)

=>

(do-nothing)

)

## up-attacker-class

语法

up-attacker-class 比较运算符 (in)(op)种属编号

功能

检查上一次触发town-under-attack的敌对物件是否为给定的种属。

示例

1. 检查上次进攻是否来自于战船。

(defrule

(up-attacker-class == warship-class) ; warship-class = 922

=>

(do-nothing)

)

## up-building-type-in-town

语法

up-building-type-in-town 类型操作符 (in)(op)建筑编号 比较运算符 (in)(op)值

功能

检查城镇范围中敌人某类建筑的数量。

示例

1. 检查sn-maximum-town-size范围内是否有可作为目标的敌方兵营。

(defrule

(up-building-type-in-town c: barracks > 0)

=>

(do-nothing)

)

## up-can-build

语法

up-can-build (in)(goal)储备资源状态 类型操作符 (in)(op)建筑编号

功能

使用变量，检查是否可以修建某建筑。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 检查不使用储备资源是否可以修建哨站。

(defrule

(up-can-build 0 c: outpost)

=>

(do-nothing)

)

2. 检查使用储备资源是否可以修建哨站。

(defconst gl-escrow-state 100)

(defrule

(true)

=>

(set-goal gl-escrow-state with-escrow)

(disable-self)

)

(defrule

(up-can-build gl-escrow-state c: outpost)

=>

(do-nothing)

)

## up-can-build-line

语法

up-can-build-line (in)(goal)储备资源状态 (in)(goal)点1-x 类型操作符 (in)(op)建筑编号

功能

检查是否可以在给定点处修建一个建筑。

若想放置城镇中心和城门，请使用其地基编号，如town-center-foundation 和 gate-ascending，而不要使用town-center 或 gate。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 检查不使用储备资源是否可以在gl-point-x处修建一个木栅栏。

(defconst gl-point-x 100)

(defconst gl-point-y 101)

(defrule

(up-can-build-line 0 gl-point-x c: palisade-wall)

=>

(do-nothing)

)

## up-can-research

语法

up-can-research (in)(goal)储备资源状态 类型操作符 (in)(op)科技编号

功能

使用变量，检查是否可以研究某科技。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

示例

1. 检查不使用储备资源是否可以研究箭羽。

(defrule

(up-can-research 0 c: ri-fletching)

=>

(do-nothing)

)

## up-can-search

语法

up-can-search (in)(const)搜索源

功能

检查本地或远程的搜索状态。

如果结果列表已满，或者索引偏移位于玩家物件列表末尾（即上次搜索结束时已到达该玩家的最后一个物件，再向前搜索已无法找到新单位），将返回假（即事实不成立）。

示例

1. 检查在使用up-find-local之前是否必须执行up-reset-search。

(defrule

(not(up-can-search search-local))

=>

(up-reset-search 1 1 0 0)

)

## up-can-train

语法

up-can-train (in)(goal)储备资源状态 类型操作符 (in)(op)单位编号

功能

使用变量，检查是否可以训练某单位。

必须满足：单位已启用、有足够的资源、人口未满，单位才“可训练”。

(in)(goal)储备资源状态 要么是0(表示不使用储备)，要么是一个值为with-escrow (0)或without-escrow (1)的目标。

(in)(op)单位编号 可以是单位编号、兵系。不接受种属。

示例

1. 检查使用储备资源是否可以训练长矛兵系。

(defconst gl-escrow-state 101)

(defrule

(true)

=>

(set-goal gl-escrow-state with-escrow) ;储备资源状态 使用with-escrow(1)。

(disable-self)

)

(defrule

(up-can-train gl-escrow-state c: spearman-line)

=>

(do-nothing)

)

## up-compare-const

语法

up-compare-const (in)(const)常数 比较运算符 (in)(op)值

功能

对常数执行比较操作。

示例

1. 检查名称为feudal-villagers的常数是否大于等于20。

(defconst feudal-villagers 30)

(defrule

(up-compare-const feudal-villagers >= 20)

=>

(do-nothing)

)

## up-compare-flag

语法

up-compare-flag (in)(goal)目标编号 比较运算符 (in)(op)标志位

功能

对目标变量执行标志位比较操作。

每个标志位只有两种状态：[已附加] 和 [已移除]。

标志位允许我们通过2的次幂把多个 [0-1状态值] 集中储存到一个目标变量中（如1，2，4，8，16……），这可以大大节省变量个数。

比较运算符 只能是 [c/g/s]:==（对应标志位已附加），以及 [c/g/s]:!=（对应标志位已移除）。

标志位与十进制数的关系：

比如，如果目标变量里存的十进制数为15，那其二进制表示为0000 0000 … 1111，那标志位8、4、2、1都将为 [已附加] 状态。

再如，如果目标变量里存的十进制数为-3，那其二进制表示为1111 1111 … 1101，那标志位16777215、8388607……8、4、1都将为 [已附加] 状态，而标志位2为 [已移除] 状态。

示例

1. 检查名称为gl-guard-flag的目标变量里，是否储存有标志位guard-flag-resource（来自保卫标志位）。

(defconst gl-guard-flag 103)

(defrule

(up-compare-flag gl-guard-flag == guard-flag-resource)

=>

(do-nothing)

)

## up-compare-goal

语法

up-compare-goal (in)(goal)目标编号 比较运算符 (in)(op)值

功能

对目标变量执行比较操作。

示例

1. 检查名称为gl-sheep-total的目标变量值是否小于4。

(defconst gl-sheep-total 101)

(defrule

(up-compare-goal gl-sheep-total < 4)

=>

(do-nothing)

)

## up-compare-sn

语法

up-compare-sn (in)(sn)策略值编号 比较运算符 (in)(op)值

功能

对策略值执行比较操作。

示例

1. 检查名称为sn-maximum-town-size的策略值是否大于40。

(defrule

(up-compare-sn sn-maximum-town-size > 40)

=>

(do-nothing)

)

## up-compare-text

语法

up-compare-text 类型操作符 (in)(op)文本常数 比较运算符 (in)(op)值

功能

判断一串预定义的字符串 是否存在于缓冲区的内容中，并将其第一次出现在缓冲区的位置索引值 与 所给的值做比较。

(in)(op)文本常数，储存“预定义的字符串”，不允许直接将 "<字符串>" 写在语句里。

(in)(op)值，范围应为-1~32767。

在使用这条命令前，你必须先使用up-store-\*命令储存文本到缓冲区。

请注意，这条语句并不能检测缓冲区里出现了多少次字符串，因为：

· 如果字符串不存在于缓冲区中，那么索引值应为 -1；

· 如果字符串　存在于缓冲区中，那么索引值应为 该字符串在缓冲区中首次出现时，其前面的字符数。比如，若缓冲区中的内容为“这是一串文字”，那么预定义字符串“一”的索引值为2。

（如果给字符标上索引值，就是这样的：“0这1是2一3串4文5字”）。

示例

1. 检查玩家1的名字中是否含有“one”，这囊括了名字为“one”“one\*”“\*one”“\*one\*”等各种可能性。例外的是，“one piece”这类情况，up-store-player-name会读取到空格后面的“piece”。

(defconst text-name-one "one")

(defrule

(true)

=>

(up-store-player-name 1)

)

(defrule

(up-compare-text c: text-name-one >= 0)

=>

(do-nothing)

)

## up-defender-count

语法

up-defender-count 比较运算符 (in)(op)值

功能

检查当前主动防御城镇的单位数量。

这可能与TSA (Town-Size-Attack)和sn-number-defend-groups有关。

示例

1. 检查当前主动防御城镇的单位数量是否大于等于10。

(defrule

(up-defender-count >= 10)

=>

(do-nothing)

)

## up-enemy-buildings-in-town

语法

up-enemy-buildings-in-town 比较运算符 (in)(op)值

功能

检查城镇中可指定的敌人建筑数量。

示例

1. 检查sn-maximum-town-size范围内是否有敌方建筑。

(defrule

(up-enemy-buildings-in-town > 0)

(enemy-buildings-in-town) ; 这与上一行是等效的

=>

(do-nothing)

)

## up-enemy-units-in-town

语法

up-enemy-units-in-town 比较运算符 (in)(op)值

功能

检查城镇中可指定的敌人单位数量。

示例

1. 检查sn-maximum-town-size范围内是否存在多于5个敌方单位，不包括村民。

(defrule

(up-enemy-units-in-town > 5)

=>

(do-nothing)

)

## up-enemy-villagers-in-town

语法

up-enemy-villagers-in-town 比较运算符 (in)(op)值

功能

检查城镇中可指定的敌人村民数量。

示例

1. 检查sn-maximum-town-size范围内是否存在至少2个敌方村民。

(defrule

(up-enemy-villagers-in-town >= 2)

=>

(do-nothing)

)

## up-gaia-type-count

语法

up-gaia-type-count 类型操作符 (in)(const)资源 比较运算符 (in)(op)值

功能

检查当前可视的自然资源数量。

这条命令执行速度相对较慢，因为它必须检查所有已发现的资源的状态，所以请考虑游戏性能。

(in)(const)资源 可以是四种基本资源，也可以是某种资源物件。

示例

1. 检查是否存在至少4块已发现的金矿。

(defrule

(up-gaia-type-count c: gold >= 4)

=>

(do-nothing)

)

2. 检查是否存在多于6只已发现的绵羊或火鸡。

(defrule

(up-gaia-type-count c: livestock-class > 6) ; livestock-class = 958

=>

(do-nothing)

)

## up-gaia-type-count-total

语法

up-gaia-type-count-total 类型操作符 (in)(const)资源 比较运算符 (in)(op)值

功能

检查当前所有发现过的自然资源。

检查食物、木材、石矿或金矿的速度很快。

(in)(const)资源 可以是四种基本资源，检查食物、木材、石矿或金矿的速度很快。但不接受食物物件（如鹿和绵羊）——不过作为一种候补方案，这时命令会重定向到较慢速的up-gaia-type-count，且只检查还现存的物件。

示例

1. 检查是否已经发现100棵树，不论其是否还存在。

(defrule

(up-gaia-type-count-total c: wood > 100)

=>

(do-nothing)

)

2. 检查是否有小于2头曾发现的鹿仍然存在（会转而使用up-gaia-type-count）。

(defrule

(up-gaia-type-count-total c: deer < 2)

=>

(do-nothing)

)

## up-group-size

语法

up-group-size 类型操作符 (in)(op)搜索组编号 比较运算符 (in)(op)值

功能

检查一个搜索组里的当前单位数量。

(in)(op)搜索组编号 只能取0~9，共10个搜索组。

示例

1.检查第一个搜索组（编号为0）里是否包含单位。

(defrule

(up-group-size c: 0 > 0)

=>

(do-nothing)

)

## up-idle-unit-count

语法

up-idle-unit-count (in)(const)闲置类型 比较运算符 (in)(op)值

功能

检查给定类型的闲置单位数量。

示例

1. 检查是否存在多于5个闲置村民。

(defrule

(up-idle-unit-count idle-type-villager > 5)

=>

(do-nothing)

)

## up-object-data

语法

up-object-data (in)(const)物件数据 比较运算符 (in)(op)值

功能

检查目标物件的特定数据，与给定值比较。

在使用这条命令前，请先设定目标物件（up-set-target-by-id或up-set-target-object）。

示例

1. 检查目标物件是否驻扎有3个以上单位。

(defrule

(up-object-data object-data-garrison-count > 3)

=>

(do-nothing)

)

## up-object-target-data

语法

up-object-target-data (in)(const)物件数据 比较运算符 (in)(op)值

功能

检查目标物件(target-object)的 [行动目标单位] 的特定信息。

比如，一个剑士正在跟随/攻击一个敌方弓兵，那剑士的“行动目标单位”就是那个弓兵了。

在使用这条命令前，请先设定目标物件(up-set-target-by-id或up-set-target-object)。

示例

1. 检查目标物件的目标单位是否驻扎有3个以上单位。

(defrule

(up-object-target-data object-data-garrison-count > 3)

=>

(do-nothing)

)

## up-object-type-count

语法

up-object-type-count 类型操作符 (in)(op)物件编号 比较运算符 (in)(op)值

功能

结合了unit-type-count和building-type-count的功能。会检查 [已存在于地图上] 的该类建筑/单位的数量。

(in)(op)物件编号 可以是单位编号、建筑编号、兵系、种属。

示例

1. 检查是否存在至少10个村民。

(defrule

(up-object-type-count c: villager >= 10)

=>

(do-nothing)

)

## up-object-type-count-total

语法

up-object-type-count-total 类型操作符 (in)(op)物件编号 比较运算符 (in)(op)值

功能

结合了unit-type-count-total和building-type-count-total的功能。会检查 [已存在于地图上] + [生产队列中] 的该类建筑/单位的数量总和。

(in)(op)物件编号 可以是单位编号、建筑编号、兵系、种属。

示例

1. 检查是否有至少20个已存在+在生产队列中的村民。

(defrule

(up-object-type-count-total c: villager > 20)

=>

(do-nothing)

)

## up-path-distance

语法

up-path-distance (in)(goal)点-x (in)(const)严格 比较运算符 (in)(op)值

功能

检查从目标物件(target-object) 到 指定点 之间的路程 (向下取整)。

注意，由于点坐标只能是整数，因此实际计算时是格子左角之间的路程。

与直线距离不同，路程是考虑障碍物的。

· 如果点是无法抵达的，则路程会返回65535；

· 如果无法获取，则路程为-2。

(in)(const)严格 设为1表示不允许为了抵达而对点做偏移，这时若格子里有障碍物，可能会判定为无法抵达；

设为0表示　允许为了能抵达而对点做少量偏移（比如从树林内移到最近的树林边缘）。

把点写为0，可使用up-set-target-point存储的点位置。

示例

1. 检查目标物件是否可能到达地图中心。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-path-distance gl-center-x 0 != 65535)

=>

(do-nothing)

)

## up-pending-objects

语法

up-pending-objects 类型操作符 (in)(op)物件编号 比较运算符 (in)(op)值

功能

为生产队列中给定物件的数量执行比较操作。

(in)(op)物件编号 可以是单位编号、建筑编号、兵系、种属。

示例

1. 检查生产队列中是否存在至少1个住房。

(defrule

(up-pending-objects c: house >= 1)

=>

(do-nothing)

)

## up-pending-placement

语法

up-pending-placement 类型操作符 (in)(op)建筑编号

功能

检查给定类型的建筑是否正在等待放置位置。

这可能与 [选址控制系统] 有关，详见up-set-placement-data。

示例

1. 检查是否一座兵营正在等待放置位置。

(defrule

(up-pending-placement c: barracks)

=>

(do-nothing)

)

## up-player-distance

语法

up-player-distance (in)(player)任一玩家 比较运算符 (in)(op)值

功能

检查距离另一个玩家最近建筑的距离，以格为单位。

示例

1. 检查玩家3是否距离自己不到50格。

(defrule

(up-player-distance 3 < 50)

=>

(do-nothing)

)

2. 检查焦点玩家是否在变量gl-distance表示的格数范围内。

(defconst gl-distance 101)

(defrule

(up-player-distance focus-player g:< gl-distance)

=>

(do-nothing)

)

3. 检查是否有敌方在40格以内。

(defrule

(up-player-distance any-enemy < 40)

=>

(do-nothing)

)

## up-players-in-game

语法

up-players-in-game (in)(const)玩家外交状态 比较运算符 (in)(op)值

功能

检查给定外交状态的活动玩家数量。

示例

1. 检查是否至少有两个同盟，即本队伍至少有3个玩家。

(defrule

(up-players-in-game ally >= 3)

=>

(do-nothing)

)

## up-point-contains

语法

up-point-contains (in)(goal)点-x 类型操作符 (in)(op)物件编号

功能

检查给定坐标的格子上是否有某种物件存在。

多种物件同时在同一个格子上时，不会互相干扰。

把点写为0，可使用up-set-target-point存储的点位置。

(in)(op)物件编号 可以是单位编号、建筑编号、兵系、种属。请注意，如果使用all-units-class (-1)，可能会获取到出乎意料的物件，比如刚好飞过格子的鸟、各种地形植物等。

示例

1. 检查地图中心是否有树。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-contains gl-center-x c: tree-class) ; tree-class = 915

=>

(do-nothing)

)

## up-point-distance

语法

up-point-distance (in)(goal)点1-x (in)(goal)点2-x 比较运算符 (in)(op)值

功能

为两点之间的直线距离执行比较操作。

把点2写为0，可使用up-set-target-point存储的点。

示例

1. 检查主城镇中心与地图中心的距离是否大于5。

(defconst gl-self-x 100)

(defconst gl-self-y 101)

(defconst gl-center-x 200)

(defconst gl-center-y 201)

(defrule

(true)

=>

(up-get-point position-self gl-self-x)

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-distance gl-self-x gl-center-x > 5)

=>

(do-nothing)

)

## up-point-elevation

语法

up-point-elevation (in)(goal)点-x 比较运算符 (in)(op)值

功能

检查给定点所处的海拔高度。

把点写为0，可使用up-set-target-point存储的点。

示例

1. 检查地图中心点的海拔高度。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-elevation gl-center-x > 1)

=>

(do-nothing)

)

## up-point-explored

语法

up-point-explored (in)(goal)点-x 比较运算符 (in)(op)已探索状态

功能

检查地图上给定点是否已被探索。把点写为0，可使用up-set-target-point存储的点。

示例

1. 检查地图中心点是否已被探索。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-explored gl-center-x != explored-no)

=>

(do-nothing)

)

## up-point-terrain

语法

up-point-terrain (in)(goal)点-x 比较运算符 (in)(op)地形

功能

检查给定点所处的地形。

把点写为0，可使用up-set-target-point存储的点。

示例

1. 检查地图中心点是否为草地地形。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-terrain gl-center-x == terrain-grass)

=>

(do-nothing)

)

## up-point-zone

语法

up-point-zone (in)(goal)点-x 比较运算符 (in)(op)值

功能

检查给定点所处的地图区域编号。

每片陆地/水域都分配有不同的区域编号，这可以用于判断物件是否处于不同的陆地/水域。更多信息详见位于主大陆。

如果你没有村民（或城镇中心？），地图区域编号可能会有所不同。

把点写为0，可使用up-set-target-point存储的点位置。

示例

1. 检查地图中心点的地图区域编号。

(defconst gl-center-x 100)

(defconst gl-center-y 101)

(defrule

(true)

=>

(up-get-point position-center gl-center-x)

)

(defrule

(up-point-zone gl-center-x > 1)

=>

(do-nothing)

)

## up-projectile-detected

语法

up-projectile-detected (in)(const)抛射物类型 比较运算符 (in)(op)值

功能

检查 自最近一次给定类型抛射物向自己射出时算起 已过去的时间，单位为ms。

注意是从射出算起，不是击中己方单位后算起。

示例

1. 检查过去2秒内是否有敌方城镇中心向自己射出箭矢。

(defrule

(up-projectile-detected projectile-town-center < 2000)

=>

(do-nothing)

)

## up-projectile-target

语法

up-projectile-target (in)(const)抛射物类型 比较运算符 (in)(op)种属编号

功能

检查 最近一次向自己射出的给定类型抛射物 所指向的目标物件 是否为指定种属。

示例

1. 检查过去2秒内是否有敌方城镇中心攻击了我方弓箭手类。

(defrule

(up-projectile-detected projectile-town-center < 2000)

(up-projectile-target projectile-town-center == archery-class) ; archery-class = 900

=>

(do-nothing)

)

## up-remaining-boar-amount

语法

up-remaining-boar-amount 比较运算符 (in)(op)值

功能

检查当前猪所剩食物量。

只有在另一头猪可捕猎、且当前猪是通过 策略值(sn) 而非 DUC(直接单位控制) 引诱时，本数据才有效，否则数据会是65535表示这是最后一头猪。

示例

1. 检查在有另一头猪的情况下，当前猪所剩食物量是否小于210。

(defrule

(up-remaining-boar-amount < 210) ; 若无效，则为65535

=>

(do-nothing)

)

## up-research-status

语法

up-research-status 类型操作符 (in)(op)科技编号 比较运算符 (in)(op)研究状态

功能

检查给定科技的研究状态。

示例

1. 检查织布机是否正在研发或已研发完毕。

(defrule

(up-research-status c: ri-loom >= research-pending)

=>

(do-nothing)

)

## up-resource-amount

语法

up-resource-amount (in)(const)资源编号 比较运算符 (in)(op)值

功能

为自己的资源值执行比较操作。

理论上支持全部资源编号。

示例

1. 检查当前AI是否正占有至少1个圣物。

(defrule

(up-resource-amount amount-relics >= 1)

=>

(do-nothing)

)

## up-resource-percent

语法

up-resource-percent (in)(const)资源编号 比较运算符 (in)(op)值

功能

为自己的资源值的100倍执行比较操作（用于判断值为小数的资源）。

理论上支持全部资源编号。

示例

1. 通过检查进贡费用资源值，来判断是否研究了铸币。

(defrule

(up-resource-percent amount-tribute-inefficiency < 30)

=>

(chat-to-all "铸币已经研究完成了。")

(disable-self)

)

