${numeric:xxx} = 将导致 "xxx" 被评估为数字表达式，然后替换

${string:xxx} = 将导致 "xxx" 被评估为字符串表达式，然后被替换

^ = 功率

% = 余数（对于模数，请参见数字函数 mod）

: = 除法

/ =除法

\* = 乘法

+ = 添加

- =减法

> = 大于

≥ = 大于或等于

< = 小于

≤ = 小于或等于

= = 等于

≠ = 不等于

圆周率 = 3.1415...

pi2 = 6.2831... (pi \* 2)

pi05 = 1.5707... (pi / 2)

pi025 = 0.7853... (pi / 4)

pi0125 = 0.3926... (pi / 8)

pitorad = 57.2957...（用于弧度/度数转换）

piofrad = 0.0174...（用于弧度/度数转换）

e = 2.7182...（自然对数的底数）

phi = 1.6180...（黄金比例）

major = 0.6180...（与下一个值的斐波那契比率）

minor = 0.3819...（斐波那契比率与第二个下一个值）

TelestoEndpoint = 本地主机

TelestoPort = 51323

OBSWebsocketEndpoint = localhost

OBSWebsocketPort = 4455

TriggernometryVersionMajor = 1

TriggernometryVersionMinor = 2

TriggernometryVersionBuild = 3

TriggernometryVersionRevision = 4

abs(x) = x 的绝对值

cos(x) = x 的余弦

cosh(x) = x 的双曲余弦

arccos(x) = x 的反余弦

cosec(x) = x 的余割

sin(x) = x 的正弦

sinh(x) = x 的双曲正弦

arcsin(x) = x 的反正弦

sec(x) = x 的正割

tan(x) = x 的正切

tanh(x) = x 的双曲正切

arctan(x) = x 的反正切

arctan2(x, y) = x 和 y 的反正切

cotan(x) = x 的余切

distance(x1, y1, x2, y2) = 两个给定点之间的距离

radtodeg(x) = 将弧度转换为度数

degtorad(x) = 将度数转换为弧度

max(...) = 最大的指定数字

min(...) = 指定数字中的最小值

random(x, y) = 大于或等于 x 且小于 y 的随机整数 (x >= ... < y)

sqrt(x) = x 的平方根

root(x, y) = x 的第 y 个根（root(25, 2) = 25 的平方根，root(125, 3) = 125 的立方根）

rem(x, y) = x 和 y 除法的余数（IEEE 标准余数，与模数不同）

pow(x, y) = x 的 y 次方

exp(x) = e 的 x 次方

log(x) = x 的自然（以 e 为底）对数

log(x, y) = x 的以 y 为底的对数

round(x) = 将 x 舍入到最接近的整数

round(x, y) = 将 x 舍入为 y 位小数

floor(x) = 向下舍入前一个整数

ceiling(x) = 向上舍入下一个整数

sign(x) = 返回 x 的符号（-1 = x 为负，0 = x 为零，1 = x 为正）

hex2dec(x) = 将 base16（十六进制）数字转换为 base10（十进制）

hex2float(x) = 将 base16（十六进制）数转换为单精度浮点数

hex2double(x) = 将 base16（十六进制）数转换为双精度浮点数

or(...) = 如果任何指定数字不为零则返回 1，否则返回 0

and(...) = 如果所有指定数字都不为零则返回 1，否则返回 0

if(x,y,z) = 如果 x 非零则返回 y，如果 x 为零则返回 z（例如，if (2>3,4,5) 将返回 5

mod(x, y) = x 除以 y 的模数

toupper = 将给定值更改为大写

tolower = 将给定值更改为小写

length = 返回值的长度

dec2hex = 将给定的 10 进制值转换为它的 16 进制表示

dec2hex2 = 将给定的 10 进制值转换为其 16 进制表示（输出 2 个字节）

dec2hex4 = 将给定的 10 进制值转换为其 16 进制表示（输出 4 个字节）

dec2hex8 = 将给定的 10 进制值转换为其 16 进制表示（输出 8 个字节）

float2hex = 将单精度浮点数转换为其 32 位 base 16 表示

double2hex = 将双精度浮点数转换为其 64 位 base 16 表示

padleft(character code, length) = 用字符代码定义的字符从左侧填充值到给定长度

padright(character code, length) = 用字符代码定义的字符从右侧填充值到给定长度

substring(start) = 返回从给定索引开始的值的子字符串

substring(start, length) = 返回从给定索引开始并包含最多给定字符长度的值的子字符串

indexof(string) = 返回给定值内搜索字符串的索引

lastindexof(string) = 返回给定值内搜索字符串的最后一个索引

trim = 去除值两边的空格

trim(character code, character code, ...) = 从值的两边修剪给定的字符

trimleft = 从值的左侧修剪空白

trimleft(character code, character code, ...) = 从值的左侧修剪给定的字符

trimright = 从值的右侧修剪空白

trimright(character code, character code, ...) = 从值的右侧修剪给定的字符

format(type, format string) = 将值格式化为 .Net 类型和指定的格式字符串，例如：${func:format(System.Double,"0.0"):1213131}

