ALIBABA CLOUD

阿里云

CDN

边缘脚本

文档版本：20220516

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

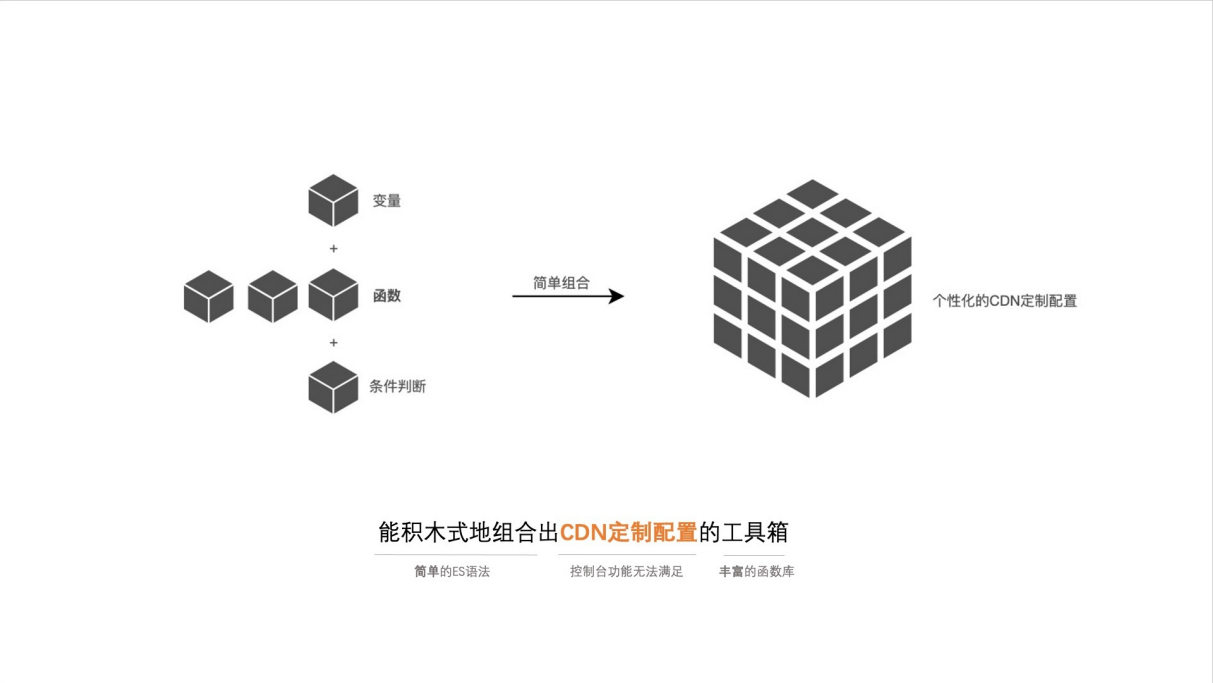
目录

1.概述 05
2.EdgeScript原理介绍 07
3.EdgeScript快速入门 09
3.1. 通过控制台使用EdgeScript 09
3.2. 通过CLI工具使用边缘脚本 12
4.EdgeScript语法 15
5.EdgeScript内置变量表 19
6.EdgeScript内置函数库 21
6.1. 概述 21
6.2. 条件判断相关 21
6.3. 数字类型相关 25
6.4. 字符串类型相关函数 35
6.5. 字典类型相关 46
6.6. 请求处理相关 51
6.7. 限速相关函数 60
6.8. 缓存相关 62
6.9. 时间相关 63
6.10. 密码算法相关 68
6.11. JSON相关 76
6.12. Misc相关 77
6.13. 数组类型相关函数 82
6.14. 请求判断相关函数 85
6.15. 附录 111
7.EdgeScript场景示例 128
8.EdgeScript实践分享 133
9.示例：通过CLI工具使用边缘脚本 134

1.概述

边缘脚本（EdgeScript，简称ES）是一个可供您快速实现CDN定制配置的工具箱，当CDN控制台上的标准配置无法满足您的业务需求时，可以尝试使用边缘脚本简单编程实现。