## up-timer-status

语法

up-timer-status (in)(const)定时器编号 比较运算符 (in)(op)定时器状态

功能

检查定时器是否为禁用、激活、运行状态或者它们的组合。

激活是指 定时器已启用、且倒计时已归零；

运行是指 定时器已启用、且倒计时未归零。

示例

1. 检查1号定时器是否为禁用或激活状态。

(defconst tm-gathering 1)

(defrule

(up-timer-status tm-gathering <= timer-triggered)

=>

(do-nothing)

)

2. 检查1号定时器是否正在运行。

(defconst tm-gathering 1)

(defrule

(up-timer-status tm-gathering == timer-running)

=>

(do-nothing)

)

## up-train-site-ready

语法

up-train-site-ready 类型操作符 (in)(op)单位编号

功能

检查给定单位的训练场所是否可用且可以训练。

示例

1. 检查是否有船坞可以开始训练渔船。

(defrule

(up-train-site-ready c: fishing-ship)

=>

(do-nothing)

)

## up-unit-type-in-town

语法

up-unit-type-in-town 类型操作符 (in)(op)单位编号 比较运算符 (in)(op)值

功能

检查城镇中敌人某种特定单位的数量。

(in)(op)单位编号 可以是单位编号、兵系、种属，也可以用 -1表示不限单位。

示例

1. 检查sn-maximum-town-size范围内是否存在敌方骑士。

(defrule

(up-unit-type-in-town c: knight-line > 0)

=>

(do-nothing)

)

## up-villager-type-in-town

语法

up-villager-type-in-town 类型操作符 (in)(op)单位编号 比较运算符 (in)(op)值

功能

检查城镇中敌人某种特定村民的数量。

示例

1. 检查sn-maximum-town-size范围内是否存在敌方金矿工。

(defrule

(or

(up-villager-type-in-town c: gold-miner-m > 0) ; gold-miner-m = 579，男金矿工

(up-villager-type-in-town c: gold-miner-f > 0) ; gold-miner-f = 581，女金矿工

)

=>

(do-nothing)

)

# 附录

## 语法说明

(in) 表示输入参数。

(out) 表示输出参数。

(inout) 表示输入/输出参数。

(const) 该参数应为任意数字或defconst定义的常数。

(goal) 该参数应为目标数字或defconst定义的目标常数。

(sn) 该参数应为策略值或defconst定义的策略值常数。

(op) 根据前方的类型操作符，选择const, goal, sn中的一种。

(timer) 该参数应为定时器编号或defconst定义的定时器编号常数。

(player) 该参数应为表示玩家的数字、defconst定义的玩家编号、表示玩家的百搭常数、或特殊标识符（focus-player和target-player）。有些语句可能不支持every-百搭常数，因为只能作用于单个玩家。

(text) 该参数应为字符串。

## 类型操作符

c: 即const，下一个数值识别为常数（包括系统定义常数）。

g: 即goal，下一个数值识别为目标。

s: 即sn，下一个数值识别为策略值。

[UserPatch人工智能简单教程](https://www.hawkaoe.net/bbs/thread-118795-1-1.html) 这个教程里详细举例说明了这些操作符、运算符的功能。

## 比较运算符

由c:, g:, s:后接>, >=, <, <=, ==, !=组成，用于指定下一个数值识别为何种类型。  
其中c:可以忽略不写，因为>, >=, <, <=, ==, !=这些默认是将下一个数值识别为常数的。  
其中==代表等于，!=代表不等于。

## 数学运算符

由c:, g:, s:后接=, +, -, \*, /, mod, min, max, neg, z/, %\*, %/ 组成。其结果会储存到up-modify-[const/flag/goal/sn] 语句的(inout)处，即这些语句的第一个参数。

\* 将 a 的数值乘以 b 的数值。

/ 将 a 的数值除以 b 的数值，并四舍五入为整数。例如 8 c:/ 3 = 2.666… = 3。

z/ 将 a 的数值除以 b 的数值，并向下舍入为整数。例如 8 c:z/ 3 = 2.666… = 2。

+ 将 a 的数值加以 b 的数值。

- 将 a 的数值减以 b 的数值。

mod 将 a 的数值除以 b 的数值，并且回报余数。例如 8 c:mod 3 = 2 (商2余2)。

%\* 以 b 的数值作为百分比值，并将 a 乘以它，亦即 (a \* b) / 100。例如 8 c:%\* 3 = 8 \* 3 / 100 = 0.24 = 0 (舍入)。

%/ 以 b 的数值作为百分比值，并将 a 除以它，亦即 (a \* 100) / b。例如 8 c:%/ 3 = 8 \* 100 / 3 = 266.7 = 267 (舍入)。

= 将 a 的数值设为 b 的数值。（不要与“==”混淆了)

neg 将 a 的数值设为 b 的数值的负数。例如 8 c:neg 3 = -3

min 返回两者间数值较低的一方。例如 8 c:min 3 = 3。

max 返回两者间数值较高的一方。例如 8 c:max 3 = 8。

## 默认防御优先级

注：这很可能是用于TSA防御性进攻时的“清理入侵单位”的优先级，不太可能是对AI自己的建筑的“保护”优先级。

-1: 默认值

5: 哨站

10: 农田, 渔网

20: 铁匠铺, 大学

30: 伐木场

40: 磨坊

50: 矿场

100: 市场

250: 住房

500: 兵营, 靶场, 马厩, 修道院, 攻城武器厂, 船坞, 城门

1000: 城镇中心 (109, 建成后), 城镇中心 (621, 地基)

2000: 瞭望台

3000: 防御箭塔

4000: 要塞

5000: 炮塔

10000: 城堡

20000: 奇观

## 默认进攻优先级

注：以下右侧说明中，除了用<xx系>，其余都是种属。且只是概括性质，并非与左侧id按顺序逐一对应。

-1: 默认值

00: 903, 927, 939, 949, 951, 952, 954 普通建筑、城墙城门、农田、巨型投石机、塔种属

01: 906, 912, 924, 935, 945, 947, 950, 956, 957 步兵、骑兵、战象、斥候、<无用种属>……

02: 928, 946 <无用种属>

03: 900, 923, 926, 936, 944 弓箭手、征服者、骑象射手、骑射手、火枪手种属

04: 6, 7 <掷矛手系>

05: 35, 422, 548 <冲车系>

06: 913, 955 攻城武器种属、弩炮

07: 904, 919, 958, 959 村民、贸易车、牲畜、国王

08: 917, 918, 943 僧侣种属

09: 922, 953 战舰种属、登舰船

10: 920 运输船

11: 902, 921 商船、渔船

## #load-if 符号扩展

（DE拓展了更多符号，待加入，暂时到这个网站查看<https://airef.github.io/tables/load-if.html>）

UP-GAME-[NAME] 定义当前扩展包。

UP-GAME-AGE2-X1 定义帝国时代2：征服者资料片。

UP-GAME-AGE2-X2 定义帝国时代2：被遗忘的帝国。

UP-AVAILABLE 定义1.1及以后的游戏版本。用来检测UserPatch的存在。

UP-VERSION-1.3 定义1.3及以后的游戏版本。

UP-VERSION-1.2 定义1.2及以后的游戏版本。

UP-VERSION-1.1 定义1.1及以后的游戏版本。

DE-AVAILABLE 定义帝国时代2：决定版（DE）

UP-PROCESS-60FPS 定义游戏采用60fps运行，只在单人游戏中可能存在。

UP-[NAME]-CIV-ALLY 定义同盟玩家。

UP-[NAME]-CIV-ENEMY 定义非同盟玩家。

UP-[1~8]-PLAYER-GAME 定义玩家数量。

UP-[1~8]-PLAYER-TEAM 定义队伍中的玩家数量。

UP-ALLY-IN-GAME 定义玩家拥有至少一个同盟。

UP-MULTIPLE-ENEMIES 定义玩家有多于一个敌人。

UP-TEAM-DISADVANTAGE 定义至少一支敌方队伍人数多于我方。

UP-MULTIPLAYER-GAME 定义多人游戏。

UP-HUMAN-IN-GAME 定义至少有一个人类玩家。

UP-NOMAD-STYLE 定义rms脚本中定义了游牧地图样式。

UP-MICHI-STYLE 定义rms脚本中定义了黑森林地图样式。

UP-NOMAD-RESOURCES 定义rms脚本中定义的nomad\_resources命令。

UP-GROUPED-BY-TEAM 定义rms脚本中定义了grouped\_by\_team命令。

UP-POCKET-POSITION 定义玩家坐中。

UP-SCENARIO-GAME 定义自定义场景游戏。

TEAM-TOGETHER 定义“组队玩家”选项。

ALL-TECHS-ENABLED 定义“完整科技”选项。

TURBO-RANDOM-MAP 定义极速随机地图模式。

VICTORY-RELICS 定义圣物胜利条件。

POPULATION-CAP-EXTENDED 定义人口上限多于200。

POPULATION-CAP-200 定义人口上限不少于200。

POPULATION-CAP-X 定义人口上限为X，X取值125，375，675等等。

DARK-AGE-END 定义封顶时代为黑暗时代。

FEUDAL-AGE-END 定义封顶时代为封建时代。

CASTLE-AGE-END 定义封顶时代为城堡时代。

IMPERIAL-AGE-END 定义封顶时代为帝王时代。

LUDIKRIS-MAP 定义地图大小为超大。

# 常数清单

注：本部分来源于UserPatchConst.per。若发现有所缺漏，可自行查找源常数清单。

## 编组类型

(defconst group-type-land-attack 100) 陆地进攻

(defconst group-type-land-explore 101) 陆地探索

(defconst group-type-water-attack 102) 海上进攻

(defconst group-type-water-explore 103) 海上探索

(defconst group-type-transport-ship 104) 运输船

(defconst group-type-fishing-ship 105) 渔船

(defconst group-type-water-trade 106) 海上贸易

(defconst group-type-forward-builder 107) 前置建筑者

(defconst group-type-monk 108) 僧侣

(defconst group-type-land-trade 109) 陆地贸易

## 保卫标志位

(defconst guard-flag-victory 1) 胜利 （失去所有保卫对象单位后，当前玩家会失败）

(defconst guard-flag-resource 2) 资源 （持有保卫对象单位期间，当前玩家能像拥有圣物一样缓慢获得资源）

(defconst guard-flag-inverse 4) 相反 （需与guard-flag-resource同时存在才能发挥效果。失去所有保卫对象单位后，当前玩家才能获得资源）

## 储备资源状态

注：储备资源状态要么是0(表示不使用储备)，要么是一个值为0或1的目标。特别注意，直接使用with-escrow和without-escrow这两个常数作为参数时是无效的。

(defconst with-escrow 0) 使用储备资源

(defconst without-escrow 1) 不使用储备资源

## 村民语言编号

(defconst lid-villager-idle 5121) 无业村民

(defconst lid-villager-shepherd 5496) 牧羊人

(defconst lid-villager-farmer 5123) 农夫

(defconst lid-villager-forager 5402) 粮草征收人

(defconst lid-villager-hunter 5124) 猎人

(defconst lid-villager-fisherman 5499) 渔夫

(defconst lid-villager-lumberjack 5125) 伐木工

(defconst lid-villager-gold-miner 5482) 金矿工

(defconst lid-villager-stone-miner 5126) 石矿工

(defconst lid-villager-builder 5122) 建筑工

(defconst lid-villager-repairer 5127) 修理工

## 地形

注：第一部分适用于AOC。地形有变化的版本如WK、DE可能不完全适用。  
（DE地形表已制作，见下文）

(defconst terrain-grass 0) 草1

(defconst terrain-grass2 12) 草2

(defconst terrain-grass3 9) 草3

(defconst terrain-grass-snow 34) 雪原

(defconst terrain-grass-old 16) 草(其他)

(defconst terrain-water 1) 浅水

(defconst terrain-water-deep 22) 深水

(defconst terrain-water-medium 23) 中水

(defconst terrain-water-bridge 28) 水桥

(defconst terrain-water-old 15) 浅水(其他)

(defconst terrain-water-beach 2) 海滩

(defconst terrain-shallows 4) 浅滩

(defconst terrain-ice 26) 冰

(defconst terrain-ice2 35) 冰2

(defconst terrain-ice-beach 37) 冰海滩

(defconst terrain-desert 14) 沙漠

(defconst terrain-dirt 6) 泥1

(defconst terrain-dirt2 11) 泥2

(defconst terrain-dirt3 3) 泥3

(defconst terrain-dirt-snow 33) 雪泥地

(defconst terrain-farm 7) 农田

(defconst terrain-farm1 29) 农田1

(defconst terrain-farm2 30) 农田2

(defconst terrain-farm3 31) 农田3

(defconst terrain-farm-dead 8) 荒芜农田

(defconst terrain-forest 10) 森林

(defconst terrain-forest-bamboo 18) 竹林

(defconst terrain-forest-jungle 17) 热带丛林

(defconst terrain-forest-oak 20) 橡树林

(defconst terrain-forest-palm 13) 棕榈沙漠

(defconst terrain-forest-pine 19) 松树林

(defconst terrain-forest-snow 21) 雪松林

(defconst terrain-foundation 27) 建筑

(defconst terrain-foundation-snow 36) 雪地建筑

(defconst terrain-koh 40) 道路(其他)

(defconst terrain-leaves 5) 树叶

(defconst terrain-road 24) 道路

(defconst terrain-road-broken 25) 烂路

(defconst terrain-road-snow 38) 雪路

(defconst terrain-road-fungus 39) 蘑菇路

(defconst terrain-snow 32) 雪

(defconst terrain-unknown 41) 未知

注：以下为DE专用（于Build 43210制表），且不在原版UserPatchConst.per里，请复制到你的AI文件里自行defconst。

(defconst terrain-de-water-2d-bridge 28) ;2D 水域，桥梁

(defconst terrain-de-underbrush 5) ;矮树丛

(defconst terrain-de-underbrush-jungle 77) ;矮树丛，丛林

(defconst terrain-de-underbrush-snow 72) ;矮树丛，积雪

(defconst terrain-de-underbrush-leaves 71) ;矮树丛，叶子

(defconst terrain-de-ice 35) ;冰原

(defconst terrain-de-ice-navigable 26) ;冰原，可通航

(defconst terrain-de-grass-jungle 60) ;草，丛林

(defconst terrain-de-grass1 0) ;草地 1

(defconst terrain-de-grass2 12) ;草地 2

(defconst terrain-de-grass3 9) ;草地 3

(defconst terrain-de-grass-jungle-rainforest 83) ;草地，丛林 (雨林)

(defconst terrain-de-grass-dry 100) ;草地，干枯

(defconst terrain-de-grass-foundation 27) ;草地，根基

(defconst terrain-de-road 24) ;道路

(defconst terrain-de-road-fungus 75) ;道路，蘑菇

(defconst terrain-de-road-gravel 78) ;道路，沙砾

(defconst terrain-de-corruption 69) ;腐蚀毒雾

(defconst terrain-de-beach 2) ;海滩

(defconst terrain-de-beach-ice 37) ;海滩，冰原

(defconst terrain-de-black 47) ;黑色

(defconst terrain-de-farm-dead 8) ;荒芜农田

(defconst terrain-de-desert-cracked 45) ;裂开的沙漠

(defconst terrain-de-water-2d-shoreless 15) ;没有海滩的 2D 水域

(defconst terrain-de-dirt1 6) ;泥地 1

(defconst terrain-de-dirt2 11) ;泥地 2

(defconst terrain-de-dirt3 3) ;泥地 3

(defconst terrain-de-dirt4 42) ;泥地 4

(defconst terrain-de-dirt-savannah 41) ;泥地，大草原

(defconst terrain-de-dirt-mud 76) ;泥地，污泥

(defconst terrain-de-farm 7) ;农田

(defconst terrain-de-farm-0 29) ;农田 0%

(defconst terrain-de-farm-33 30) ;农田 33%

(defconst terrain-de-farm-67 31) ;农田 67%

(defconst terrain-de-grass-other 16) ;其它草地

(defconst terrain-de-shallows 4) ;浅滩

(defconst terrain-de-shallows-mangrove 54) ;浅滩，红树林

(defconst terrain-de-shallows-azure 59) ;浅滩，天蓝色

(defconst terrain-de-forest-acacia 50) ;森林，刺槐

(defconst terrain-de-forest-jungle 17) ;森林，丛林

(defconst terrain-de-forest-mediterranean 88) ;森林，地中海

(defconst terrain-de-forest-bush 89) ;森林，灌木丛

(defconst terrain-de-forest-mangrove 55) ;森林，红树林

(defconst terrain-de-forest-baobab 49) ;森林，猴面包树

(defconst terrain-de-forest-dead 106) ;森林，枯死

(defconst terrain-de-forest-dragon-tree 48) ;森林，龙血树

(defconst terrain-de-forest-reeds 92) ;森林，芦苇

(defconst terrain-de-forest-reeds-shallows 90) ;森林，芦苇 (浅滩)

(defconst terrain-de-forest-reeds-beach 91) ;森林，芦苇 (沙滩)

(defconst terrain-de-forest-autumn 104) ;森林，秋季

(defconst terrain-de-forest-autumn-snow 105) ;森林，秋季雪

(defconst terrain-de-forest-palm-desert 13) ;森林，沙漠棕榈树

(defconst terrain-de-beach-wet 107) ;森林，湿润

(defconst terrain-de-forest-pine 19) ;森林，松树

(defconst terrain-de-forest-oak 10) ;森林，橡树

(defconst terrain-de-forest-oak-bush 20) ;森林，橡树丛

(defconst terrain-de-forest-pine-snow 21) ;森林，雪松

(defconst terrain-de-forest-rainforest 56) ;森林，雨林

(defconst terrain-de-forest-bamboo 18) ;森林，竹林

(defconst terrain-de-gravel-default 70) ;沙砾，默认

(defconst terrain-de-gravel-desert 102) ;沙砾，沙漠

(defconst terrain-de-desert-quicksand 46) ;沙漠，流沙

(defconst terrain-de-beach-non-navigable 79) ;沙滩 (不可航行)

(defconst terrain-de-beach-non-navigable-wet-sand 80) ;沙滩 (不可航行)，湿沙

(defconst terrain-de-beach-non-navigable-wet-gravel 81) ;沙滩 (不可航行)，湿沙砾

(defconst terrain-de-beach-non-navigable-wet-rock 82) ;沙滩 (不可航行)，湿岩石

(defconst terrain-de-beach-white 53) ;沙滩，白色

(defconst terrain-de-beach-white-vegetation 51) ;沙滩，白色，植被

(defconst terrain-de-beach-wet-gravel 108) ;沙滩，湿沙砾

(defconst terrain-de-beach-wet-rock 109) ;沙滩，湿岩石

(defconst terrain-de-beach-vegetation 52) ;沙滩，植被

(defconst terrain-de-desert-sand 14) ;砂质沙漠

(defconst terrain-de-swamp-bogland 101) ;湿地，沼泽

(defconst terrain-de-rice-farm 63) ;水稻田

(defconst terrain-de-rice-farm-0 65) ;水稻田，0%

(defconst terrain-de-rice-farm-33 66) ;水稻田，33%

(defconst terrain-de-rice-farm-66 67) ;水稻田，66%

(defconst terrain-de-rice-farm-dead 64) ;水稻田，枯死

(defconst terrain-de-water-green 95) ;水域，绿

(defconst terrain-de-water-shallow 1) ;水域，浅

(defconst terrain-de-water-deep 22) ;水域，深

(defconst terrain-de-water-deep-ocean 57) ;水域，深海

(defconst terrain-de-water-azure 58) ;水域，天蓝色

(defconst terrain-de-water-medium 23) ;水域，中等

(defconst terrain-de-water-brown 96) ;水域，棕色

(defconst terrain-de-road-broken 25) ;损坏的道路

(defconst terrain-de-snow-strong 74) ;雪，大雪

(defconst terrain-de-snow-light 73) ;雪，小雪

(defconst terrain-de-snow 32) ;雪地

(defconst terrain-de-snow-foundation 6) ;雪地，根基

(defconst terrain-de-rock1 40) ;岩石 1

## 定时器状态

(defconst timer-disabled 0) 禁用

(defconst timer-triggered 1) 已触发（倒计时到达0）

(defconst timer-running 2) 运行中

## 地基编号

(defconst town-center-foundation 621) 城镇中心地基

(defconst gate-ascending 487) 城门 (／) - 地基

(defconst gate-ascending-closed 64) 城门 (／) - 关闭

(defconst gate-ascending-open 78) 城门 (／) - 开启

(defconst gate-descending 490) 城门 (＼) - 地基

(defconst gate-descending-closed 88) 城门 (＼) - 关闭

(defconst gate-descending-open 99) 城门 (＼) - 开启

(defconst gate-horizontal 665) 城门 (－) - 地基

(defconst gate-horizontal-closed 659) 城门 (－) - 关闭

(defconst gate-horizontal-open 661) 城门 (－) - 开启

(defconst gate-vertical 673) 城门 (｜) - 地基

(defconst gate-vertical-closed 667) 城门 (｜) - 关闭

(defconst gate-vertical-open 669) 城门 (｜) - 开启

## 放置模式

(defconst place-normal 0) 常规 （与动作build相同）

(defconst place-forward 1) 前置 （与动作build-forward相同，不过如果位置因sn-placement-zone-size设为了很大的值而受影响，就可以用前置模式来环绕敌方玩家）