compare(stringtocompare) = 区分大小写的字符串比较（-1 = 按字母顺序排列的字符串较低，0 = 字符串相等，1 = 字符串较高）

compare(stringtocompare, ignorecase) = case (in)sensitive string compare (-1 = string lower in alphabetical order, 0 = strings equal, 1 = string higher)

utctime = 将 UNIX 时间戳转换为 UTC 时间的默认日期/时间字符串格式

utctime(format) = 将 UNIX 时间戳转换为给定的 UTC 时间日期/时间字符串格式

localtime = 将 UNIX 时间戳转换为本地时间的默认日期/时间字符串格式

localtime(format) = 将 UNIX 时间戳转换为您当地时间的给定日期/时间字符串格式

${xxx} = 检索名为捕获组 xxx 的正则表达式的值，如果 xxx 是数字且不存在具有相同名称的命名组，则检索第 n 个捕获组

${numeric:xxx} = 将 xxx 计算为数值表达式

${string:xxx} = 将 xxx 计算为字符串表达式

${var:xxx} = 检索变量 xxx 值

${evar:xxx} = 检查变量 xxx 是否存在（1 = 存在，0 = 不存在）

${lvar:xxx.size} = 返回列表变量 xxx 的大小

${lvar:xxx.indexof(hello warudo)} = 搜索给定的字符串（在本例中为“hello warudo”），并返回找到的第一个实例的索引号，如果未找到则返回零

${lvar:xxx.lastindexof(hello warudo)} = 搜索给定的字符串（在本例中为“hello warudo”），并返回找到的最后一个实例的索引号，如果未找到则返回零

${lvar:xxx[1]} = 检索列表变量 xxx 的第一个索引中的值

${lvar:xxx[2]} = 检索列表变量 xxx 的第二个索引中的值

${lvar:xxx[last]} = 检索列表变量 xxx 的最后一个索引中的值

${elvar:xxx} = 检查列表变量 xxx 是否存在（1 = 存在，0 = 不存在）

${tvar:xxx.width} = 返回表格变量 xxx 的宽度（列数）

${tvar:xxx.height} = 返回表变量 xxx 的高度（行数）

${tvar:xxx[2][4]} = 检索表变量 xxx 第 4 行第 2 列中的值

${tvar:xxx[last][2]} = 检索表变量 xxx 第 2 行最后一列中的值

${tvar:xxx[2][last]} = 检索表变量 xxx 第 2 列最后一行的值

${tvarcl:xxx[My Column Name][3]} = 执行基于列的查找（第一行是标题行），并检索该列第三行的值

${tvarrl:xxx[My Row Name][2]} = 执行基于行的查找（第一列是标题列），并检索该行第二列的值

${etvar:xxx} = 检查表变量 xxx 是否存在（1 = 存在，0 = 不存在）

${func:xxx(yyy):zzz} = 在值 zzz 上执行函数 xxx（带有参数 yyy，如果需要的话）

${pvar:xxx} = 检索持久变量 xxx 值

${epvar:xxx} = 检查持久变量 xxx 是否存在（1 = 存在，0 = 不存在）

${plvar:xxx.size} = 返回持久列表变量 xxx 的大小

${plvar:xxx.indexof(hello warudo)} = 搜索给定的字符串（在本例中为“hello warudo”），并返回找到的第一个实例的索引号，如果找不到则返回零

${plvar:xxx.lastindexof(hello warudo)} = 搜索给定的字符串（在此示例中为“hello warudo”），并返回找到的最后一个实例的索引号，如果未找到则返回零