边缘脚本通过简单易学的语法和庞大的函数库，能够像积木式地组合出个性化的CDN定制配置。



边缘脚本内置了CDN节点可以识别的变量、简单的判断语句，同时提供了大量阿里云CDN封装好的函数供您直接调用。通过简单的变量判断并调用现成的函数，即可满足您绝大部分定制的 鉴权、缓存、限速、请求头增减等定制配置需求。可以有效地帮您解决定制化配置需求无法实现、业务变更不敏捷的问题。

- 变量信息，请参见EdgeScript内置变量表。

- 函数信息，请参见EdgeScript内置函数库。
- 条件判断，请参见条件判断相关。

使用费用

边缘脚本功能目前不收费。

应用场景

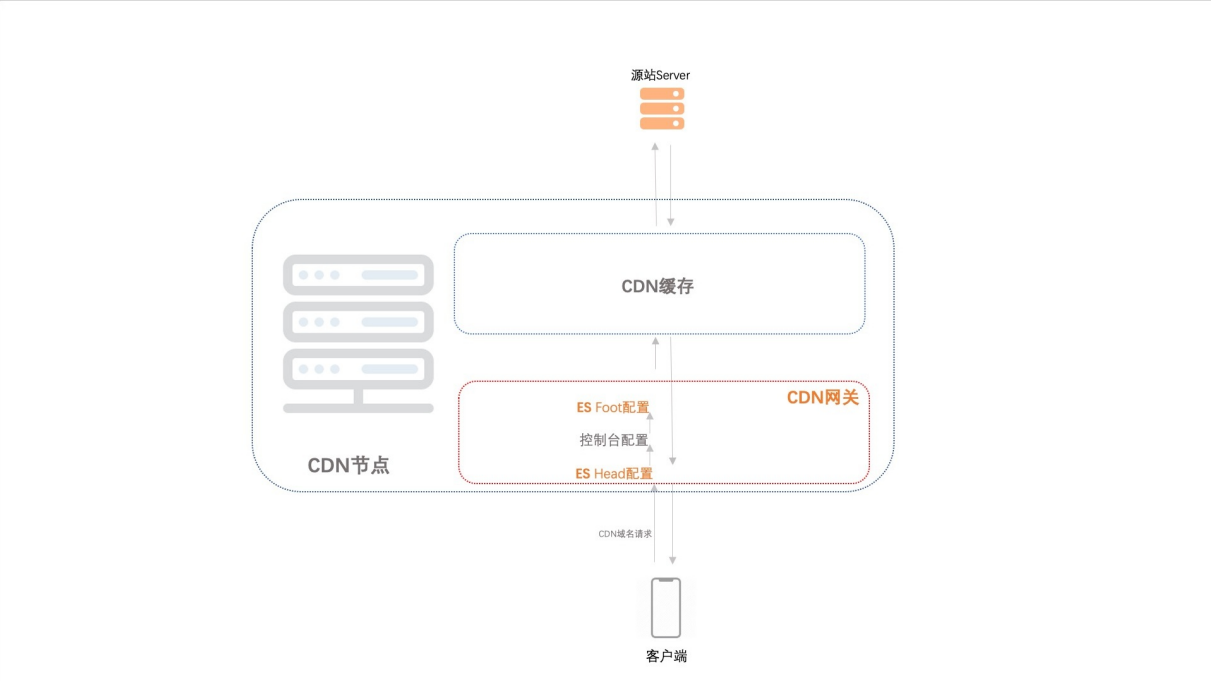
场景	描述
定制化鉴权逻辑	鉴权场景在视频点播、视频直播中的防盗链需求中十分普遍。基于请求参数、Cookie或其他复杂算法等各类鉴权需求，帮您快速实现鉴权需求，从而完成对资源的保护。
请求头&响应头控制	可以使用边缘脚本对请求参数、请求头等变量进行灵活修改。
改写&重定向	多应用在多语言版本的网站之上，比如中文网站可能会302重定向到1个位置，英文网站或者德文网站，可能会302重定向到不同的位置。
A/B Test	当源站上一个新功能时，可能会有A/B Test的需求，很可能需要CDN侧做支持。通过CDN携带不同的回源请求头或不同的URL，去区分触发源站的不同功能，从而来实现这个全链路的A/B Test。
缓存控制	在某些自定义业务场景下，标准功能的缓存时长或者缓存策略不能够满足需求时，可以通过边缘脚本去完成缓存业务定制化。
限速控制	当需要对免费客户和付费客户进行限速的区分。可以通过边缘脚本来实现。
封禁拦截	针对某些地区、某些特殊的逻辑或者特殊的客户端IP去进行封禁拦截，都可以通过边缘脚本来实现定制化。同时，还可以针对防爬策略的实现，防止爬虫爬取资源。