(defconst place-control 2) 选址控制系统 （需先运行up-set-placement-data。这里有个帖子讨论各个影响因素和生效机理：[place-control（选址控制系统）的影响参数的探讨交流](https://www.hawkaoe.net/bbs/thread-146375-1-1.html)）

(defconst place-point 3) 建造于点 （需先通过up-set-target-point设置目标点）

## 攻击姿态

(defconst stance-aggressive 0) 进攻

(defconst stance-defensive 1) 防御

(defconst stance-stand-ground 2) 坚守

(defconst stance-no-attack 3) 不反击

## 进度类型

注：经测试发现，或许这些未被收录——非建筑且非巨型投石机: -2，闲置: 0，攻击: 9（仅巨型投石机），组装: 124，拆装: 125。

(defconst progress-type-train 102) 训练 （DE中，似乎在检测其他玩家正训练单位的建筑时，进度类型恒为0）

(defconst progress-type-research 103) 研究

## 命令编号

注：这个单位属性对应于AGE里的“界面类型”属性。

(defconst cmdid-flag 0) 旗帜（没有命令编号的单位为0，比如尸体、抛射物、大部分装饰单位等）

(defconst cmdid-livestock-gaia 1) 资源物件（注意，其中会有少量单位并未含有资源）

(defconst cmdid-civilian-building 2) 经济建筑

(defconst cmdid-villager 3) 村民

(defconst cmdid-military 4) 军事单位

(defconst cmdid-trade 5) 贸易单位

(defconst cmdid-monk 6) 僧侣

(defconst cmdid-transport 7) 运输船

(defconst cmdid-relic 8) 圣物（包括携带圣物的僧侣）

(defconst cmdid-fishing-ship 9) 渔船

(defconst cmdid-military-building 10) 军事建筑

## 行动编号

（正在进行的动作）

(defconst actionid-attack 600) 攻击 包含正站定/正前往攻击、自动追击、自动寻敌攻击

(defconst actionid-defend 601) 警戒 若下跟随命令时不需要移动则为601，若跟随期间需要移动则在移动时为610、站定后为-1；因警戒而前往/攻击敌人时会转为600-攻击

(defconst actionid-build 602) 建造 村民、渔船建造建筑时

(defconst actionid-heal 603) 治疗 包含手动治疗、自动治疗

(defconst actionid-convert 604) 转化 包含手动转化、反击式转化

(defconst actionid-explore 605) 探索 未知。可能与AI的探索编队、DE的自动巡逻有关

(defconst actionid-stop 606) 停止 未知。一般按停止按钮 或 站定无动作时为-1

(defconst actionid-runaway 607) 逃跑 未知。国王被攻击时的逃离为-1，鹿被赶跑时为610，鹿被射击而逃离时为-1

(defconst actionid-retreat 608) 撤退 未知。已确定并不是收到up-retreat-\*指令后的撤退，也不是单位最小距离和ability-to-maintain-distance持距力导致的后退

(defconst actionid-gather 609) 采集 村民正前往/正采集 果树丛或金矿石矿或鱼群等无生命资源时为609，从前述动作转为移动后仍为613，即使站定了也仍为613，只有按停止按钮时才转为-1

(defconst actionid-move 610) 移动 除移动外，还包含无攻击目标的单独巡逻、需移动的警戒、需移动的跟随

(defconst actionid-patrol 611) 巡逻 未完全探明。人类玩家单位单独巡逻但无目标时为610，AI单位单独巡逻但无目标时为611；单独/集队巡逻且找到目标前往/正在攻击时为600，集队巡逻但无目标时为-1。

(defconst actionid-follow 612) 跟随 若下跟随命令时不需要移动则为612，若跟随期间需要移动则在移动时为610、站定后为-1

(defconst actionid-hunt 613) 捕猎 村民正前往/正猎杀/正采集 野生或蓄养动物或树木时为613，从前述动作转为移动后仍为613，即使站定了也仍为613，只有按停止按钮时才转为-1。村民自动还击狼时为600，手动攻击狼时为613

(defconst actionid-transport 614) 运输 未知。运输船似乎没有运输命令

(defconst actionid-trade 615) 贸易 贸易车、商船前往贸易时为615，转为移动时不会变为610，停止后才变为-1

(defconst actionid-evade 616) 躲避 未知。因ability-to-dodge-missiles回避力躲抛射物时为610

(defconst actionid-enter 617) 进驻 正前往驻扎时为617，驻扎成功后为-1

(defconst actionid-repair 618) 维修 村民自维修转为行走时不会变为610，自维修(或维修转为行走再)转为站定时才变为-1

(defconst actionid-train 619) 训练 建筑训练单位时

(defconst actionid-research 620) 研究 未知。建筑研发科技时似乎为-1

(defconst actionid-unload 621) 卸载 运输船从第一次前往卸载起一直为621，刻意按停止按钮才变为-1

(defconst actionid-relic 631) 遗迹 僧侣前往拾取圣物时为631，拾取成功后为-1

## 执行编号

（正在执行的命令）

(defconst orderid-attack 700) 攻击 包含追击、正前往攻击。人类玩家自动寻敌时为-1，AI玩家自动寻敌、因最小射程、因ability-to-maintain-distance持距力而边走边打时为700

(defconst orderid-defend 701) 警戒 执行警戒命令期间的任何时候，都为701

(defconst orderid-build 702) 建造 村民、渔船建造建筑时

(defconst orderid-heal 703) 治疗 僧侣手动治疗时为703，自动治疗时为-1

(defconst orderid-convert 704) 转化 僧侣手动转化时为704，反击式转化时为-1

(defconst orderid-explore 705) 探索 未知

(defconst orderid-stop 706) 停止 未知。一般按停止按钮 或 站定无动作时为-1

(defconst orderid-runaway 707) 逃跑 未知。国王被攻击时的逃离为-1，鹿被赶跑时为710，鹿被射击而逃离时为-1

(defconst orderid-retreat 708) 撤退 未知。已确定并不是收到up-retreat-\*指令后的撤退，也不是单位最小距离和ability-to-maintain-distance持距力导致的后退

(defconst orderid-gather 709) 采集 村民正前往/正采集 果树丛或金矿石矿或鱼群等无生命资源时为709，从前述动作转为移动后仍为713，即使站定了也仍为713，只有按停止按钮时才转为-1

(defconst orderid-move 710) 移动 执行移动命令期间的任何时候，都为710

(defconst orderid-patrol 711) 巡逻 单独巡逻但无目标时为为711，集队巡逻但无目标时为-1，单独/集队巡逻时发现并前往/正在攻击敌人时为-1，单独巡逻时杀死当前敌人后短暂的呆滞时间内为-1。

(defconst orderid-follow 712) 跟随 执行跟随命令期间的任何时候，都为712

(defconst orderid-hunt 713) 捕猎 村民正前往/正猎杀/正采集 野生或蓄养动物或树木时为713，从前述动作转为移动后仍713，即使站定了也仍为713，只有按停止按钮时才转为-1。村民自动还击狼时为-1，手动攻击狼时为713

(defconst orderid-transport 714) 运输 未知。运输船似乎没有运输命令

(defconst orderid-trade 715) 贸易 贸易车、商船前往贸易时为715，转为移动时不会变为710，停止后才变为-1

(defconst orderid-evade 716) 躲避 未知。因ability-to-dodge-missiles回避力躲抛射物时为710

(defconst orderid-enter 717) 进驻 正前往驻扎时为717，驻扎成功后为-1

(defconst orderid-repair 718) 维修 村民自维修转为移动时不会变为710，停止后才变为-1

(defconst orderid-train 719) 训练 建筑训练单位时

(defconst orderid-research 720) 研究 未知。建筑研发科技时似乎为-1

(defconst orderid-unload 721) 卸载 运输船从第一次前往卸载起一直为721，刻意按停止按钮才变为-1

(defconst orderid-relic 731) 遗迹 僧侣前往拾取圣物时为731，拾取成功后为-1

## 指派行动

注：若不特别注明，则该命令都适用于up-target-point和up-target-objects。“target-point -> action-move”的含义为“使用up-target-point时相当于action-move”，以此类推。

(defconst action-default 0) 默认 相当于框选单位右键点击目标点地面或目标单位。target-point -> action-move

(defconst action-move 1) 移动 命令单位前往目标点或目标单位所在点

(defconst action-patrol 2) 巡逻 命令单位向目标点或目标单位所在点巡逻。注意，使用精确模式向精确点巡逻时游戏可能会崩溃。

(defconst action-guard 3) 警戒 target-objects -> 命令单位护卫目标单位；target-point -> action-move

(defconst action-follow 4) 跟随 target-objects -> 命令单位跟随目标单位；target-point -> action-move

(defconst action-stop 5) 停止 命令单位停止，相当于按下单位“停止”按钮

(defconst action-ground 6) 强制攻击 命令单位攻击目标点或目标单位处的地面。对于无强制攻击能力的单位相当于action-none

(defconst action-garrison 7) 进驻 target-objects -> 命令单位进驻到目标单位里；如果目标单位并不能进驻，则大部分时候相当于action-stop。target-point -> action-move

(defconst action-delete 8) 自杀 命令能自杀的单位自杀(Del)

(defconst action-unload 9) 卸载 target-point -> 命令建筑释放驻扎的单位到目标点（但实测只相当于按下释放按钮，想要集结到某点，还得先通过action-gather来设置集结点），攻城武器(种属913)则将运送到目标点附近合适的地方卸载。target-objects -> action-none

(defconst action-train 10) 训练 target-point -> 命令建筑训练单位，并且必须改用这个语法：[up-target-point (in)(goal)储备资源状态 action-train 类型操作符 (in)(op)单位编号]，例如(up-target-point 0 action-train c: spearman-line)。target-objects -> action-none

(defconst action-gather 11) 集结 命令建筑将集结点设为目标点或目标单位所在点

(defconst action-lock 12) 锁定 target-point -> 命令城门切换锁定状态，已解锁的城门会锁定，已锁定的城门会解锁。target-objects -> action-none

(defconst action-work 13) 返工 target-point -> 命令建筑释放驻扎的村民返回工作。target-objects -> action-none

(defconst action-ungarrison 14) 取消驻扎 target-point -> 命令单位离开所驻扎建筑。target-objects -> action-none

(defconst action-drop-relic 15) 放下圣物 target-point -> 命令僧侣放下圣物。target-objects -> action-none

(defconst action-pack 16) 组装 target-point -> 命令抛石机组装。target-objects -> action-none

(defconst action-unpack 17) 展开 target-point -> 命令抛石机展开。target-objects -> action-none

(defconst action-none 18) 不作为 被命令单位的当前行动不会变更

## 抛射物类型

(defconst projectile-town-center 0) 城镇中心

(defconst projectile-castle 1) 城堡

(defconst projectile-watch-tower 2) 箭塔

(defconst projectile-bombard-tower 3) 炮塔

(defconst projectile-ship 4) 战船

(defconst projectile-siege 5) 攻城武器（弩炮除外）

(defconst projectile-fortification 6) 防御工事（城堡和塔）

(defconst projectile-any 7) 任意

## 事实编号

注：以下大部分事实编号都与征服者原版的事实语句对应，有些事实可以使用参数，我会用[0或 参数]来注明。有些事实编号只能在检查自己（up-get-fact或使用my-player-number的up-get-player-fact）时使用，有些只能对自己和盟友使用，有些则可以对所有人使用，我会用[self或any-ally或any]来注明。

**定义** **说明** **参数** **对象限制**

(defconst game-time 0) 游戏时间，即屏幕顶部的计时器。（秒） [0] [self]

(defconst population-cap 1) 游戏最大人口上限。（比如200人口的游戏中这个 [0] [self]  
值就是200。哥特帝王+10点32号资源也会添加  
到这个值上，触发也可以修改玩家的32号资源值）

(defconst population-headroom 2) 剩余最大人口上限。（当前人口上限离最大人口 [0] [self]  
上限的差值。即通过盖房子最多还能加多少人口  
上限。计算方式是：游戏最大人口上限+32号  
- 4号- 11号资源）

(defconst housing-headroom 3) 人口余量。（4号资源） [0] [self]

(defconst idle-farm-count 4) 闲置农田数 [0] [self]

(defconst food-amount 5) 食物数量 [0] [self]

(defconst wood-amount 6) 木材数量 [0] [self]

(defconst stone-amount 7) 石料数量 [0] [self]

(defconst gold-amount 8) 黄金数量 [0] [self]

(defconst escrow-amount 9) 储备资源数。（食物、木材、黄金、石料） [资源] [self]

(defconst commodity-buying-price 10) 商品买入价格。（食物、木材、石料） [资源] [self]

(defconst commodity-selling-price 11) 商品卖出价格。（食物、木材、石料） [资源] [self]

(defconst dropsite-min-distance 12) 资源放置点最小距离 [资源] [self]

(defconst soldier-count 13) 陆上部队数。（部队即军事单位） [0] [self]

(defconst attack-soldier-count 14) 陆上攻击部队数。（仅计算在Group- ATK、 [0] [self]  
ATK攻击部队里的单位）

(defconst defend-soldier-count 15) 陆上防御部队数。（仅计算不在Group-ATK、 [0] [self] ATK攻击部队里的单位）

(defconst warboat-count 16) 水上部队数。（部队即军事单位） [0] [self]

(defconst attack-warboat-count 17) 水上攻击部队数。（同上） [0] [self]

(defconst defend-warboat-count 18) 水上防御部队数。（同上） [0] [self]

(defconst current-age 19) 当前时代 [0] [any]

(defconst current-score 20) 当前分数 [0] [any]

(defconst civilization 21) 民族 [0] [any]

(defconst player-number 22) 玩家编号 [0] [any]

(defconst player-in-game 23) 玩家正在游戏。（该玩家是否存在且仍在游戏） [0] [any]

(defconst unit-count 24) 累计拥有过的单位总数。（只有见过的单位才会 [0] [any]  
被统计，训练中的不算。每个地图ID对应的  
单位只会被记1次，且死亡后计数也不会减1。  
故只会累加不会减少）

(defconst unit-type-count 25) 当前某种单位数。（不包括训练中的单位。检测 [单位编号] [any]  
敌人时，单位需要在视野内(?)；检测盟友时则可  
无视战雾，强制统计）

(defconst unit-type-count-total 26) 当前某种单位总数。（总数包括已存在的+训练中 [单位编号] [self]  
的。只能检测自己）

(defconst building-count 27) 当前建筑总数 [0] [any]

(defconst building-type-count 28) 当前某种建筑数 [建筑编号] [any]

(defconst building-type-count-total 29) 当前某种建筑总数。（还包括正在建造的地基） [建筑编号] [self]

(defconst population 30) 已占用人口。（11号资源） [0] [any]

(defconst military-population 31) 军事人口。（还包括僧侣和运输船） [0] [any]

(defconst civilian-population 32) 经济人口。（还包括渔船和贸易单位） [0] [any]

(defconst random-number 33) 随机数。（通过generate-random-number生成） [0] [self]

(defconst resource-amount 34) 资源数量。（对自己和盟友几乎任何资源都可检 [资源编号] [any-ally]  
测，对敌方玩家则任意资源都无法检测）

(defconst player-distance 35) 玩家间距。（自己TC到该玩家最近建筑的距离） [0] [any]

(defconst allied-goal 36) 同盟目标值 [目标编号] [any-ally]

(defconst allied-sn 37) 同盟策略值值 [策略值编号] [any-ally]

(defconst resource-percent 38) 资源数量百分数。（保留2位小数，再×100） [资源编号] [any]

(defconst enemy-buildings-in-town 39) 城镇中敌人建筑数 [0] [self]

(defconst enemy-units-in-town 40) 城镇中敌人单位数 [0] [self]

(defconst enemy-villagers-in-town 41) 城镇中敌人村民数 [0] [self]

(defconst players-in-game 42) 符合外交状态的玩家数。（AI的同盟包括自己） [玩家外交状态] [self]

(defconst defender-count 43) 防御部队数。（即负责清理城镇内部威胁的部队） [0] [self]

(defconst building-type-in-town 44) 城镇中某种敌方建筑数 [建筑编号] [self]

(defconst unit-type-in-town 45) 城镇中某种敌方单位数 [单位编号] [self]

(defconst villager-type-in-town 46) 城镇中某种敌方村民数 [单位编号] [self]

(defconst gaia-type-count 47) 某种自然单位数。（当前可视的(?)） [资源/单位编号] [self]

(defconst gaia-type-count-total 48) 某种自然单位总数。（累计探索到的(?)） [资源/单位编号] [self]

(defconst cc-gaia-type-count 49) 全图某种自然单位总数。（无需探索） [资源/单位编号] [self]

(defconst current-age-time 50) 当前时代停留时间 [0] [any]

(defconst timer-status 51) 定时器状态 [定时器编号] [self]

(defconst players-tribute 52) 收到给定玩家的贡品数。（整局游戏累计） [资源] [any]

(defconst players-tribute-memory 53) 收到给定玩家贡品的记忆。（从上次清除记忆起） [资源] [any]

## 搜索排序

(defconst search-order-none 0) 原顺序 （通常在删除重复项时使用，但处理速度可能会稍慢）

(defconst search-order-asc 1) 升序 （从小到大）

(defconst search-order-desc 2) 降序 （从大到小）

## 搜索源

(defconst search-local 1) 本地 （使用up-find-local命令搜索后储存搜索结果的列表。在其中的单位可作为up-target-\*的操控对象，或作它用。本地搜索只能搜索AI玩家自己。本地列表最大可容纳240个物件搜索结果）

(defconst search-remote 2) 远程 （使用up-find-remote一类命令搜索后储存搜索结果的列表。在其中的单位可作为up-target-objects的目标对象，或作它用。远程搜索的焦点玩家由sn-focus-player-number指定，而sn-focus-player-number可以是自己，也可以是其他玩家(1~8)，甚至可以是盖亚(0)。远程列表最大可容纳40个物件搜索结果）

## 玩家外交状态

注：以下常数未在UserPatchConst.per里定义，但已在帝国原系统里定义。

;(defconst ally 0) 同盟

;(defconst neutral 1) 中立

;(defconst any 2) 任意

;(defconst enemy 3) 敌对

## 位于主大陆

注：每片陆地/水域都分配有不同的区域编号，这可以用于判断物件是否处于不同的陆地/水域。使用这个参数时会将该单位的地图区域编号与第一片地图区域的编号(#0)对比（或许是跟游戏开始时位于的地图区域编号做对比，以判断是否“位于本土”？）。

(defconst off-mainland 0) 否

(defconst on-mainland 1) 是 （主大陆是指地图上生成的第一片区域）

## 微调指派点

注：这是一组未在动作和事实里使用的常数。与sn-target-point-adjustment的7个值和object-data-tile-position的5个值对应。

(defconst adjust-default 0) 默认

(defconst adjust-left 1) 左

(defconst adjust-top 2) 顶

(defconst adjust-right 3) 右

(defconst adjust-bottom 4) 底

(defconst adjust-middle 5) 中

(defconst adjust-precise 6) 精确

## 位置类型

(defconst position-center 0) 地图中心

(defconst position-opposite 1) 对面 （以地图中心为对称中心，position-self的中心对称点）

(defconst position-corner 2) 角落 （离position-self最近的地图角落）

(defconst position-enemy 3) 敌人 （目标敌人最近的非墙建筑±8格内的随机位置。未发现敌人时相当于position-opposite）

(defconst position-border 4) 边缘 （离position-self最近的地图边缘）

(defconst position-mirror 5) 镜像 （以分隔两队的线为对称轴，position-self的镜像点）

(defconst position-flank 6) 侧翼 （敌方队伍的侧翼玩家大概应处的位置）

(defconst position-zero 7) 地图最左 （0, 0）

(defconst position-map-size 8) 地图最右 （等于整个地图尺寸）

(defconst position-self 9) 自己 （最早的城镇中心处）

(defconst position-target 10) 目标玩家 （目标玩家最近的非墙建筑处。目标玩家与my-player-number相同的情况未被定义）

(defconst position-focus 11) 焦点玩家 （焦点玩家最近的非墙建筑处。焦点玩家与my-player-number相同的情况未被定义）

(defconst position-object 12) 目标物件 （target-object）

(defconst position-point 13) 目标点 （target-point）

## 侦查方法

(defconst scout-center 0) 地图中心 （侦查地图中心周围）

(defconst scout-opposite 1) 对面 （以地图中心为对称中心，position-self的中心对称点）

(defconst scout-corner 2) 角落 （离position-self最近的地图角落）

(defconst scout-enemy 3) 敌人 （会让侦查兵回到目标敌人最近的建筑的随机方向±8格内，以更好地探索目标敌人的城镇。如果你的盟友发现了敌人的城镇，你可以把它作为攻击目标，但不能用于前置建筑，因为你并不能建在黑色区域里。使用scout-enemy，你可以发现由盟友找到的建筑其旁边的地形）

(defconst scout-border 4) 边缘 （离position-self最近的地图边缘）

(defconst scout-mirror 5) 镜像 （以分隔两队的线为对称轴，position-self的镜像点）

(defconst scout-flank 6) 侧翼 （敌方队伍的侧翼玩家（边家）大概应处的位置）

## 物件清单

(defconst list-active 0) 活动 （包括大部分物件，尤其是活着的物件，也包括地基。活着的野生动物/牲畜、活着/砍倒的树木通过两种清单都能找到）