${plvar:xxx[1]} = 检索持久列表变量 xxx 的第一个索引中的值

${plvar:xxx[2]} = 检索持久列表变量 xxx 的第二个索引中的值

${plvar:xxx[last]} = 检索持久列表变量 xxx 的最后一个索引中的值

${eplvar:xxx} = 检查持久列表变量 xxx 是否存在（1 = 存在，0 = 不存在）

${ptvar:xxx.width} = 返回持久表变量 xxx 的宽度（列数）

${ptvar:xxx.height} = 返回持久表变量 xxx 的高度（行数）

${ptvar:xxx[2][4]} = 检索持久表变量 xxx 第 4 行第 2 列中的值

${ptvar:xxx[last][2]} = 检索持久表变量 xxx 第 2 行最后一列中的值

${ptvar:xxx[2][last]} = 检索持久表变量 xxx 第 2 列最后一行的值

${ptvarcl:xxx[My Column Name][3]} = 执行基于列的查找（第一行是标题行），并检索该列第三行的值

${ptvarrl:xxx[My Row Name][2]} = 执行基于行的查找（第一列是标题列），并检索该行第二列的值

${eptvar:xxx} = 检查持久表变量 xxx 是否存在（1 = 存在，0 = 不存在）

\_duration = ACT 中当前遭遇的持续时间（以秒为单位），如果没有主动遭遇则返回 0

\_event = 触发触发器的完整事件文本（仅在特定触发器的上下文中有效）

\_incombat = 标志确定是否有一个活跃的战斗遭遇（0 = 不在战斗中，1 = 当前在战斗中）

\_since = 自触发触发器以来的秒数（仅在该特定触发器的上下文中有效）

\_sincems = 自触发触发器以来的毫秒数（仅在该特定触发器的上下文中有效）

\_triggerid = 触发器的唯一 ID（仅在特定触发器的上下文中有效）

\_triggername = 触发器的名称（仅在特定触发器的上下文中有效）

\_timestamp = 触发器触发时刻的 UNIX 时间戳（仅在特定触发器的上下文中有效）

\_timestampms = 触发器触发时刻的 UNIX 时间戳（仅在特定触发器的上下文中有效），以毫秒为单位

\_systemtime = 当前系统时间的 UNIX 时间戳

\_systemtimems = 当前系统时间的 UNIX 时间戳，以毫秒为单位

\_zone = 触发触发器的区域名称（仅在特定触发器的上下文中有效）

\_response = 从远程端点收到的响应（仅在该特定操作的上下文中有效）

\_jsonresponse[path] = 从远程端点收到的响应（仅在该特定操作的上下文中有效），解析为 JSON（例如，您可以执行 \_jsonresponse[firstnode/secondnode/myitem]）

\_screenwidth = 主监视器的宽度（以像素为单位）

\_screenheight = 主监视器的高度（以像素为单位）

\_lastencounter = 上次相遇的 ACT DPS 信息，采用默认的 miniparse 格式

\_activeencounter = 来自正在进行的默认 miniparse 格式的 ACT DPS 信息

\_env[varname] = 检索指定环境变量的值

\_const[varname] = 检索在 Triggernometry 配置中指定的常量值

\_responsecode = 由触发器触发上下文中的最后一个 JSON 操作填充的 HTTP 状态代码（n = 状态代码，如果未执行操作则为 0，失败时为 -1）

\_loopiterator = 循环经历的迭代次数，从零开始（= 第一次运行时为零）

\_x = 当前 X 位置

\_y = 当前 Y 位置

\_w 或 \_width = 当前宽度

\_h 或 \_height = 当前高度

\_opacity = 当前不透明度，单位为 % (0 ... 100)

\_textaura[id].property = 允许您访问具有给定 id 的文本光环的属性（可用属性：x、y、w、h、不透明度、文本）

\_imageaura[id].property = 允许您访问具有给定 id 的图像光环的属性（可用属性：x、y、w、h、不透明度）

\_ffxivparty = 一个特殊的数组类型变量，代表当前党员（最多八名玩家）

\_ffxiventity = 一个特殊的数组类型变量，代表游戏世界中所有当前已知的实体

\_ffxivplayer = 您当前登录角色名称的简写，由 \_ffxivparty 取代，但仍保持向后兼容性

\_ffxivtime = 以分钟为单位为您提供当前的 Eorzean 时间（0 = 午夜，720 = 中午）

\_ffxivpartyorder = 提供一些关于当前派对顺序的调试信息

\_ffxivprocid = FFXIV ACT 插件正在侦听的实例的进程 ID

\_ffxivprocname = FFXIV ACT 插件正在侦听的实例的进程名称

\_ffxivzoneid = 当前游戏区域的 ID

\_ffxivparty[1] = 列表中的第一个党员

\_ffxiventity[1] = 战斗人员列表中的第一个演员

\_ffxivparty[我的名字] = 同名的党员

\_ffxiventity[Titan] = 战斗员列表中具有给定名称的演员（如果存在多个，则返回找到的第一个）

\_ffxivparty[1234ABCD] = 给定ID的党员

\_ffxiventity[1234ABCD] = 在战斗员列表中具有给定 ID 的演员

name = 演员的名字

job = 当前工作的三个字母首字母缩写词（AST、CUL、MIN、SMN 等）

jobid = 当前工作的数字（内部游戏）表示

currenthp, currentmp, currentcp, currentgp = 当前 HP、MP、CP 和 GP

maxhp, maxmp, maxcp, maxgp = 最大 HP、MP、CP 和 GP

level = 演员的等级

x、y 和 z = 演员的位置信息

inparty = 党员身份标志（0 = 角色不存在或不在党内，1 = 角色存在且在党内）

id = 演员的十六进制 ID

order = \_ffxivparty 结构中参与者的订单号（对于 \_ffxiventity，此属性为 0）

worldid = 角色家乡世界的数字ID

worldname = 角色所在世界的名称

currentworldid = 角色当前世界的数字ID

heading = 实体的弧度方向（South 0, North ±PI）

targetid = 当前被选为目标的 actor 的十六进制 ID（0 表示没有选择目标）

casttargetid = 当前演员表所针对的演员的十六进制 ID（0 表示没有正在进行的演员表）

distance = 从主要战斗员（你）到实体的距离

role = 演员的角色（“坦克”、“治疗者”、“DPS”、“工匠”、“采集者”，如果不可用则留空）