(defconst list-inactive 1) 非活动 （包括为了性能最大化而被从活动列表里移除的物件。比如鱼群、携带资源的已死动物、被杀死而正在倒下的单位）

## 物件状态

注：极少数情况如用于闪光时会出现7和8，作用未知。默认或up-reset-filter后为2。

(defconst status-pending 0) 建造中

(defconst status-ready 2) 正常 （绝大部分活的单位。如建筑、部队、活着的野生动物/牲畜、树木）

(defconst status-resource 3) 资源块 （砍倒的树木、果树丛、金矿、石矿、闪光、树桩、一堆石料）

(defconst status-down 4) 倒下 （被杀死而正在倒下，但不包括砍倒的树木）

(defconst status-gather 5) 可采集 （携带食物的已死动物、各种尸体、一堆食物/木材/黄金。其中尸体无法通过搜索找到）

## 物件编号

注：“物件”包括了单位和建筑，有些语句可支持使用单位/建筑编号、单位系和种属编号。这里只列举了一部分单位编号和种属编号，更多物件编号可以查找AGE，然后在你的AI或UserPatchConst.per里添加。标注[\*]号的表示未包含在UserPatchConst.per里，需要自行defconst。

(defconst flare 274) 闪光信号

(defconst town-center-foundation 621) 城镇中心地基

……（更多物件编号，请自行查AGE）

(defconst all-units-class -1) 所有种属

(defconst archery-class 900) 弓箭手种属

(defconst artifact-class 901) 文物种属 [\*]

(defconst trade-cog-class 902) 商船种属 [\*]

(defconst building-class 903) 建筑种属

(defconst villager-class 904) 村民种属

(defconst ocean-fish-class 905) 海洋鱼种属

(defconst infantry-class 906) 步兵种属

(defconst forage-class 907) 浆果丛种属 [\*]

(defconst stone-mine-class 908) 石矿种属 [\*]

(defconst prey-animal-class 909) 被动型动物种属 [\*]

(defconst predator-animal-class 910) 攻击型动物种属 [\*]

(defconst miscellaneous-class 911) 杂类种属 [\*]

(defconst cavalry-class 912) 骑兵种属

(defconst siege-weapon-class 913) 攻城武器种属

(defconst terrain-class 914) 地形物种属 [\*]

(defconst tree-class 915) 树木种属 [\*]

(defconst monastery-class 918) 修道院种属

(defconst trade-cart-class 919) 贸易车种属 [\*]

(defconst transport-ship-class 920) 运输船种属 [\*]

(defconst fishing-ship-class 921) 渔船种属 [\*]

(defconst warship-class 922) 战船种属

(defconst cavalry-cannon-class 923) 征服者种属

(defconst wall-class 927) 城墙种属

(defconst flare-class 930) 旗帜种属 [\*]

(defconst gold-mine-class 932) 金矿种属 [\*]

(defconst shore-fish-class 933) 沿海鱼种属

(defconst petard-class 935) 破坏者种属

(defconst cavalry-archer-class 936) 骑射手种属

(defconst gate-class 939) 城门种属

(defconst relic-class 942) 圣物种属 [\*]

(defconst monk-with-relic-class 943) 带着圣物的僧侣种属 [\*]

(defconst archery-cannon-class 944) 火枪手种属种属

(defconst scout-cavalry-class 947) 斥候骑兵种属 [\*]

(defconst farm-class 949) 农田种属

(defconst packed-trebuchet-class 951) 打包的攻城武器种属

(defconst tower-class 952) 塔种属

(defconst unpacked-trebuchet-class 954) 展开的攻城武器种属

(defconst scorpion-class 955) 弩炮种属

(defconst livestock-class 958) 牲畜种属

(defconst king-class 959) 国王种属

## 物件数据

(defconst object-data-index -1) 索引 （仅用于up-remove-objects，搜索结果中的每个单位都有一个索引编号，第一个为0，往后递增）

(defconst object-data-id 0) 地图ID （地图上第一个放置的单位为0，往后依次累加）

(defconst object-data-type 1) 单位编号 （比如4-步弓手，83-男村民。可在AGE中查阅）

(defconst object-data-class 2) 种属 （900~959。可看“物件编号”常数部分，或在AGE中查阅）

(defconst object-data-category 3) 类型 （一般为70-可作战 和80-建筑，可在AGE中查阅）

(defconst object-data-cmdid 4) 命令编号

(defconst object-data-action 5) 行动编号

(defconst object-data-order 6) 执行编号

(defconst object-data-target 7) 目标的种属 （行动的目标单位的种属，如攻击目标、跟随目标。种属编号900~959，请参阅 物件编号）

(defconst object-data-point-x 8) X坐标 （注：UP-AI体系里的X坐标为↗方向，y坐标为↘方向，与AOKTS相反）

(defconst object-data-point-y 9) Y坐标

(defconst object-data-hitpoints 10) 当前生命值

(defconst object-data-maxhp 11) 最大生命值

(defconst object-data-range 12) 射程

(defconst object-data-speed 13) 精确速度 （每秒移动1格=100点精确速度=AGE里的1点速度）

(defconst object-data-dropsite 14) 资源放置点 （来自[airef.github.io](https://airef.github.io/parameters/parameters-details.html" \l "ObjectData)：The UnitId of the dropsite that belongs to the resource that is being gathered by the target-object. This will always be the UnitId of a mill, lumber camp, and mining camp for food, wood, and gold/stone, even if resources are being dropped off at a town center. A villager who hasn't been assigned a role (name change) will also given the result "mill". All villagers given a non-gathering task, and all other objects will return -1 as a result.，如猎人的为68-磨坊）

(defconst object-data-resource 15) 携带资源类型 （适用于基础资源物件、村民、贸易车、僧侣、闪光、尸体，可以为0~3四种基础资源、信仰值、尸体腐烂计时器。若单位无法携带资源则为-1）

(defconst object-data-carry 16) 携带资源数量 （适用于基础资源物件、村民、贸易车、僧侣(资源#34-信仰值)、闪光&尸体(资源#12-腐烂计时器)，其他单位该值恒为0）

(defconst object-data-garrisoned 17) 已进驻

(defconst object-data-garrison-count 18) 驻扎单位数

(defconst object-data-status 19) 物件状态

(defconst object-data-player 20) 所属玩家 （0~8，0表示盖亚）

(defconst object-data-attack-stance 21) 攻击姿态

(defconst object-data-action-time 22) 动作时长 （大概是：若每3游戏秒内行动编号不变，则累积1，动作编号转为-1时该值重置为0。某些行动如巡逻和移动表现比较奇怪，未能探明规律）

(defconst object-data-target-id 23) 目标地图ID （行动的目标单位的地图ID，如攻击目标、跟随目标）

(defconst object-data-formation-id 24) 阵型编号 （似乎并不能检测4种阵型。需框选2个及以上单位一同行走才可有效检测，单独行走时为-1。每次站定(攻击或停下)暂时变为-1，每次恢复行走累积1，且一直附带于该单位上，不会归零。可以理解为“第几次下达移动指令”。最新消息称，这个数值是游戏内部对单位编组的编号，每次组织行动都会生成新的ID，或许与attack-group有所关联，且或许可以用于判断几个单位是否是同行，或是否在任何编组中）

(defconst object-data-patrolling 25) 巡逻中 （单独巡逻时为1。集队巡逻时为0。不巡逻时为0。巡逻期间攻击敌人时为0）

(defconst object-data-strike-armor 26) 近战防御

(defconst object-data-pierce-armor 27) 远程防御

(defconst object-data-base-attack 28) 基础攻击 （即当前该单位面板上的攻击力值，不包含隐藏的攻击加成。）

(defconst object-data-locked 29) 已锁定 （城门一类）

(defconst object-data-garrison-id 30) 驻扎载体编号 （该单位所驻扎进的载体(如冲车、建筑等)的地图ID）

(defconst object-data-train-count 31) 训练队列数量 （建筑的训练队列中的单位数量）

(defconst object-data-tasks-count 32) 被指定数 （①UP1.5：被敌人右键或自动寻敌前来攻击时（指行进途中），每个敌人记为2；正被敌人攻击（指播放攻击图像）时，每个敌人记为1；有抛射物指向自己射来时，每个抛射物记为1；被跟随或被警戒时，每个随从单位记为1；自己用抛射物射击敌人时，每个抛射物记为1。  
②DE：被敌人右键攻击时，每个敌人记为2；被敌人自动寻敌攻击时，每个敌人记为1；有抛射物指向自己射来时，每个抛射物记为1；被单个单个地跟随或警戒时，每个随从单位记为1，如果是框选2个以上一同跟随/警戒该单位，则会记为0；自己用抛射物射击敌人时，每个抛射物记为1）

(defconst object-data-attacker-count 33) 攻击者数量 （原文：not increased for certain situations like hunting。而事实上被野猪攻击时也可检测到）

(defconst object-data-attacker-id 34) 攻击者地图ID （只获取最早的攻击者，没有攻击者时为-2。原文：not set for certain situations like hunting。而事实上被野猪攻击时也可检测到）

(defconst object-data-under-attack 35) 正被攻击 （原文：not set for certain situations like hunting。而事实上被野猪攻击时也可检测到）

(defconst object-data-attack-timer 36) 受击计时器 （建筑每次遭受攻击时重置为60秒，重新倒计时(仅对建筑有效)）

(defconst object-data-point-z 37) 海拔高度 （对于非飞行单位，0=高地1，1=高地2，以此类推）

(defconst object-data-precise-x 38) 精确X坐标 （实际坐标保留2位小数，再×100。下同）

(defconst object-data-precise-y 39) 精确Y坐标 （同上）

(defconst object-data-precise-z 40) 精确海拔高度 （同上）

(defconst object-data-researching 41) 研究中

(defconst object-data-tile-position 42) 格内方位 （1左 2上 3右 4下 5中，与sn-target-point-adjustment的值对应）

(defconst object-data-tile-inverse 43) 格内相反方位 （1⇌3，2⇌4，5⇌5。比如格内方位为1，则相反方位为3）

(defconst object-data-distance 44) 距离 （自身精确坐标到普通目标点(target-point)之间的直线距离，保留至整数。比如距离为41.28，则该项为41）

(defconst object-data-precise-distance 45) 精确距离 （自身精确坐标到普通目标点(target-point)之间的直线距离，保留2位小数，再×100。比如距离为41.28，则该项为4128）

(defconst object-data-full-distance 46) 精确距离平方 （自身精确坐标到普通目标点(target-point)之间的直线距离，保留两位小数，平方后再保留至整数。比如距离为41.28，则该项为1704）

(defconst object-data-map-zone-id 47) 地图区域编号 （每片陆地/水域都分配有不同的区域编号，这可以用于判断物件是否处于不同的陆地/水域）

(defconst object-data-on-mainland 48) 位于主大陆

(defconst object-data-idling 49) 闲置中 （闲置时为1，否则为0。集队巡逻中的单位为0）

(defconst object-data-move-x 50) 移动目的地X坐标 （对移动、攻击、跟随等行动有效，攻击、跟随时目的地即为目标单位的坐标。自动索敌时也有效，但持续追击期间该值不会变更。UP1.5中单独巡逻移动时只能获取到0，DE中单独巡逻时可获取到最后一个巡逻目的地(巡逻期间恒定，不会因折返而更新)，DE中集队巡逻时只能获取到-1）

(defconst object-data-move-y 51) 移动目的地Y坐标 （同上）

(defconst object-data-precise-move-x 52) 移动目的地精确X坐标 （实际坐标保留2位小数，再×100。下同）

(defconst object-data-precise-move-y 53) 移动目的地精确Y坐标 （同上）

(defconst object-data-reload-time 54) 攻击间隔 （单位：ms。未探明是否会受加射速的科技影响）

(defconst object-data-next-attack 55) 下次攻击时间 （单位：ms。首个攻击图像帧播放时重置为攻击间隔，军事单位杀死动物时会令作为事实的(up-get-object-data object-data-next-attack <goal>）失效）

(defconst object-data-train-site 56) 训练场所 （可训练该物件的建筑(如民兵→兵营)或单位(如建筑→建筑工)）

(defconst object-data-train-time 57) 训练时间 （单位：s。对于一个单位而言，这是一个恒定值，不受民族加成或科技影响）

(defconst object-data-blast-radius 58) 溅射半径 （半径×100.0）

(defconst object-data-blast-level 59) 攻击伤害等级 （溅射的模式。等级为0可以溅射到任何单位，1可以溅射到树木，2可以溅射伤害周围单位，3+没有溅射效果。单位只能伤害防御等级≥该值的单位。不过有很多没有溅射半径的单位其攻击伤害等级为0，故判断单位是否有溅射能力时使用>0更稳妥）

(defconst object-data-progress-type 60) 进度类型 （类型80(建筑和巨型投石机)执行训练、研究、组装拆装时有效，没有执行任何操作时为0。无法执行上述操作的单位恒为-2）

(defconst object-data-progress-value 61) 进度百分比 （类型80(建筑和巨型投石机)执行训练、研究、组装拆装时有效，没有执行任何操作时为0。无法执行上述操作的单位恒为-2）

(defconst object-data-min-range 62) 最小射程

(defconst object-data-target-time 63) 专注时长 （单位：ms，仅在专注于攻击目标时有效。站立 即空闲时为-2，有目标但不是去攻击时移动为0 (如跟随和警戒)，没有目标时移动也为0，有攻击目标包括前往攻击期间持续累加，一旦丢失或切换攻击目标则重置为0。对工作中的村民/贸易单位/治疗中的僧侣无效，牧羊(赶羊)时的村民专注时长为0到-1000的负值）

(defconst object-data-heresy 64) 异端邪说 （单位是否正受益于该科技。亦即192号资源值-异端邪说）

(defconst object-data-faith 65) 信仰 （单位是否正受益于该科技。亦即77号资源值-转化抗性，研究信念后该值为3）

(defconst object-data-redemption 66) 救赎 （单位是否正受益于该科技。亦即28号资源值-可转化建筑）

(defconst object-data-atonement 67) 赎罪 （单位是否正受益于该科技。亦即27号资源值-可转化僧侣）

(defconst object-data-theocracy 68) 神权 （单位是否正受益于该科技。亦即193号资源值-神权统治）

(defconst object-data-spies 69) 间谍 （单位是否正受益于该科技。亦即183号资源值-间谍）

(defconst object-data-ballistics 70) 弹道学 （检查该单位的主抛射物是否拥有弹道学效果，若没有抛射物则为-2）

(defconst object-data-gather-type 71) 采集类型 （该资源物件携带的资源，或该职业村民所对应的可采集的资源，对载有黄金的贸易单位也有效，未探明对市场、僧侣、尸体是否有效。食物0 木材1 石料2 黄金3 或-1表示无）

(defconst object-data-language-id 72) 语言编号 （由语言文件定义，每条字符串对应一个数字ID。可以用来检查村民的职业：村民语言编号）

(defconst object-data-group-flag 73) 编队编号 （对人类玩家的单位，为由ctrl+1~9设置的编队；对AI玩家的单位，为通过up-modify-group-flag分配的编队编号。可能的值有0到9，或-2表示无编队。UP1.5中可以获取到其他玩家单位的编队编号，DE中则只能获取AI玩家自己单位的编队编号）

(defconst object-data-hero-flags 74) 英雄模式 （40号科技属性，即二进制属性 - 英雄状态标志之和。标志1=完整的英雄，标志2=免疫转化，标志4=自愈，标志8=默认防御姿态，标志16=阵型保护对象，标志32=自杀式攻击，标志64=波及伤害随距离衰减，标志128=禁止强制攻击，标志256=清除所有标志）

(defconst object-data-hero 75) 完整英雄标志 （一般英雄都为1，一般非英雄都为0。在场景中可被触发更改）

(defconst object-data-auto-heal 76) 自愈标志 （狂战士为0，因为狂战士自愈与英雄自愈是不同的）

(defconst object-data-no-convert 77) 免疫转化标志 （城镇中心和城堡为0，因为它们的免疫转化是硬编码）

(defconst object-data-frame-delay 78) 帧延迟 （抛射物射出之前图像经过的帧数）

(defconst object-data-attack-count 79) 持续攻击次数 （近战单位每次挥刀即将攻击到敌人时、或远程每发射出一次抛射物时，记为1次；主动移动或停止会维持当前值并在下次攻击时重置为0；追击移动则不会重置，可继续累加；改变目标或杀死当前目标会立即重置为0）

(defconst object-data-to-precise 80) 双精确距离 （自身精确坐标到精确目标点(precise-target-point)的直线距离。比如距离为41.53，则该值为4153）

(defconst object-data-base-type 81) 基础单位编号 （该单位所属的单位系里的第一个，比如即使在研究弩手后，弓兵系的基础单位也为4-步弓手）

(defconst object-data-upgrade-type 82) 升级单位编号 （该单位所属的单位系里的当前单位，比如研究弩手后，弓兵系的升级单位编号为24-弩手）

(defconst object-data-ownership 83) 捕获所有权 （DE新增。该可捕获物件(如纪念碑)的当前所属玩家ID或附近人数占优的玩家ID。如果是盖亚的，或当前没有玩家占领，则为-1。如果无法被捕获，则为-2）

(defconst object-data-capture-flag 84) 捕获标志 （DE新增。该物件的捕获类型。0为无法捕获，1为可捕获一次(盖亚单位,且绵羊除外)，2为可捕获多次，3为捕获牲畜或遗迹。如果不存在捕获类型，则为-2）

## 闲置单位类型

(defconst idle-type-villager 0) 闲置村民

(defconst idle-type-trade-cart 1) 闲置贸易车

(defconst idle-type-fishing-ship 2) 闲置渔船

(defconst idle-type-trade-cog 3) 闲置商船

## 寻找玩家方法

(defconst find-attacker 0) 寻找攻击者 （最近一个攻击自己的玩家。如果未受到攻击，会返回无效编号）

(defconst find-random 1) 寻找随机玩家 （随机性堪忧，慎用！使用up-find-next-player时，不会寻找先前已找到过的玩家(即给定的起始玩家编号)，但其他任何玩家都可被找到。）

(defconst find-closest 2) 寻找最接近者 （建筑离自己最近的玩家。如果给定外交的玩家未找到，会返回无效编号。使用up-find-next-player时，找到的将是下一个最近的玩家，到最远玩家后下一次寻找会循环回最近玩家）

(defconst find-ordered 3) 寻找编号 （使用up-find-player时，会寻找编号最大的玩家。使用up-find-next-player时，会按玩家编号升序排序，在匹配到的玩家中依次递增，即从1到8，到最大玩家编号后下一次寻找会循环回1）

## 研究状态

(defconst research-unavailable 0) 不可用 （前提时代和前提科技未达到，或者因科技树而受限）

(defconst research-available 1) 可用 （前提条件已全部达到，但玩家尚未开始研究）

(defconst research-pending 2) 正在研究

(defconst research-complete 3) 研究完成

## 已探索状态

(defconst explored-no 0) 未探索 （黑域中）

(defconst explored-yes 128) 已探索 （战雾中）

(defconst explored-active 15) 当前可视 （正被某个物件打亮）

## 优先级类型

(defconst priority-offense 0) 进攻性优先级

(defconst priority-defense 1) 防御性优先级

## 资源

注：以下常数未在UserPatchConst.per里定义而是以注释方式保存，但已在帝国原系统里定义。有些语句支持资源物件的种属编号，可以参阅“物件编号”；有些语句还支持物件编号如野猪-48。

;(defconst food 0) 食物

;(defconst wood 1) 木材

;(defconst stone 2) 石料

;(defconst gold 3) 黄金

;(defconst hunting 4) 正捕猎的野猪或鹿。只能用于dropsite-min-distance。获取失败时返回255而非-1

;(defconst boar-hunting 5) 正捕猎的活着/死亡的野猪。只能用于dropsite-min-distance。获取失败时返回255而非-1

;(defconst deer-hunting 6) 正捕猎的鹿。只能用于dropsite-min-distance。获取失败时返回255而非-1

;(defconst live-boar 7) 活着的野猪。只能用于dropsite-min-distance。获取失败时返回255而非-1

## 资源编号

注：标注(\*)号的表示未包含在UserPatchConst.per里，需要自行defconst。使用某些指令(如up-get-fact resource-amount <资源编号> gl-data)时，可以使用一些未列举的资源编号。

(defconst amount-food 0) 食物 （与food等效）

(defconst amount-wood 1) 木材 （与wood等效）

(defconst amount-stone 2) 石料 （与stone等效）

(defconst amount-gold 3) 黄金 （与gold等效）

(defconst amount-population-cap 4) 人口余量

(defconst amount-religion 5) <!无效>转化距离

(defconst amount-current-age 6) 当前时代 （0=黑暗时代，1=封建时代，以此类推）

(defconst amount-relics 7) 圣物数

(defconst amount-trade-bonus 8) <!无效>贸易奖励

(defconst amount-trade-goods 9) 贸易商品

(defconst amount-trade-production 10) <!无效>贸易生产比率

(defconst amount-population 11) 人口占用

(defconst amount-decay 12) 尸体腐烂时间

(defconst amount-discovery 13) <!无效>发现

(defconst amount-ruins 14) 纪念碑占领数

(defconst amount-meat 15) <!无效>动物食物

(defconst amount-berries 16) <!无效>浆果食物

(defconst amount-fish 17) <!无效>鱼类食物

(defconst amount-units-owned 19) 累计拥有过的单位数 (\*)

(defconst amount-kills 20) 杀敌数

(defconst amount-research-count 21) 研究科技数

(defconst amount-exploration 22) 地图探索百分比

(defconst amount-convert-priest 27) 「赎罪」效果

(defconst amount-convert-building 28) 「救赎」效果

(defconst amount-building-limit 30) <!无效>建筑上限

(defconst amount-food-limit 31) <!无效>食物上限

(defconst amount-unit-limit 32) 最大人口上限

(defconst amount-maintenance 33) <!无效>维护

(defconst amount-faith 34) 信仰值

(defconst amount-faith-recharge-rate 35) 信仰值回复速率

(defconst amount-farm-food 36) 农田食物产量

(defconst amount-civilian-population 37) 最高村民数

(defconst amount-all-techs-achieved 39) 科技总数

(defconst amount-military-population 40) 最大军队

(defconst amount-conversions 41) 转化数

(defconst amount-wonder 42) 现存奇观数

(defconst amount-razings 43) 摧毁建筑数

(defconst amount-kill-ratio 44) 杀敌比率

(defconst amount-player-killed 45) 最终幸存

(defconst amount-tribute-inefficiency 46) 贡品税率

(defconst amount-gold-bonus 47) 黄金资源利用率

(defconst amount-town-center-unavailable 48) 城镇中心不可用

(defconst amount-gold-counter 49) 黄金计数器

(defconst amount-writing 50) 「制图学」效果

(defconst amount-monasteries 52) 现存修道院数

(defconst amount-tribute 53) 送出贡品

(defconst amount-hold-ruins 54) 捕获所有纪念碑

(defconst amount-hold-relics 55) 捕获所有圣物

(defconst amount-ore 56) 矿石储存

(defconst amount-captured-unit 57) 挟持储存

(defconst amount-trade-good-quality 59) <!无效>贸易商品质量

(defconst amount-trade-market-level 60) <!无效>贸易市场等级

(defconst amount-formations 61) <!无效>阵型

(defconst amount-building-housing-rate 62) <!无效>房屋建造速率

(defconst amount-gather-tax-rate 63) <!无效>收集税率

(defconst amount-gather-accumulator 64) <!无效>收集累加器

(defconst amount-salvage-decay-rate 65) <!无效>野生动物腐烂比率

(defconst amount-allow-formations 66) <!无效>允许阵型

(defconst amount-can-convert 67) 允许转化

(defconst amount-player1-kills 69) 杀死P1单位

(defconst amount-player2-kills 70) 杀死P2单位

(defconst amount-player3-kills 71) 杀死P3单位

(defconst amount-player4-kills 72) 杀死P4单位

(defconst amount-player5-kills 73) 杀死P5单位

(defconst amount-player6-kills 74) 杀死P6单位

(defconst amount-player7-kills 75) 杀死P7单位

(defconst amount-player8-kills 76) 杀死P8单位

(defconst amount-convert-resistance 77) 转化抗性

(defconst amount-trade-vig-rate 78) 交易税率

(defconst amount-stone-bonus 79) 石矿资源利用率

(defconst amount-queued-count 80) 训练队列总单位数

(defconst amount-training-count 81) 训练中的单位数

(defconst amount-raider 82) 以PTWC开局

(defconst amount-boarding-recharge-rate 83) 登舰能力恢复速率

(defconst amount-starting-villagers 84) 起始村民数

(defconst amount-research-cost-mod 85) 研究成本倍率

(defconst amount-research-time-mod 86) 研究时间倍率

(defconst amount-convert-boats 87) 允许登舰

(defconst amount-fish-trap-food 88) 渔网食物产量

(defconst amount-heal-rate-modifier 89) 治疗速率倍率

(defconst amount-heal-range 90) 治疗距离

(defconst amount-starting-food 91) 奖励起始食物

(defconst amount-starting-wood 92) 奖励起始木材

(defconst amount-starting-stone 93) 奖励起始石料

(defconst amount-starting-gold 94) 奖励起始黄金

(defconst amount-raider-ability 95) 启用PTWC/挟持/掠夺

(defconst amount-berserker-heal-timer 96) 狂战士回复间隔

(defconst amount-dominant-sheep-control 97) 掳获动物优势

(defconst amount-object-cost-summation 98) 经济积分

(defconst amount-research-cost-summation 99) 科技积分

(defconst amount-relic-income-summation 100) 圣物收入总和

(defconst amount-trade-income-summation 101) 贸易收入总和

(defconst amount-player1-tribute 102) 进贡给P1数

(defconst amount-player2-tribute 103) 进贡给P2数

(defconst amount-player3-tribute 104) 进贡给P3数

(defconst amount-player4-tribute 105) 进贡给P4数

(defconst amount-player5-tribute 106) 进贡给P5数

(defconst amount-player6-tribute 107) 进贡给P6数

(defconst amount-player7-tribute 108) 进贡给P7数

(defconst amount-player8-tribute 109) 进贡给P8数

(defconst amount-player1-kill-value 110) 消灭P1单位积分

(defconst amount-player2-kill-value 111) 消灭P2单位积分

(defconst amount-player3-kill-value 112) 消灭P3单位积分

(defconst amount-player4-kill-value 113) 消灭P4单位积分

(defconst amount-player5-kill-value 114) 消灭P5单位积分

(defconst amount-player6-kill-value 115) 消灭P6单位积分

(defconst amount-player7-kill-value 116) 消灭P7单位积分

(defconst amount-player8-kill-value 117) 消灭P8单位积分

(defconst amount-player1-razings 118) 摧毁P1建筑

(defconst amount-player2-razings 119) 摧毁P2建筑

(defconst amount-player3-razings 120) 摧毁P3建筑

(defconst amount-player4-razings 121) 摧毁P4建筑

(defconst amount-player5-razings 122) 摧毁P5建筑

(defconst amount-player6-razings 123) 摧毁P6建筑

(defconst amount-player7-razings 124) 摧毁P7建筑

(defconst amount-player8-razings 125) 摧毁P8建筑

(defconst amount-player1-razing-value 126) 摧毁P1建筑积分

(defconst amount-player2-razing-value 127) 摧毁P2摧毁积分

(defconst amount-player3-razing-value 128) 摧毁P3摧毁积分

(defconst amount-player4-razing-value 129) 摧毁P4摧毁积分

(defconst amount-player5-razing-value 130) 摧毁P5摧毁积分

(defconst amount-player6-razing-value 131) 摧毁P6摧毁积分

(defconst amount-player7-razing-value 132) 摧毁P7摧毁积分

(defconst amount-player8-razing-value 133) 摧毁P8摧毁积分

(defconst amount-castle 134) 现存城堡数

(defconst amount-kills-by-player1 136) 被P1杀死单位

(defconst amount-kills-by-player2 137) 被P2杀死单位

(defconst amount-kills-by-player3 138) 被P3杀死单位

(defconst amount-kills-by-player4 139) 被P4杀死单位

(defconst amount-kills-by-player5 140) 被P5杀死单位

(defconst amount-kills-by-player6 141) 被P6杀死单位

(defconst amount-kills-by-player7 142) 被P7杀死单位

(defconst amount-kills-by-player8 143) 被P8杀死单位

(defconst amount-razings-by-player1 144) 被P1摧毁建筑

(defconst amount-razings-by-player2 145) 被P2摧毁建筑

(defconst amount-razings-by-player3 146) 被P3摧毁建筑

(defconst amount-razings-by-player4 147) 被P4摧毁建筑

(defconst amount-razings-by-player5 148) 被P5摧毁建筑

(defconst amount-razings-by-player6 149) 被P6摧毁建筑

(defconst amount-razings-by-player7 150) 被P7摧毁建筑

(defconst amount-razings-by-player8 151) 被P8摧毁建筑

(defconst amount-value-killed-by-others 152) 损失单位积分

(defconst amount-value-razed-by-others 153) 损失建筑积分

(defconst amount-killed-by-others 154) 损失单位

(defconst amount-razed-by-others 155) 损失建筑

(defconst amount-tribute-from-player1 156) 接收 P1贡品

(defconst amount-tribute-from-player2 157) 接收 P2贡品

(defconst amount-tribute-from-player3 158) 接收 P3贡品

(defconst amount-tribute-from-player4 159) 接收 P4贡品

(defconst amount-tribute-from-player5 160) 接收 P5贡品

(defconst amount-tribute-from-player6 161) 接收 P6贡品

(defconst amount-tribute-from-player7 162) 接收 P7贡品

(defconst amount-tribute-from-player8 163) 接收 P8贡品

(defconst amount-value-current-units 164) 现存单位积分

(defconst amount-value-current-buildings 165) 现存建筑积分

(defconst amount-food-total 166) 食物采集

(defconst amount-wood-total 167) 木材采集

(defconst amount-stone-total 168) 石料采集

(defconst amount-gold-total 169) 黄金采集

(defconst amount-total-value-of-kills 170) 消灭单位总分 （军事积分）

(defconst amount-total-tribute-received 171) 接收贡品总数

(defconst amount-total-value-of-razings 172) 摧毁建筑积分

(defconst amount-total-castles-built 173) 已建造城堡总数

(defconst amount-total-wonders-built 174) 已建造奇观总数

(defconst amount-tribute-score 175) 进贡积分

(defconst amount-convert-min-adj 176) 转化最小值调整值

(defconst amount-convert-max-adj 177) 转化最大值调整值

(defconst amount-convert-resist-min-adj 178) 转化抗性最小值调整值

(defconst amount-convert-resist-max-adj 179) 转化抗性最大值调整值

(defconst amount-convert-building-min 180) 最小转化建筑时间

(defconst amount-convert-building-max 181) 最大转化建筑时间

(defconst amount-convert-building-chance 182) 转化建筑概率

(defconst amount-spies 183) 「间谍」效果

(defconst amount-value-wonders-castles 184) 社会积分

(defconst amount-food-score 185) 食物分数

(defconst amount-wood-score 186) 木材分数

(defconst amount-stone-score 187) 石料分数

(defconst amount-gold-score 188) 黄金分数

(defconst amount-wood-bonus 189) 木材资源利用率

(defconst amount-food-bonus 190) 食物资源利用率

(defconst amount-relic-rate 191) 圣物黄金产生速率

(defconst amount-heresy 192) 「异端邪说」效果

(defconst amount-theocracy 193) 「神权政治」效果

(defconst amount-crenellations 194) 「垛口」效果

(defconst amount-construction-rate-mod 195) 建造效率倍率

(defconst amount-hun-wonder-bonus 196) 奇观额外年份\*10

(defconst amount-spies-discount 197) 「无神论」效果 （间谍费用折扣）

(defconst amount-unused-0 220) 未使用-0

(defconst amount-unused-1 221) 未使用-1

(defconst amount-unused-2 222) 未使用-2

(defconst amount-unused-3 223) 未使用-3

(defconst amount-unused-4 224) 未使用-4

注：以下3个资源未投入使用。

;(defconst amount-houses 51) 房屋计数

;(defconst amount-hit-points-killed 68) 杀敌生命值总和

;(defconst amount-hit-points-razings 135) 摧毁建筑生命值

## 阵型

(defconst formation-line 2) 横列

(defconst formation-box 4) 方阵

(defconst formation-stagger 7) 分散

(defconst formation-flank 8) 两翼

# 策略值常数

## 本部分格式说明

每个策略值的描述分为三个逻辑行。第一行是策略值名称；第二行是对该策略值功能的描述；第三行分为三个部分，分别是有效取值范围、默认值、内部编号。过时的策略值用红色字体标明。

## sn-add-starting-resource-food

作弊用。增加初始食物资源。

[有效范围] -32767~32768 [默认值] 0 [编号] 138

## sn-add-starting-resource-gold

作弊用。增加初始黄金资源。

[有效范围] -32767~32768 [默认值] 0 [编号] 139

## sn-add-starting-resource-stone

作弊用。增加初始石料资源。

[有效范围] -32767~32768 [默认值] 0 [编号] 140

## sn-add-starting-resource-wood

作弊用。增加初始木材资源。

[有效范围] -32767~32768 [默认值] 0 [编号] 141

## sn-allow-adjacent-dropsites

设为1，则建造磨坊、伐木场、矿场时与资源不留1格的缓冲区；设为0，则留出1格缓冲区。

[有效范围] 0~1 [默认值] 0 [编号] 272

## sn-allow-civilian-defense

设为0，禁止村民防御；设为1，村民会防御弱的、非远程单位；设为2，村民会防御除战船和速度快于村民的所有单位；设为3，村民会防御除战船外所有单位。

[有效范围] 0~3 [默认值] 1 [编号] 225

## sn-allow-civilian-offense

设为1，允许村民像士兵一样参与城镇范围进攻（TSA）；设为2，即使有防御部队，村民仍会进攻敌方村民和建筑；设为0，村民只会攻击没有研究垛口的敌人的前置塔。

[有效范围] 0~2 [默认值] 0 [编号] 258

## sn-allow-diplomacy-change-on-ally-attack

设为1，允许该电脑玩家被同盟进攻时改变对其外交状态；设为0则不允许。

[有效范围] 0~1 [默认值] 1 [编号] 215

## sn-allow-diplomacy-change-on-tribute

设为1，当该电脑玩家收到某玩家进贡后，允许改变对其外交状态。设为0则不允许。

[有效范围] 0~1 [默认值] 0 [编号] 217

## sn-attack-coordination

设置电脑玩家进攻时的协调方式。0表示无协调；1表示某一时间可以有一支队伍进攻；2表示某一时间可以有多支队伍同时进攻。

[有效范围] 0~2 [默认值] -1 [编号] 47

## sn-attack-diplomacy-impact

电脑玩家受到攻击时，向外交系统倾注的正面或负面影响度。

[有效范围] 0~100 [默认值] 10 [编号] 178

## sn-attack-group-gather-spacing

编组的单位被判定为集合完毕所需要的相对于集结点的密集度。亦即一队单位从分散状态重新聚集时，单位要距离队伍集结点多近，才会被系统认为集结完毕。

[有效范围] 1~255 [默认值] 4 [编号] 41

## sn-attack-group-makeup

[有效范围] -32767~32768 [默认值] 0 [编号] 37

## sn-attack-group-size-randomness

进攻编组大小的随机性。这个数字会为进攻编组的最小大小的随机性设定最大值。某个编组的随机性在创建时就确定了，范围在0到本数字之间。

[有效范围] 0~32768 [默认值] 1 [编号] 98

## sn-attack-intelligence

设定是否使用智能进攻系统。智能进攻系统尝试在攻击时躲避敌人单位，尝试从多角度进攻。配合sn-attack-coordination设为2，可以创造多线作战效果。1表示开启，0表示关闭。

[有效范围] 0~1 [默认值] 0 [编号] 103

## sn-attack-response-separation-time

设置一次敌人进攻产生的求救信号之后，单位作出反应前必须经过的时间。

[有效范围] 0~32768 [默认值] -1 [编号] 48

## sn-attack-separation-time

设置电脑玩家进攻间隔。

[有效范围] 0~32768 [默认值] -1 [编号] 46

## sn-attack-separation-time-randomness

设置电脑玩家进攻间隔时间的上下随机浮动范围。必须非负且小于sn-attack-separation-time。

[有效范围] 0~32768 [默认值] 15 [编号] 102

## sn-attack-winning-player

设置电脑玩家是否攻击领先玩家。

[有效范围] 0~1 [默认值] 1 [编号] 188

## sn-attack-winning-player-factor

若sn-attack-winning-player为1，本数字表示使用该判定的可能性。

[有效范围] -32767~32768 [默认值] 25 [编号] 195

## sn-auto-build-docks

（过时的）设置电脑玩家是否决定何时何地修建船坞。

[有效范围] 0~1 [默认值] -1 [编号] 208

## sn-auto-build-dropsites

（过时的）设置电脑玩家是否决定何时何地修建资源放置点。

[有效范围] 0~1 [默认值] -1 [编号] 205

## sn-auto-build-farms

（过时的）设置电脑玩家修建完毕建筑队列之后，若有剩余木材是否修建农田。

[有效范围] 0~1 [默认值] -1 [编号] 206

## sn-auto-build-fishing-boats

（过时的）设置电脑玩家是否决定何时何地修建渔船。

[有效范围] 0~1 [默认值] -1 [编号] 209

## sn-auto-build-houses

（过时的）设置电脑玩家能否自主决定修建住房。

[有效范围] 0~1 [默认值] -1 [编号] 180

## sn-auto-build-towers

（过时的）设置电脑玩家能否使用多余石料修建塔。

[有效范围] 0~1 [默认值] -1 [编号] 207

## sn-auto-build-transports

（过时的）设置电脑玩家是否决定何时何地修建运输船。

[有效范围] 0~1 [默认值] -1 [编号] 210

## sn-auto-build-warships

（过时的）设置电脑玩家是否决定何时何地修建战船。

[有效范围] 0~1 [默认值] -1 [编号] 223

## sn-blot-exploration-map

设置电脑玩家是否重新探索已探索区域。1表示重新探索，0禁止重新探索。

[有效范围] 0~1 [默认值] 1 [编号] 135

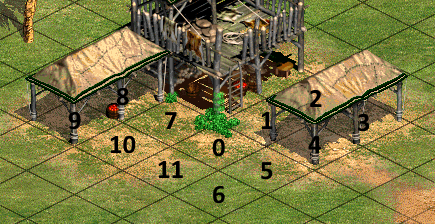
## sn-blot-size

设置电脑玩家标记为重新探索区域的大小。

[有效范围] 1~255 [默认值] 15 [编号] 136

## sn-boar-lure-destination

调整引猪目的地在城镇中心的相对位置。各个值对应的位置见下图。给普通值(0~11) 加12可以把点下移到格子角处。



[有效范围] 0~23 [默认值] 0 [编号] 295

## sn-build-frequency

设置两次训练或研究尝试间遍历战略AI更新的次数。

[有效范围] 0~32768 [默认值] 1 [编号] 101

## sn-build-plan-divisions

电脑玩家把建筑队列分割成的部分数量，用来计算集聚百分比。必须大于等于1。

[有效范围] 1~32768 [默认值] 10 [编号] 155

## sn-building-targeting-mode

设为0，允许以所有建筑为目标；设为1，忽略城墙、城门；设为2，忽略城墙、城门和资源放置点。请避免使用本策略值，因为在特定情况下本数字会失效。

[有效范围] 0~2 [默认值] 0 [编号] 255

## sn-camp-max-distance

设置伐木场和矿场距离城镇中心的最远距离。

[有效范围] 7~255 [默认值] 25 [编号] 86

## sn-cap-civilian-builders

设置分配为建筑工的最大数量。设为-1则不限。

[有效范围] -1~32768 [默认值] 2 [编号] 4

## sn-cap-civilian-explorers

设置分配为村民探索者的最大数量。设为-1则不限。

[有效范围] -1~32768 [默认值] 2 [编号] 3

## sn-cap-civilian-gatherers

设置分配为采集者的最大数量。设为-1则不限。

[有效范围] -1~32768 [默认值] -1 [编号] 5

## sn-choke-point-defend-priority

（过时的）设置阻塞点的防御优先级。

[有效范围] 0~1 [默认值] 0 [编号] 53

## sn-consecutive-idle-unit-limit

设置从一个编组的所有单位闲置到该编组被判定为闲置所经历的秒数。只在攻击和撤退阶段使用。

[有效范围] 0~32768 [默认值] 15 [编号] 76

## sn-coop-demand-tribute-interval

（过时的）设置电脑玩家两次向同盟请求贡品的最短间隔时间。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 186

## sn-coop-demand-tribute-maximum

（过时的）设置电脑玩家单次向同盟请求贡品的最大数量。必须大于等于1。

[有效范围] 1~32768 [默认值] 0 [编号] 187

## sn-coop-share-attacking

设置同盟电脑玩家是否可以为了互相防御而攻击。

[有效范围] 0~1 [默认值] 1 [编号] 196

## sn-coop-share-attacking-interval

设置电脑玩家两次向同盟求助的最短间隔时间。

[有效范围] 0~32768 [默认值] 120 [编号] 197

## sn-coop-share-information

设置同盟电脑玩家能否分享其探索信息。机制与绘图法科技不同，与人类玩家聊天类似。

[有效范围] 0~1 [默认值] 1 [编号] 194

## sn-defend-group-makeup

（过时的）

[有效范围] -32767~32768 [默认值] 0 [编号] 39

## sn-defend-important-group-leaders

（过时的）设置一个编组中其他单位是否保护重要的领导单位。

[有效范围] 0~1 [默认值] -1 [编号] 121

## sn-defend-overlap-distance

设置一个防御编组的影响力。游戏会调整防御编组，使得互相的影响力范围不会重叠。

[有效范围] 0~255 [默认值] 5 [编号] 92

## sn-defense-distance

设置单位受保护的距离。在该数量的距离内，AI的对象会受到保护（不包括城镇）。

[有效范围] 0~255 [默认值] 3 [编号] 57

## sn-defer-dropsite-update

设为1，则资源放置点建造完毕时，变量dropsite-min-distance才会更新；设为0，则放置资源放置点地基时，上述变量就会更新。

[有效范围] 0~1 [默认值] 0 [编号] 273

## sn-desired-number-docks

（过时的）电脑玩家需求的船坞数量。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 212

## sn-desired-number-fish-escorts

（过时的）电脑玩家渔船需求的护卫战船数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 65

## sn-desired-number-fishing-boats

（过时的）电脑玩家需求的渔船数量。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 213

## sn-desired-number-trade-escorts

（过时的）电脑玩家贸易船需求的护卫战船数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 64

## sn-desired-number-transport-escorts

（过时的）电脑玩家运输船需求的护卫战船数量。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 66

## sn-desired-number-transports

（过时的）电脑玩家需求的运输船数量。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 214

## sn-desired-number-warships

（过时的）电脑玩家需求的战船数量。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 224

## sn-disable-attack-groups

设为1，禁止进攻编组自动寻敌，一旦创建了编组，就可以使用城镇范围进攻(TSA)进行防御性进攻。设为0，进攻编组会像通常一样执行具有侵略性的寻敌。

[有效范围] 0~1 [默认值] 0 [编号] 271

## sn-disable-builder-assistance

设为1，禁止建筑工建造完当前建筑后自动帮助建造邻近地基。设为0则相反。

[有效范围] 0~1 [默认值] 0 [编号] 285

## sn-disable-defend-groups

附加多个二进制标志位来禁用多种防御系统。

多个标志位之间可以累加以同时生效（如6=4+2，表示禁用其中4和2对应的两种防御系统）。

设为0，单位会正常对整个城镇范围内的威胁作出反应。

标志位1为禁用防御性进攻系统(TSA)；

标志位2为禁用sn-safe-town-size内的援助；

标志位4为禁用sn-safe-town-size与sn-maximum-town-size之间的援助；

标志位8为禁用sn-maximum-town-size之外的援助（相关示意图见sn-safe-town-size条目）。

请注意，当援助被禁用时，单位将只会反击位于他视野(或搜索半径?)内的敌人。

[有效范围] 0~15 [默认值] 0 [编号] 277

## sn-disable-sighted-response-cap

设为1，取消sn-enemy-sighted-response-distance最大值为50的限制；设为0，上述sn最大值为50。

[有效范围] 0~1 [默认值] 0 [编号] 284

## sn-disable-tower-priority

设为1，禁止本地寻敌系统给塔、城镇中心、城堡等防御工事赋予特殊优先级；设为0，这些建筑会正常收到特殊优先级。

[有效范围] 0~1 [默认值] 0 [编号] 267

## sn-disable-trade-evasion

设为1，禁止贸易车为了躲避攻击者而放弃他们的黄金和贸易路线。设为0，贸易车会正常试图躲避攻击者。

[有效范围] 0~1 [默认值] 0 [编号] 294

## sn-disable-villager-garrison

设为3，禁止村民以任何原因自动进驻。

设为2，如果村民可以进驻到城镇中心，他们将只会因为敌人的攻击而进驻（盖亚会被忽略）。

设为1，村民会因为任何敌人的攻击而进驻（盖亚会被忽略）。

设为0，村民会正常进驻。

[有效范围] 0~3 [默认值] 0 [编号] 291

## sn-do-not-scale-for-difficulty-level

设为1，禁止自动根据难度扩大各种策略值。

设为0，则系统会自动将设好的策略值，依据难度乘以不同的比例。最易25%，标准50%，中等75%，难/困难/极限100%。这个机制往往不太可控，若有条件自己为不同难度适配不同的策略值数值，则建议关掉它。

[有效范围] 0~1 [默认值] 0 [编号] 229

## sn-do-not-transport-from-same-zone

设为1来影响运输船的行为。

[有效范围] 0~1 [默认值] 0 [编号] 241

## sn-dock-avoidance-factor

设置避免在同一片水域修建多个码头的倾向因子。正值会防止在同一片水域修建，负值则倾向于这样修建。

[有效范围] -10000~10000 [默认值] 1000 [编号] 280

## sn-dock-defend-priority

设置船坞的防御优先级。

[有效范围] 0~1 [默认值] 0 [编号] 70

## sn-dock-placement-mode

设置船坞修建地点的倾向性。设为1，倾向于前置；设为-1，倾向于后置；设为0，没有倾向。

[有效范围] -1~1 [默认值] 0 [编号] 278

## sn-dock-proximity-factor

设置船坞修建相对密集程度。正值表示更大的间距，负值表示更小的间距。

[有效范围] -10000~10000 [默认值] 10000 [编号] 279

## sn-dock-training-filter

设为1或2，启用船坞的智能训练过滤功能。这个功能避免船坞训练在本水域很可能无用的船只种类。

设为1，则会继续在没有发现目标的水域里训练船只；

设为2，则不会继续训练。

设为0，船坞训练船只时不会考虑水域状况。

[有效范围] 0~2 [默认值] 0 [编号] 281

## sn-dropsite-separation-distance

设置资源放置点之间的最小间距。较大的值有利于资源采集者受到攻击时躲避。

[有效范围] 1~255 [默认值] 10 [编号] 248

## sn-easier-reaction-percentage

设置简单难度单人场景或战役中，电脑远程单位会对进入视野的该百分数内的敌对单位作出有效反应。

[有效范围] 0~100 [默认值] 100 [编号] 219

## sn-easiest-reaction-percentage

设置最易难度单人场景或战役中，电脑远程单位会对进入视野的该百分数内的敌对单位作出有效反应。

[有效范围] 0~100 [默认值] 100 [编号] 218

## sn-enable-boar-hunting

设为1，可以鹿和猪为目标；设为2，忽略鹿。

[有效范围] 0~2 [默认值] 0 [编号] 244

## sn-enable-new-building-system

只须设为1一次，请求使用新建造系统，其特征是允许同时进行多个建造及取消控制。

[有效范围] 0~1 [默认值] 0 [编号] 242

## sn-enable-offensive-priority

设为1，允许attack-now和进攻编组使用up-set-offense-priority设置的优先级。

[有效范围] 0~1 [默认值] 0 [编号] 254

## sn-enable-patrol-attack

设为1，启用巡逻式的本地寻敌系统。当进攻远方目标时，这会使得单位迅速切换目标，转而去攻击沿途所见单位，而不是等到他们靠近原先目标之后才重新索敌。在TSA期间，巡逻式寻敌系统可大大提升单个士兵组成的编组的行动效率，但对于多个士兵组成的编组则收效甚微。

[有效范围] 0~1 [默认值] 0 [编号] 247

## sn-enable-training-queue

设为1~14，则允许在每个建筑的训练队列中总共添加2~15个单位，就像人类玩家一样。这可以有效避免在衔接生产时出现0~1秒的延迟；

设为0，每个建筑任一时刻只能有一个单位在训练队列中。

[有效范围] 0~14 [默认值] 0 [编号] 264

## sn-enemy-sighted-response-distance

设置当敌人攻击时，对该攻击作出反应的己方单位距离攻击现场的最大距离。简称：敌视回应距离。

[有效范围] 0~50 [默认值] 25 [编号] 20

## sn-explore-group-makeup

（过时的）

[有效范围] -32767~32768 [默认值] 0 [编号] 45

## sn-filter-under-attack

设为1或2，撤退命令只会对受到攻击的单位生效。设为2，受到威胁的单位旁边的单位也会撤退，这可能会影响性能。设为0，不会进行过滤操作。

[有效范围] 0~2 [默认值] 0 [编号] 276

## sn-focus-player-number

设置为任一玩家编号，在事实和动作中的可通过focus-player标识符来引用这个策略值。0代表盖亚。

[有效范围] 0~8 [默认值] 0 [编号] 251

## sn-food-dropsite-distance

设置电脑玩家愿意在放置食物时行走的最大格数。

[有效范围] 3~255 [默认值] 3 [编号] 163

## sn-food-gatherer-percentage

设置食物采集者数量的参数，用于以下公式：食物采集者数量 = (sn-food-gatherer-percentage + sn-food-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] 0~100 [默认值] 0 [编号] 117

## sn-food-modifier-percentage

设置食物采集者数量的参数，用于以下公式：食物采集者数量 = (sn-food-gatherer-percentage + sn-food-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 156

## sn-forage-defend-priority

设置浆果采集点的防御优先级。

[有效范围] 0~1 [默认值] 0 [编号] 52

## sn-free-siege-targeting

1代表投石机，2代表火炮战船，将上述标志位加总得到本sn的值。

设置了标志位的相应单位在给出attack-now命令或组成进攻编组时，会启用自由寻敌。这会导致这些单位除进攻建筑外，还会进攻其他单位。

若设为0，不会进攻非建筑，除非sn-enemy-sighted-response-distance要求它们必须做出回击。

[有效范围] 0~3 [默认值] 0 [编号] 282

## sn-garrison-rams

设为1，电脑玩家在进攻编组出发前会尝试把步兵部队驻扎进冲车；设为0则不会。

[有效范围] 0~1 [默认值] 1 [编号] 240

## sn-gate-type-for-wall

设为1，AI会选择在城墙上建造木门；设为0为石门。DE版Build 42848新增。

[有效范围] 0~1 [默认值] 0 [编号] 232

## sn-gather-defense-units

设为1，指派部队防御正在修建的建筑。

[有效范围] 0~1 [默认值] 0 [编号] 232

## sn-gather-idle-soldiers-at-center

设为1，则城镇防御集合点会设为地图中心周围6格的随机范围内。如果地图中心不可到达则失效。一旦设置，所有闲置和撤退部队都会尝试去地图中心。在占山为王及类似的模式中有用。

[有效范围] 0~1 [默认值] -1 [编号] 239

## sn-gold-defend-priority

设置金矿的防御优先级。0表示不防御金矿，1表示金矿具有最高防御优先级。

[有效范围] 0~1 [默认值] 0 [编号] 50

## sn-gold-dropsite-distance

设置电脑玩家愿意在放置黄金时行走的最大格数。

[有效范围] 3~255 [默认值] 3 [编号] 166

## sn-gold-gatherer-percentage

设置黄金采集者数量的参数，用于以下公式：黄金采集者数量 = (sn-gold-gatherer-percentage + sn-gold-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 118

## sn-gold-modifier-percentage

设置黄金采集者数量的参数，用于以下公式：黄金采集者数量 = (sn-gold-gatherer-percentage + sn-gold-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 159

## sn-group-commander-selection-method

设置选择编组领导者的方法。0表示选择生命值最高者；1表示选择生命值最低者；2表示选择射程最远者。编组领导者在编组创建时确定，只有当领导者死亡，才会重新选择。

[有效范围] 0~3 [默认值] 3 [编号] 75

## sn-group-fill-method

设置电脑玩家填充编组单位的方法。

0表示单一编组填充，1表示逐级编组填充。

在单一编组填充中，所有可用单位先被尽可能填充进第一个未满编组中，填满第一个后，再填充第二个，第二个未填充满前，不会填充第三个，以此类推。

在逐级编组填充中，所有编组是同时填满的，每个单位都考虑所有编组。在单一编组填充至每个编组都达到认为组满的最小单位数量后，会转而使用逐级编组填充方法。

两种方法都会遵守最大编组规模的控制。

[有效范围] 0~1 [默认值] -1 [编号] 40

## sn-group-form-distance

设置可以进入同一编组的部队间的最大间距。如果生产军事单位的建筑相距很远，需要较大的此值。

[有效范围] 0~30 [默认值] 20 [编号] 230

## sn-group-leader-defense-distance

设置防御部队防守进攻编组中重要领导者的防御距离。

[有效范围] 1~255 [默认值] 3 [编号] 131

## sn-hits-before-alliance-change

设置电脑玩家改变对某玩家外交关系前可以忍受的来自其进攻的次数。

[有效范围] 0~32768 [默认值] 3 [编号] 221

## sn-home-exploration-time

设置应该用于探索主城镇中心周围区域的最大时间，单位为秒。

[有效范围] 0~32768 [默认值] 300 [编号] 256

## sn-house-overage

（过时的）设置电脑玩家能容纳50人口后，自动额外修建的住房数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 152

## sn-ignore-attack-group-under-attack

设为1，表示防御部队会忽略遭受攻击的进攻部队。

[有效范围] 0~1 [默认值] 0 [编号] 231

## sn-ignore-tower-elevation

设为1，修建塔时忽略高低地形；设为0，修建塔时尽可能修在高地。

[有效范围] 0~1 [默认值] 0 [编号] 265

## sn-initial-attack-delay

设置电脑玩家初始强制不可进攻的秒数。

[有效范围] 0~32768 [默认值] 0 [编号] 104

## sn-initial-attack-delay-type

设置初始进攻延迟的种类。

1表示建筑队列修建完毕标志着允许进攻；

2表示使用sn-initial-attack-delay的设置值；

3表示电脑玩家被非盖亚玩家攻击后即可发起攻击；

0表示上述3种情况的任一种都会允许电脑玩家进攻。

[有效范围] 0~3 [默认值] 0 [编号] 134

## sn-initial-exploration-required

设置电脑玩家可以修建建筑前必须探索完毕的地图百分比。

[有效范围] 0~100 [默认值] 2 [编号] 167

## sn-intelligent-gathering

设置智能采集系统是否启用。

[有效范围] 0~1 [默认值] 0 [编号] 142

## sn-livestock-defend-priority

设为1，允许绵羊加入到可作为目标的可视物件列表。

[有效范围] 0~1 [默认值] 0 [编号] 287

## sn-livestock-to-town-center

设为1，强制绵羊一类牲畜聚集在城镇中心；设为0，则也允许牲畜聚集在磨坊。

[有效范围] 0~1 [默认值] 0 [编号] 263

## sn-local-targeting-mode

设置本地寻敌模式。

设为1，优先寻找高攻击加成且高每击伤害值的目标；

设为2，优先攻击高基础远程防御目标，如冲车；

设为0，不使用以上两种模式，正常寻敌。

目标的进攻优先级(-1~11)会作为权重加入模式1和2的计算中。

[有效范围] 0~2 [默认值] 0 [编号] 286

## sn-lock-attack-and-attack-response

设为1，把sn-attack-separation-time和sn-attack-response-separation-time视作同一值。

[有效范围] 0~1 [默认值] -1 [编号] 71

## sn-lumber-camp-max-distance

非零时，设置放置伐木场的最大城镇范围。若为0，则会使用sn-camp-max-distance的值。

[有效范围] 0~255 [默认值] 0 [编号] 260

## sn-max-build-plan-gatherer-percentage

设置根据游戏前建筑计划的要求，电脑玩家指派的采集者数量最大百分比。

[有效范围] 0~100 [默认值] 50 [编号] 160

## sn-max-farms

（过时的）设置电脑玩家修建农田的最大数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 220

## sn-max-granaries

（过时的）设置电脑玩家修建谷仓的最大数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 151

## sn-max-retask-gather-amount

设置一个采集者在被重新分配任务前可以采集的最大资源数量。某些其他代码会覆盖本设置。

[有效范围] 0~32768 [默认值] 40 [编号] 149

## sn-max-skips-per-attempt

设置建造计划中能够跳过的未建项目的最大数量。

[有效范围] 1~32768 [默认值] 10 [编号] 115

## sn-max-storage-pits

（过时的）设置电脑玩家修建仓库的最大数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 150

## sn-max-towers

设置电脑玩家修建塔的最大数量。必须非负。默认情况下开始为0，一段时间后变成10。

[有效范围] 0~32768 [默认值] 10 [编号] 222

## sn-maximum-attack-group-size

设置陆地进攻编组的最大规模。

[有效范围] 0~32768 [默认值] 10 [编号] 26

## sn-maximum-boat-attack-group-size

设置海上进攻编组的最大规模。

[有效范围] 0~32768 [默认值] 5 [编号] 60

## sn-maximum-boat-defend-group-size

设置海上防御编组的最大规模。

[有效范围] 0~32768 [默认值] 0 [编号] 69

## sn-maximum-boat-explore-group-size

设置海上探索编组的最大规模。

[有效范围] 0~32768 [默认值] 2 [编号] 63

## sn-maximum-defend-group-size

设置陆地防御编组的最大规模。

[有效范围] 0~32768 [默认值] 4 [编号] 28

## sn-maximum-explore-group-size

设置陆地探索编组的最大规模。

[有效范围] 0~1 [默认值] 1 [编号] 44

## sn-maximum-fish-boat-drop-distance

设置渔船可以采集的鱼类资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。若设为0，所有渔船都只会探索水域。

[有效范围] -1~255 [默认值] -1 [编号] 236

## sn-maximum-food

（过时的）设置电脑玩家愿意持有的食物库存的最大数量。

[有效范围] 0~32768 [默认值] 3000 [编号] 190

## sn-maximum-food-drop-distance

设置电脑玩家采集食物时资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。

[有效范围] -1~255 [默认值] -1 [编号] 234

## sn-maximum-gaia-attack-response

设置被自然动物攻击时，派去帮助被攻击者的村民的最大数量。

[有效范围] 0~32768 [默认值] 3 [编号] 100

## sn-maximum-garrison-fill

设置每条命令控制的驻扎入某个目标的单位数量。若设为0，则不作限制。

[有效范围] 0~20 [默认值] 0 [编号] 274

## sn-maximum-gold

（过时的）设置电脑玩家愿意持有的黄金库存的最大数量。

[有效范围] 0~32768 [默认值] 3000 [编号] 193

## sn-maximum-gold-drop-distance

设置电脑玩家采集黄金时资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。

[有效范围] -1~255 [默认值] -1 [编号] 237

## sn-maximum-houses-before-dropsites

（过时的）设置修建第一个资源放置点前能修建房屋的最大数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 174

## sn-maximum-hunt-drop-distance

设置电脑玩家捕猎时资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。

[有效范围] -1~255 [默认值] -1 [编号] 235

## sn-maximum-stone

（过时的）设置电脑玩家愿意持有的石料库存的最大数量。

[有效范围] 0~32768 [默认值] 1500 [编号] 192

## sn-maximum-stone-drop-distance

设置电脑玩家采集石料时资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。

[有效范围] -1~255 [默认值] -1 [编号] 238

## sn-maximum-town-size

设置最大城镇范围，相当于现实中的领土范围。城镇范围的中心为AI主城镇中心，若AI没有了城镇中心，可能这个机制会失效。

AI会指派防御部队去进攻出现在最大城镇范围内的敌人，直至将城镇内的敌人全部消灭。

这通常用于TSA（town-size-attack）防御性进攻模式——只要将城镇范围扩大到足以覆盖其他玩家的领地，AI就会用防御部队去进攻敌方领地。进入TSA模式的部队，中途受到攻击会立刻反击，且由于部队是各自单独行动，因此机动性不会受到攻城武器、僧侣等的拖累。当AI军队数量庞大时，这会形成排山倒海一般的冲锋，不过如果军队数量匮乏时发动TSA，会出现挨个送死的行为。

另外，TSA模式下，AI几乎不会动用海军，也不会试图用运输船登陆作战。

本策略值与其他策略值的搭配效果见sn-safe-town-size。

[有效范围] 0~255 [默认值] 20 [编号] 74

## sn-maximum-wood

（过时的）设置电脑玩家愿意持有的木材库存的最大数量。

[有效范围] 0~32768 [默认值] 2000 [编号] 191

## sn-maximum-wood-drop-distance

设置电脑玩家采集木材时资源距离资源放置点的最大距离。-1表示忽略最大距离设置，即任意远距离都会采集。

[有效范围] -1~255 [默认值] -1 [编号] 233

## sn-mill-max-distance

设置磨坊能被修建在距离城镇中心的最大距离。

[有效范围] 4~255 [默认值] 100 [编号] 87

## sn-minimum-amount-for-trading

设置一种资源能被出卖前必须积累到的最小数量。

[有效范围] 0~32768 [默认值] 50 [编号] 216

## sn-minimum-attack-group-size

设置陆地进攻编组的最小规模。一个编组只有达到这个规模才会被系统指派进攻，人数过少的编组不会出击。游戏会通过sn-scale-minimum-attack-group-size及其放缩机制来自动改变此值，不过这可能会造成游戏后期此值累加得过高，导致AI进攻积极性降低。

[有效范围] 0~32768 [默认值] 4 [编号] 16

## sn-minimum-boar-hunt-group-size

设置电脑玩家杀猪之前必须集合到的村民最小数量。

[有效范围] 0~8 [默认值] 5 [编号] 204

## sn-minimum-boar-lure-group-size

设置电脑玩家派遣收集猪肉(?)（或加入初始引猪组里(?)）的村民的数量。

[有效范围] 0~8 [默认值] 0 [编号] 252

## sn-minimum-boat-attack-group-size

设置海上进攻编组的最小规模。一个编组只有达到这个规模才能下达命令。

[有效范围] 0~32768 [默认值] 1 [编号] 59

## sn-minimum-boat-defend-group-size

设置海上防御编组的最小规模。

[有效范围] 0~32768 [默认值] 0 [编号] 68

## sn-minimum-boat-explore-group-size

设置海上探索编组的最小规模。

[有效范围] 0~32768 [默认值] 1 [编号] 62

## sn-minimum-civilian-explorers

设置派去探索的村民的最小数量。

[有效范围] 0~32768 [默认值] 0 [编号] 35

## sn-minimum-defend-group-size

设置陆地防御编组的最小规模。一个编组只有达到这个规模才能下达命令。

[有效范围] 0~32768 [默认值] 1 [编号] 25

## sn-minimum-dropsite-buffer

设置电脑玩家修建的资源放置点距离敌人城镇中心的最小距离。

[有效范围] 0~255 [默认值] 25 [编号] 202

## sn-minimum-explore-group-size

设置陆地军队探索编组的最小规模。一个编组只有达到这个规模才能下达命令。

[有效范围] 0~1 [默认值] 1 [编号] 43

## sn-minimum-food

（过时的）设置电脑玩家愿意持有的食物库存最小数量。

[有效范围] 0~32768 [默认值] 200 [编号] 170

## sn-minimum-gold

（过时的）设置电脑玩家愿意持有的黄金库存最小数量。

[有效范围] 0~32768 [默认值] 200 [编号] 173

## sn-minimum-number-hunters

强制捕猎。为达成杀猪时的最佳效果，可配合sn-minimum-boar-hunt-group-size使用。

[有效范围] 0~32768 [默认值] 0 [编号] 245

## sn-minimum-peace-like-level

设置电脑玩家愿意与另一个玩家结盟所要达到的最低好感度。

[有效范围] 0~100 [默认值] 85 [编号] 29

## sn-minimum-stone

（过时的）设置电脑玩家愿意持有的石料库存最小数量。

[有效范围] 0~32768 [默认值] 200 [编号] 172

## sn-minimum-town-size

设置最小城镇范围。这或许可用于TSA（town-size-attack）防御性进攻模式 ( ? 不过大部分情况下都是通过调整sn-maximum-town-size实现)。

[有效范围] 0~255 [默认值] 12 [编号] 73

## sn-minimum-water-body-size-for-dock

设置电脑玩家愿意修建船坞于其上的海域需要的最小面积，以格计。

[有效范围] 10~32768 [默认值] 300 [编号] 112

## sn-minimum-wood

（过时的）设置电脑玩家愿意持有的木材库存最小数量。

[有效范围] 0~32768 [默认值] 200 [编号] 171

## sn-mining-camp-max-distance

非0时，为矿场的放置设置最大城镇范围。若设为0，则使用sn-camp-max-distance值。

[有效范围] 0~255 [默认值] 0 [编号] 261

## sn-most-needed-resource-look-ahead

（过时的）设置电脑玩家前瞻计算需求资源时所考虑的建筑队列中建筑数量。只用于动态百分比采集设定。必须非负。

[有效范围] 0~32768 [默认值] 10 [编号] 147

## sn-number-attack-groups

设置陆地进攻编组数量。

若为0，AI将停止指派新的防御部队加入进攻编组，当前已在进攻编组的单位仍会继续进攻。

若不为0，AI将立刻把尽可能多的防御部队编入指定组数的进攻编组去进攻敌人，此时：

若所有进攻编组都满员了则剩余的防御部队维持现状，这个过程会持续进行直到本策略值重新设为0；

若在进攻的同时，将sn-maximum-attack-group-size和sn-minimum-attack-group-size设为1，则可实现类似TSA的大部队分散冲锋的效果，行军前无需集中，行军途中会自动反击，缺点是巨型投石机可能会在原地重复拆装而不进攻——这种进攻模式被称为NAG(number-attack-groups)或GroupATK。

在GroupATK期间，sn-percent-attack-soldiers可有可无，因为后者是在执行attack-now那一刻才生效一次。

[有效范围] 0~32768 [默认值] 0 [编号] 36

## sn-number-boat-attack-groups

设置海上进攻编组数量。sn-percent-attack-boat通常工作得更好。

[有效范围] 0~32768 [默认值] 0 [编号] 58

## sn-number-boat-defend-groups

设置海上防御编组数量。

[有效范围] 0~32768 [默认值] 0 [编号] 67

## sn-number-boat-explore-groups

设置海上战船探索编组数量。不受sn-total-number-explorers影响。

[有效范围] 0~32768 [默认值] 0 [编号] 61

## sn-number-build-attempts-before-skip

设置一个建筑在被跳过修建前能尝试的最大修建次数。

[有效范围] 1~32768 [默认值] 25 [编号] 114

## sn-number-civilian-militia

设置能用来进攻的村民的最大数量。

[有效范围] 0~200 [默认值] 10 [编号] 257

## sn-number-defend-groups

设置陆地防御编组数量。

[有效范围] 0~32768 [默认值] 0 [编号] 38

## sn-number-enemy-objects-required

设置电脑玩家把探索村民数量降为最小值前，必须发现的敌方物件数量。

[有效范围] 0~32768 [默认值] 10 [编号] 145

## sn-number-explore-groups

设置陆地军队探索编组数量。

[有效范围] 0~32768 [默认值] 0 [编号] 42

## sn-number-forward-builders

设置电脑玩家用来修建前置建筑的村民数量。这里的前置村民特指必须乘运输船穿过一片其他方式无法通过的水域，或者路径被墙封掉的情况。除非必须登运输船，否则不需要设置“前置村民”。

[有效范围] 0~32768 [默认值] 0 [编号] 226

## sn-number-garrison-units

设置每条命令能命令多少部队驻扎。若设为0，则等效于最大值40。

[有效范围] 0~40 [默认值] 0 [编号] 275

## sn-number-max-skip-cycles

[有效范围] 1~32768 [默认值] 50 [编号] 146

## sn-number-tasked-units

设置用up-target-objects和up-target-point指派单位时，每组所需的最小单位数量。

设为0：

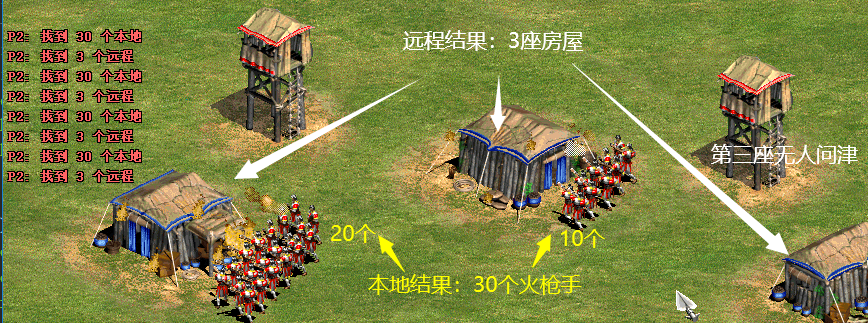
①对于up-target-objects，本地结果里的单位会被尽可能平均分配到所有远程结果上，注意，如果本地结果数量 < 远程结果数量，那么靠后的那些远程结果自然就“无人问津”了；

②对于up-target-point，会均匀分成2组，分别前往目的地。

设为>0的数：

比如设为20，但本地结果里共有30个单位，那么会分成20、10个共2组，各组分别前往各自的目标。

再如设为40，但本地结果里只有30个单位，那么只分成30个共1组，一次只能前往一个目标。



[有效范围] 0~40 [默认值] 0 [编号] 288

## sn-number-wall-gates

设置环绕电脑玩家的城墙中需要修建城门的数量。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 84

## sn-object-repair-level

设置维修级别。

0=奇观，1=城堡&修道院，2=城镇中心，4=兵营，8=靶场，

16=马厩，32=攻城武器厂，64=船坞，128=市场，256=大学，

512=铁匠铺，1024=伐木场&矿场&磨坊，2048=住房，4096=塔，8192=城墙&城门，16384=攻城武器。

将上述标志位加总得到本sn的值。

在场景和战役中，默认值为1。

[有效范围] 0~32767 [默认值] 16387 [编号] 246

## sn-percent-attack-boats

设置下一次attack-now命令发出后，多少百分比的防御船只会被派遣战斗。

值会根据难度作调整。

所有新建造的船只默认为防御船只，直到attack-now命令发出前，都会保持为防御船只。attack-now命令发出时，陆地和海上进攻部队都会进攻。

本sn只须设置一次就持续生效，当然，需要改变时可以改变它。

[有效范围] 0~100 [默认值] 75 [编号] 228

## sn-percent-attack-soldiers

设置下一次attack-now命令发出后，多少百分比的防御部队会被派遣战斗。

值会根据难度作调整。

所有新建造的部队默认为防御部队，直到attack-now命令发出前，都会保持为防御部队。例如，当前有10防御单位，本sn设为50，那么其中5单位组成进攻编组并进攻。

本sn只须设置一次就持续生效，当然，需要改变时可以改变它。

本sn在不使用sn-number-defend-groups时能发挥最大效果。

[有效范围] 0~100 [默认值] 75 [编号] 227

## sn-percent-building-cancellation

设置能被取消的正在修建建筑的最大已修建百分比。

[有效范围] 1~100 [默认值] 100 [编号] 243

## sn-percent-civilian-builders

设置分配为建筑工的村民百分比。

[有效范围] 0~100 [默认值] 0 [编号] 1

## sn-percent-civilian-explorers

设置派遣去探索的村民百分比。

[有效范围] 0~100 [默认值] 34 [编号] 0

## sn-percent-civilian-gatherers

设置派遣去采集资源的村民百分比。

[有效范围] 0~100 [默认值] 66 [编号] 2

## sn-percent-death-retreat

设置一个编组死亡多少百分比的单位后才撤退。必须大于等于1且小于等于100。

[有效范围] 0~100 [默认值] -1 [编号] 31

## sn-percent-enemy-sighted-response

设置某个单位受到攻击后，闲置部队响应该攻击的数量百分比。简称：敌视回应比率。

[有效范围] 0~100 [默认值] 50 [编号] 19

## sn-percent-exploration-required

设置电脑玩家给探索村民重新分配任务前必须探索完毕的地图百分比。

[有效范围] 0~100 [默认值] 100 [编号] 32

## sn-percent-half-exploration

设置探索多少百分比的地图之后，电脑玩家才会分配原来一半的探索者。

[有效范围] 0~100 [默认值] 30 [编号] 179

## sn-percent-health-retreat

设置一个编组失去总血量的多少百分比之后才撤退。必须大于等于1且小于等于100。

[有效范围] 0~100 [默认值] -1 [编号] 30

## sn-percent-unit-health-retreat

设置一个单位失去总血量的多少百分比之后才撤退。必须大于等于1且小于等于100。

[有效范围] 0~100 [默认值] -1 [编号] 91

## sn-percent-victory-clamp

[有效范围] 0~100 [默认值] 75 [编号] 24

## sn-percentage-explore-exterminators

设置多少百分比的部队探索编组被设为死斗编组。必须非负且小于等于100。

[有效范围] 0~100 [默认值] 50 [编号] 198

## sn-placement-fail-delta

设置每次遍历后，无法在up-set-placement-data指定的地点修建建筑时，尝试增加到选址距离的值，亦即对up-set-placement-data设置的(in)(op)选址距离 进行逐次的累加直到成功放置地基。

这里有个帖子讨论各个影响因素和生效机理：[place-control（选址控制系统）的影响参数的探讨交流](https://www.hawkaoe.net/bbs/thread-146375-1-1.html)。

[有效范围] -10~10 [默认值] 0 [编号] 269

## sn-placement-to-center

设为1，强制选址控制系统使用地图中心作为第二个参考点，第一个参考点是up-set-placement-data选定的。

设为0，会使用当前目标敌人最近的建筑作为第二个参考点。如果sn-target-player-number是0，则会使用sn-attack-winning-player确定目标敌人。

这里有个帖子讨论各个影响因素和生效机理：[place-control（选址控制系统）的影响参数的探讨交流](https://www.hawkaoe.net/bbs/thread-146375-1-1.html)。

[有效范围] 0~1 [默认值] 0 [编号] 270

## sn-placement-zone-size

设置前置和选址控制系统使用的区域大小。

伴随着每次成功呼叫(call)后，所有的build命令都会存储本策略值的值和up-set-placement-data信息。每次遍历中若建筑修建不成功，其区域大小都会从这个起始值开始增加。

这里有个帖子讨论各个影响因素和生效机理：[place-control（选址控制系统）的影响参数的探讨交流](https://www.hawkaoe.net/bbs/thread-146375-1-1.html)。

[有效范围] 0~255 [默认值] 20 [编号] 268

## sn-preferred-mill-placement

设置磨坊优先修建于何种资源旁边。0表示果树丛，1表示鹿，2表示沿海鱼群。

[有效范围] 0~2 [默认值] 0 [编号] 253

## sn-preferred-trade-distance

设置偏好的贸易距离，即本地和远程的贸易建筑间距。

[有效范围] 0~255 [默认值] 100 [编号] 259

## sn-random-placement-factor

设置电脑玩家建筑选址的随机因素。

[有效范围] 0~32768 [默认值] 50 [编号] 168

## sn-relic-defend-priority

（过时的）设置圣物车队的防御优先级。

[有效范围] 0~7 [默认值] 0 [编号] 55

## sn-relic-return-distance

设置电脑玩家愿意拾取圣物的最远距离。

[有效范围] 0~255 [默认值] 10 [编号] 23

## sn-required-first-building

（过时的）设置电脑玩家必须先修建何种建筑。0表示无限制，1表示仓库或谷仓，2表示仓库，3表示谷仓，4表示仓库和谷仓。例外是城镇中心和sn-maximum-houses-before-dropsites。

[有效范围] 0~4 [默认值] -1 [编号] 161

## sn-required-forest-tiles

设置电脑玩家修建伐木场前必须探索到的森林格数。

[有效范围] 0~32768 [默认值] 10 [编号] 169

## sn-retask-gather-amount

设置允许给一个资源采集者重新分配任务之前，该采集者必须采集到的资源数量。某些代码会覆盖其效果。

[有效范围] 0~32768 [默认值] 20 [编号] 148

## sn-retreat-after-target-destroyed

设置攻击过程中，目标死亡或摧毁时部队如何行动。

0表示进攻部队永不撤退，按照其位置重新选择目标；

1表示选择不到其他目标时撤退；

2表示目标死亡后部队会立即撤退；

3表示部队会进入死斗模式，去探索未开拓地图，并攻击任何路过的敌人。

[有效范围] 0~3 [默认值] -1 [编号] 49

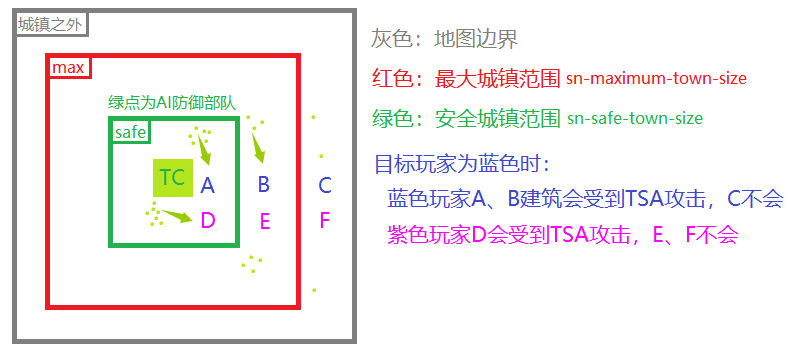
## sn-ruins-defend-priority

设置废墟的防御优先级。

[有效范围] 0~7 [默认值] 0 [编号] 54

## sn-safe-town-size

如果敌方建筑既位于sn-maximum-town-size内，也同时位于本sn值范围内，就会被防御部队所攻击。如果敌方建筑修建在sn-maximum-town-size内，但在本sn区域外，那么只有该建筑属于sn-target-player-number玩家时，防御部队才会攻击它。



[有效范围] 1~255 [默认值] 255 [编号] 250

## sn-save-scenario-information

设置场景结束时某个电脑玩家的学习信息是否存储。

[有效范围] 0~1 [默认值] 0 [编号] 105

## sn-scale-maximum-attack-group-size

设置最大进攻编组规模的放缩值。当AI执行放缩操作时，本数值加到sn-maximum-attack-group-size中。

[有效范围] 0~32768 [默认值] 0 [编号] 94

## sn-scale-minimum-attack-group-size

设置最小进攻编组规模的放缩值。当AI执行放缩操作时，本数值加到sn-minimum-attack-group-size中。这或许会造成游戏时间过长时，sn-minimum-attack-group-size被放缩机制增加得过大，从而降低AI玩家的部队的进攻积极性，如果想避免麻烦，可以考虑把这个策略值设为0。

[有效范围] 0~32768 [默认值] 1 [编号] 93

## sn-scale-percent-death-retreat

设置死亡百分比撤退的放缩值。当AI执行放缩操作时，本数值加到sn-percent-death-retreat中。

[有效范围] 0~32768 [默认值] -1 [编号] 96

## sn-scale-percent-health-retreat

设置血量百分比撤退的放缩值。当AI执行放缩操作时，本数值加到sn-percent-health-retreat中。

[有效范围] 0~32768 [默认值] -1 [编号] 95

## sn-scale-percent-unit-health-retreat

设置单位血量百分比撤退的放缩值。当AI执行放缩操作时，本数值加到sn-percent-unit-health-retreat中。

[有效范围] 0~32768 [默认值] -1 [编号] 97

## sn-scaling-frequency

设置AI两次执行放缩操作的间隔分钟数。这或许会造成游戏时间过长时，sn-minimum-attack-group-size被放缩机制增加得过大，从而降低AI玩家的部队的进攻积极性。

[有效范围] 0~32768 [默认值] 10 [编号] 99

## sn-sentry-distance

设置防御城镇的距离。

[有效范围] 0~255 [默认值] 12 [编号] 22

## sn-sentry-distance-variation

设置允许防御距离上下浮动的范围。

[有效范围] 0~32768 [默认值] 2 [编号] 72

## sn-size-wall-gates

（过时的）设置电脑玩家城镇周围城墙内城门的大小，以格计。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 85

## sn-special-attack-influence1

设置电脑玩家寻敌评价系统中，特殊进攻类型1的评级乘数。

大于0表示电脑玩家倾向进攻特殊类型1；

小于0表示电脑玩家不倾向进攻特殊类型1。

[有效范围] -32767~32768 [默认值] 0 [编号] 109

## sn-special-attack-influence2

设置电脑玩家寻敌评价系统中，特殊进攻类型2的评级乘数。

大于0表示电脑玩家倾向进攻特殊类型2；

小于0表示电脑玩家不倾向进攻特殊类型2。

[有效范围] -32767~32768 [默认值] 0 [编号] 110

## sn-special-attack-influence3

设置电脑玩家寻敌评价系统中，特殊进攻类型3的评级乘数。

大于0表示电脑玩家倾向进攻特殊类型3；

小于0表示电脑玩家不倾向进攻特殊类型3。

[有效范围] -32767~32768 [默认值] 0 [编号] 111

## sn-special-attack-type1

设为1表示以修道院和拿着圣物的僧侣为目标。

[有效范围] -1~1 [默认值] -1 [编号] 106

## sn-special-attack-type2

设为任意单位、建筑或编组编号，从而引导攻击。

[有效范围] -1~32768 [默认值] -1 [编号] 107

## sn-special-attack-type3

设为1表示以奇观为目标。

[有效范围] -1~1 [默认值] -1 [编号] 108

## sn-specific-build-item-time

设置需要分配给sn-specific-build-item-to-build的分钟数。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 176

## sn-specific-build-item-to-build

设置需要插入到电脑玩家建筑队列中的特定建造项目。必须是有效的建筑编号。

[有效范围] 0~32768 [默认值] -1 [编号] 175

## sn-stone-defend-priority

设置石矿的防御优先级。

[有效范围] 0~1 [默认值] 0 [编号] 51

## sn-stone-dropsite-distance

设置电脑玩家愿意采集石料的最大距离。

[有效范围] 3~255 [默认值] 3 [编号] 165

## sn-stone-gatherer-percentage

设置石料采集者数量的参数，用于以下公式：石料采集者数量 = (sn-stone-gatherer-percentage + sn-stone-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 119

## sn-stone-modifier-percentage

设置石料采集者数量的参数，用于以下公式：石料采集者数量 = (sn-stone-gatherer-percentage + sn-stone-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 158

## sn-tactical-update-frequency

设置战术AI两次更新间隔的秒数。必须非负。

[有效范围] 0~32768 [默认值] -1 [编号] 88

## sn-target-evaluation-ally-proximity

设置电脑玩家寻敌评价系统中，同盟密集度（范围中的同盟数量）的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 81

## sn-target-evaluation-attack-attempts

设置电脑玩家寻敌评价系统中，进攻尝试次数的评级乘数。

[有效范围] -32767~32768 [默认值] -25 [编号] 89

## sn-target-evaluation-boat

设置陆地单位进攻船只的倾向性。大于0表示陆地单位倾向于进攻船只，0表示没有特殊倾向，小于0表示陆地单位倾向于不进攻船只。

[有效范围] -32767~32768 [默认值] 0 [编号] 144

## sn-target-evaluation-continent

设置电脑玩家攻击与其进攻编组领导者位于同一大陆上的敌人的倾向性。大于0表示倾向于攻击同一大陆上的敌人，小于0反之，0没有倾向性。

[有效范围] -32767~32768 [默认值] 100 [编号] 122

## sn-target-evaluation-damage-capability

设置电脑玩家寻敌评价系统中，伤害能力的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 79

## sn-target-evaluation-distance

设置电脑玩家寻敌评价系统中，距离的评级乘数。

[有效范围] -32767~32768 [默认值] 50 [编号] 77

## sn-target-evaluation-hitpoints

设置电脑玩家寻敌评价系统中，生命值的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 78

## sn-target-evaluation-in-progress

设置电脑玩家寻敌评价系统。设置继续攻击已经正在受到攻击的敌人的倾向性。

[有效范围] -32767~32768 [默认值] 50 [编号] 185

## sn-target-evaluation-kills

设置电脑玩家寻敌评价系统中，杀伤数的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 80

## sn-target-evaluation-randomness

设置电脑玩家寻敌评价系统中，随机因素的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 83

## sn-target-evaluation-range

设置电脑玩家寻敌评价系统中，射程的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 90

## sn-target-evaluation-rof

设置电脑玩家寻敌评价系统中，攻击频率的评级乘数。

[有效范围] -32767~32768 [默认值] 0 [编号] 82

## sn-target-evaluation-siege-weapon

设置电脑玩家寻敌评价系统。控制让攻城武器攻击固定目标，同时非攻城武器单位不攻击这些固定目标的倾向性。大于0表示倾向于使用攻城武器攻击固定目标，小于0反之，0无倾向性。

[有效范围] -32767~32768 [默认值] 0 [编号] 123

## sn-target-evaluation-time-kill-ratio

设置电脑玩家寻敌评价系统。控制在决定进攻何种单位时，杀死目标单位所需的时间这一因素的评价乘数。

[有效范围] -32767~32768 [默认值] 20 [编号] 184

## sn-target-player-number

设置攻击目标玩家编号。在事实和动作中的可通过target-player标识符来引用这个策略值。

若设为-1，则发起攻击时会协助同盟进攻。

若设为0，则由sn-attack-winning-player设置目标。

若设为不可以进攻的玩家（同盟或自己），会引起未定义的行为。

[有效范围] -1~8 [默认值] 0 [编号] 249

## sn-target-point-adjustment

调整up-target-point所指派点的格内方位。

设为0，则动作会指派到格子的最左角处。

1~5对应的方位见 微调指派点。

设为6，开启精确模式后，你可以让up-target-point指派到一对有效的精确目标点处。

[有效范围] 0~6 [默认值] 0 [编号] 292

## sn-task-ungrouped-soldiers

设置电脑玩家的未编组部队是否分散开并保卫城镇地区。

[有效范围] 0~1 [默认值] 1 [编号] 143

## sn-total-number-explorers

设置陆地探索者的最大数量。事先会计算村民探索者比例，并且部队编组进行完毕。设为-1则忽略。

[有效范围] -1~32768 [默认值] 4 [编号] 18

## sn-town-center-placement

设置城镇中心选址时模仿的建筑编号。若设为0，则按普通方法放置城镇中心。

[有效范围] 0~899 [默认值] 0 [编号] 266

## sn-town-defend-priority

设置城镇的防御优先级。

[有效范围] 0~7 [默认值] 7 [编号] 56

## sn-track-player-history

设置人类玩家的倾向是否被跟踪。

[有效范围] 0~1 [默认值] 0 [编号] 201

## sn-tribute-amount

（过时的）设置电脑玩家需要的黄金贡品数量。若设为0，则特殊系统不会监视电脑玩家进贡情况，使用正常进贡系统。若大于0，电脑玩家会期望收到黄金贡品。

[有效范围] 0~32768 [默认值] 0 [编号] 124

## sn-tribute-chat-frequency

（过时的）设置电脑玩家两次送出聊天内容请求贡品的间隔秒数。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 125

## sn-tribute-chat-randomness

（过时的）设置电脑玩家两次送出聊天内容请求贡品的间隔秒数的上下随机浮动。必须非负且小于sn-tribute-chat-frequency。

[有效范围] 0~32768 [默认值] 0 [编号] 126

## sn-tribute-failure-outcome

（过时的）设置电脑玩家没有在期望的时间内收到期望的贡品时发生的情况。设为0，什么都不会发生；设为1，电脑玩家会设置外交关系为敌对。

[有效范围] 0~1 [默认值] 0 [编号] 130

## sn-tribute-persistence

（过时的）设置互动进贡系统仅生效一次还是持续有效。设为1，则持续有效，即电脑玩家在整个游戏中，按照sn-tribute-chat-frequency的设置，每隔一段时间就要求贡品。设为0，则互动进贡只生效一次。

[有效范围] 0~1 [默认值] 0 [编号] 132

## sn-tribute-player

（过时的）设置电脑玩家期望从哪个玩家收到贡品。必须是游戏中的有效玩家编号。

[有效范围] 0~8 [默认值] -1 [编号] 128

## sn-tribute-revoke-on-attack

（过时的）设置当电脑玩家被sn-tribute-player进攻后，是否撤销已收到的贡品造成的和平结果。1表示撤销友好判定，0反之。

[有效范围] 0~1 [默认值] 0 [编号] 133

## sn-tribute-success-outcome

（过时的）设置电脑玩家在期望时间内成功收到了期望贡品时发生的情况。设为0，什么都不会发生；设为1，电脑玩家会设置外交关系为同盟。

[有效范围] 0~1 [默认值] 0 [编号] 129

## sn-tribute-timeout

（过时的）设置电脑玩家要求贡品在该秒数内必须上交。必须非负。

[有效范围] 0~32768 [默认值] 0 [编号] 127

## sn-unexplored-construction

设为1，允许AI把建筑修建在未探索的格子上。设为0，AI需要正常地在探索了格子后才能在那里建造。请不要在没有 #load-if-not-defined REVEAL-NORMAL 的情况下启用这个sn。

[有效范围] 0~1 [默认值] 0 [编号] 293

## sn-unskippable-item-type

禁止电脑玩家跳过建筑队列中的某个单位。必须是有效的建筑编号。

[有效范围] 0~32768 [默认值] -1 [编号] 177

## sn-upgrade-to-bronze-age-asap

（过时的）设置电脑玩家是否不顾一切地尽可能快地升级到铜器时代。

[有效范围] 0~1 [默认值] -1 [编号] 182

## sn-upgrade-to-iron-age-asap

（过时的）设置电脑玩家是否不顾一切地尽可能快地升级到铁器时代。

[有效范围] 0~1 [默认值] -1 [编号] 183

## sn-upgrade-to-tool-age-asap

（过时的）设置电脑玩家是否不顾一切地尽可能快地升级到工具时代。

[有效范围] 0~1 [默认值] -1 [编号] 181

## sn-use-by-type-max-gathering

设置允许采集者重新分配任务前，是否必须有符合逻辑的特定类型的采集者采集了指定数量的资源。

[有效范围] 0~1 [默认值] 0 [编号] 203

## sn-wall-targeting-mode

设为1，允许军队自动攻击附近的城墙和城门；设为0，则部队会忽略城墙城门。

[有效范围] 0~1 [默认值] 0 [编号] 262

## sn-warship-targeting-mode

设置战船的寻敌模式。1表示根据战船编组中射程最短者寻找防御工事目标；2表示不根据射程寻找目标；0表示根据战船编组中射程最长者寻找防御工事目标。

[有效范围] 0~2 [默认值] 0 [编号] 283

## sn-wood-dropsite-distance

设置电脑玩家愿意采集木材的最大距离。

[有效范围] 3~255 [默认值] 3 [编号] 164

## sn-wood-gatherer-percentage

设置木材采集者数量的参数，用于以下公式：木材采集者数量 = (sn-wood-gatherer-percentage + sn-wood-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 120

## sn-wood-modifier-percentage

设置木材采集者数量的参数，用于以下公式：木材采集者数量 = (sn-wood-gatherer-percentage + sn-wood-modifier-percentage) \* 资源采集者数量 \* 0.01 + 0.5。

[有效范围] -100~100 [默认值] 0 [编号] 157

## sn-zero-priority-distance

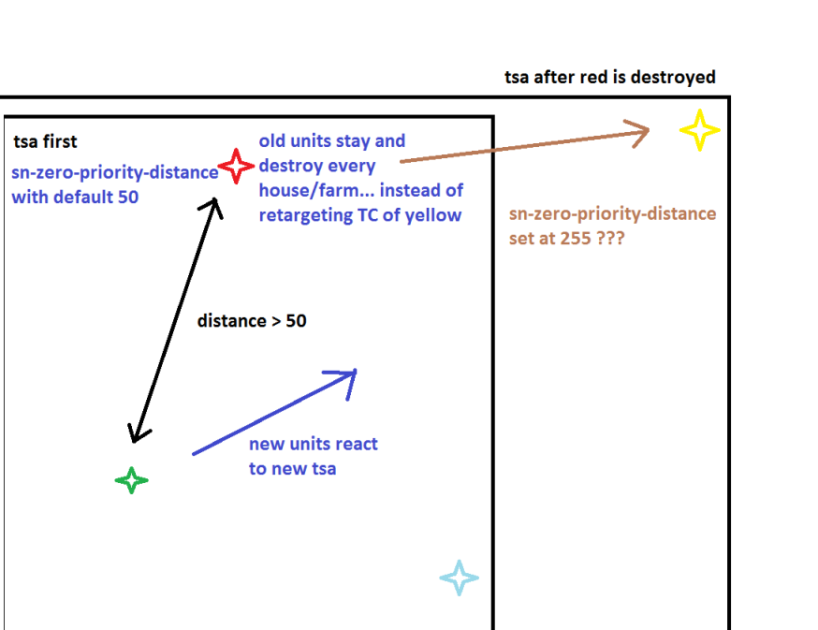
可使得在这个距离外电脑玩家给单位下达的命令都只具有最低的0优先级。

比如使用默认值50时，在主城镇中心50格外的部队将不会依据防御优先级来重新索敌 (使用TSA时) ，而是只会攻击视野内的敌人，或者由敌视回应机制来帮助附近友军反击，或者遵循DUC中的指派、巡逻、警戒等行动指令；

另外，这个sn貌似也会导致电脑的部队在攻陷敌人主TC后，热衷于清理50格范围内的剩余建筑和单位，而不是大举进攻50格外的另一个敌人；

或许当本sn设为255后，AI会在攻陷一个玩家后，快速切换去进攻另一个玩家。

上述结论来自Discord – AI Scripter频道上@interlepus的测试，于2020.6.21，下图是他的示意图，其中四角星代表不同玩家的主TC。



[有效范围] 0~255 [默认值] 50 [编号] 34

# 随机地图脚本(rms)参考

## ai\_info\_map\_type

位于节

<PLAYER\_SETUP>

语法

ai\_info\_map\_type (in)(const)地图名称 (in)(const)是否游牧 (in)(const)是否黑森林 (in)(const)任务栏显示

功能

为AI脚本设置地图类型和多个地图特征。

(in)(const)地图名称 用于让AI快速鉴别你的rms脚本。

(in)(const)是否游牧 设为1可启用UP\_NOMAD\_STYLE标志。

(in)(const)是否黑森林 设为1可启用UP\_MICHI\_STYLE标志。

(in)(const)任务栏显示 设为1可在任务栏中显示[地图名称]。

示例

1. AI中#load-if-defined COASTAL-MAP将为真。玩家将会在任务栏里看到“Coastal”（沿海）。

<PLAYER\_SETUP>

random\_placement

ai\_info\_map\_type COASTAL 0 0 0

2. AI中#load-if-defined MONGOLIA-MAPE和UP-NOMAD-STYLE将为真。

<PLAYER\_SETUP>

random\_placement

ai\_info\_map\_type MONGOLIA 1 0 0

3. AI中#load-if-defined BLACK-FOREST-MAP和UP-MICHI-STYLE将为真。

<PLAYER\_SETUP>

random\_placement

ai\_info\_map\_type BLACK\_FOREST 0 1 0

## assign\_to

位于节

<LAND\_GENERATION>:: create\_land

语法

assign\_to (in)(const)分配类型 (in)(const)数字 (in)(const)模式 (in)(const)标志

功能

将陆地分配给一个玩家编号、颜色、或队伍。

这个命令用于辅助direct\_placement。

(in)(const)分配类型 可使用AT\_PLAYER、AT\_COLOR、AT\_TEAM。

使用AT\_PLAYER或AT\_COLOR时，(in)(const)数字 可以设为1~8。

使用AT\_TEAM时：

(in)(const)数字 可以设为0~4以把陆地分配给特定的队伍（0为无队伍），设为负数可分配给队伍外的任一玩家，或设为-1分配给任一玩家。

(in)(const)模式 设为0为随机选择，设为-1为有序选择。

(in)(const)标志 可取以下值：

1 = 重置在开始之前就已被分配好的玩家；

2 = 不要记住分配这个玩家(?)；

3 = 1+2的组合。

示例

1. 在地图中心给队伍#1的随机玩家分配一块新的陆地。

<PLAYER\_SETUP>

direct\_placement /\*replaces random\_placement\*/

<LAND\_GENERATION>

create\_land

{

terrain\_type GRASS

number\_of\_tiles 128

land\_position 50 50

assign\_to AT\_TEAM 1 0 0

}

## base\_elevation

位于节

<LAND\_GENERATION>:: create\_player\_lands, create\_land

语法

base\_elevation (in)(const)基础海拔

功能

修改玩家陆地和标准陆地的基础海拔，海拔范围为1~7。若使用本语句，那么<ELEVATION\_GENERATION>节必须存在。

示例

1. 创建一块海拔为2的棕榈沙漠。

<LAND\_GENERATION>

create\_land

{

terrain\_type PALM\_DESERT

number\_of\_tiles 128

base\_elevation 2

}

<ELEVATION\_GENERATION>

## direct\_placement

位于节

< PLAYER\_SETUP >

语法

direct\_placement

功能

设置这个标志后，你可以通过assign\_to\_player和land\_position在create\_land命令里直接在地图上放置玩家。使用本命令后，!P 将会附加到任务栏窗口里的地图名称里。

示例

1. 直接将玩家1放置到地图中间 (50%, 50%)。

<PLAYER\_SETUP>

direct\_placement

<LAND\_GENERATION>

base\_terrain FOREST

create\_land

{

terrain\_type DESERT

land\_percent 3

land\_position 50 50

assign\_to\_player 1

}

## effect\_amount

位于节

< PLAYER\_SETUP >

语法

effect\_amount (in)(const)科技效果 (in)(const)科技对象 (in)(const)科技属性 (in)(const)整数值

功能

给所有玩家应用一个科技型效果。

你可能需要用 #const 来定义额外的条目名称。

(in)(const)科技对象 可以是已命名的物件或资源常数，是该命令作用于的对象。当修改物件时，你可能需要给它的所有变种逐个指定效果。

请考虑游戏中的物件升级，以避免升级造成单位最大生命值超过32768而死亡。

如果你通过这条指令禁用物件，可能会被游戏中的科技或时代重新启用。科技树也同样有可能覆盖你的修改。

使用本命令后，!C 将会附加到任务栏窗口里的地图名称里。

(in)(const)科技效果、(in)(const)科技属性 等参数，可见[UserPatchConst.rms.txt](https://www.hawkaoe.net/bbs/thread-145102-1-1.html)。

示例

1. 给城堡增加1000点生命值。

#const SET\_ATTRIBUTE 0

#const ATTR\_HITPOINTS 0

<PLAYER\_SETUP>

random\_placement

effect\_amount SET\_ATTRIBUTE CASTLE ATTR\_HITPOINTS 1000

## effect\_percent

位于节

< PLAYER\_SETUP >

语法

effect\_percent (in)(const)科技效果 (in)(const)科技对象 (in)(const)科技属性 (in)(const)百分数值

功能

给所有玩家应用一个科技型效果。这个命令与effect\_amount基本相同，不同的是这里的百分数值会先除100再投入使用。

示例

1. 给所有村民增加0.3点移动速度。

#const ADD\_ATTRIBUTE 4

#const ATTR\_SPEED 5

#const VILLAGER\_CLASS 904

<PLAYER\_SETUP>

random\_placement

effect\_percent ADD\_ATTRIBUTE VILLAGER\_CLASS ATTR\_SPEED 30 /\* 0.3 = 30/100 \*/

## grouped\_by\_team

位于节

<PLAYER\_SETUP>

语法

grouped\_by\_team

功能

使同一组队的玩家基地靠近着放置。本命令与random\_placement互斥。create\_player\_lands中的base\_size指定了组队内玩家之间的距离。使用本命令后，AI中UP-GROUPED-BY-TEAM标志将被defined。

示例

1. 启用组队玩家集中功能。

<PLAYER\_SETUP>

grouped\_by\_team /\*替代random\_placement\*/

## guard\_state

位于节

< PLAYER\_SETUP >

语法

guard\_state (in)(const)单位编号 (in)(const)资源编号 (in)(const)资源增量 (in)(const)保卫标志位

功能

给游戏设置保卫状态属性。

(in)(const)单位编号 会跟随基础单位的升级。

根据需要把以下标志位加起来以创建 (in)(const)保卫标志位：

1 = guard-flag-victory；

2 = guard-flag-resource；

4 = guard-flag-inverse。

比如3 (1+2) 代表同时附加guard-flag-victory和guard-flag-resource。

如果附加了guard-flag-resource，且[单位编号]对应的物件仍留存，那么[资源增量]/100的结果将缓慢添加到[资源编号]里；

如果附加了guard-flag-resource和guard-flag-inverse，那么仅当已没有[单位编号]对应的物件留存时，资源才会被增加；

如果附加了guard-flag-victory，那么当已经没有[单位编号]对应的物件留存后，AI将被打败。

如果你想给村民设置保卫状态，请使用VILLAGER\_CLASS而非VILLAGER。

使用本命令后，!G 将会附加到任务栏窗口里的地图名称里，且会附带保卫状态的详细设定。

示例

1. 通过设置保卫标志位为3 (guard-flag-victory和guard-flag-resource叠加) 来实现：当有村民留存时玩家可以获得圣物式的黄金细流，且如果已没有一个村民则该玩家将直接判负。

<PLAYER\_SETUP>

random\_placement

guard\_state VILLAGER\_CLASS AMOUNT\_GOLD 10 3

## nomad\_resources

位于节

<PLAYER\_SETUP>

语法

nomad\_resources

功能

在游牧地图使用，为初始资源增加一份城镇中心(275木材,100石料)的资源。使用本命令后，AI中UP-NOMAD-RESOURCES标志将被defined。

示例

1. 使初始资源增加275木材，100石料。

<PLAYER\_SETUP>

random\_placement

nomad\_resources

## resource\_delta

位于节

<OBJECTS\_GENERATION>:: create\_object

语法

resource\_delta (in)(const)资源增量

功能

修改某个特定物件包含的资源量。

示例

1. 使某一块金矿所含黄金减少100。

<OBJECTS\_GENERATION>

create\_object GOLD

{

number\_of\_objects 7

resource\_delta -100

}

## rnd(Min,Max)

位于节

<ANYWHERE>

语法

rnd(Min,Max)

功能

将脚本里的数值型参数随机化。

这可以用于替换任何数值型参数，生效时会随机转变为一个介于(包含首尾)Min和Max之间的值。Max必须大于Min。

rnd指令内不允许有空格。

为了确保兼容性，你可以检查UP\_AVAILABLE和UP\_EXTENSION这两个rms标志。

示例

1. 给7个金矿随机调整黄金储量。

<OBJECTS\_GENERATION>

create\_object GOLD

{

number\_of\_objects 7

resource\_delta rnd(-200,300)

}

## terrain\_state

位于节

<PLAYER\_SETUP>

语法

terrain\_state (in)(const)模式 (in)(const)参数1 (in)(const)参数2 (in)(const)值

功能

给游戏设置多种地形属性。

你可以在 [模式] = 0时，给 [值] 添加以下标志位：

标志位1 = 启用浅滩地形建造，启用后，诸如树木、黄金、石头和果树丛等资源物件也将可以出现在浅滩地形上，其原理是将地形限制#4、#8、#10和#11里浅滩地形的可通行性/可建造性(accessibility)从0.0改为1.0；

标志位2 = 获得更窄的浅滩/沙滩混合带(thinner shallow/beach blending)，其原理是将浅滩地形的渗透等级改为111；

标志位4 = 使冰无法融合(alternate ice blending)，其原理是将冰地形的渗透形式改为4。

示例

1. 启用可建造的浅滩地形。

<PLAYER\_SETUP>

random\_placement

terrain\_state 0 0 0 1

2. 启用可建造的浅滩地形，且使浅滩/沙滩的混合带更窄。

<PLAYER\_SETUP>

random\_placement

terrain\_state 0 0 0 3

## weather\_type

位于节

<PLAYER\_SETUP>

语法

weather\_type (in)(const)天气 (in)(const)生态颜色 (in)(const)战雾颜色 (in)(const)水流方向

功能

改变地图的天气、光影、水流。

[天气] 设为0，则没有天气，2为雨，3为雷暴，4为雪，对应的负值为反向。

[生态颜色] 应为征服者色盘的256颜色索引值，范围0~255。

[战雾颜色] 同上。

[水流方向] 设为0，则取随机方向，1为向右，-1为向左。

示例

1. 启用向东的雷暴。

<PLAYER\_SETUP>

random\_placement

weather\_type -3 0 0 0

作者：yty

时间：2013年8月13日

UP1.5修订：newtonerdai

初版时间：2020年6月15日

翔鹰帝国论坛：[www.hawkaoe.net](http://www.hawkaoe.net)

发布地址：[UserPatch 1.5 脚本编写参考](http://www.hawkaoe.net/bbs/thread-146439-1-1.html" \o "【可用于检查本文是否有新版本】)