2.EdgeScript原理介绍

本文为您介绍EdgeScript的运行原理、规则模型、执行位置与优先级、命中与中断执行。

运行原理

您配置的边缘脚本规则与CDN控制台上的标准配置一样，都是对CDN请求进行处理。边缘脚本的执行位置如图所示，当客户端请求到达CDN节点后，CDN节点网关会根据您在控制台上设置的标准配置、边缘脚本规则对请求进行处理。以CDN控制台上的标准配置为参照物，边缘脚本可选择在请求处理的最前面或最后面生效。



规则模型

EdgeScript的规则模型如下：

- EdgeScript的规则模型的核心出发点是将不同业务功能隔离至不同规则，以及控制规则的执行流。
- EdgeScript的规则模型中的每条规则可以各自选择规则的执行位置（即介入请求处理的位置）和优先级。
- EdgeScript的规则模型以域名为度。

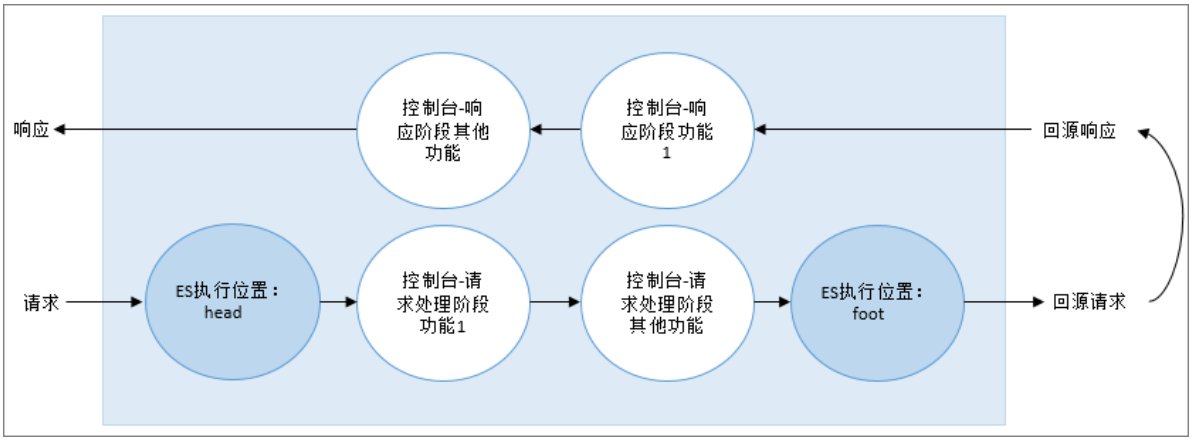
执行位置与优先级

EdgeScript的执行位置与优先级如下：

- 执行位置

  EdgeScript规则的执行位置为请求处理开始和请求处理结尾。

  - 请求处理开始：常用应用场景为一次鉴权、拦截、限速等。
  - 请求处理结束：常用应用场景为缓存设置、回源鉴权、AB测试等。



- 优先级

  处于同一执行位置的多条规则，可以通过选择优先级，决定其先后执行顺序。

命中与中断执行

EdgeScript的命中与中断执行如下：

- EdgeScript规则的命中定义
  - 当规则结尾是以 `return true` 返回时：规则模型认为该规则命中。
  - 当规则结尾是以 `return false` 返回时：规则模型认为该规则未命中。
- EdgeScript规则的中断执行

  对于命中规则，可以选择终止本阶段剩余规则的执行。

视频讲解

本视频主要从以下几个方面为您介绍EdgeScript：

1. EdgeScript的用户需求场景（业务复杂，通用功能配置无法满足，有定制配置需求）。
2. 阿里云CDN提供的三种不同灵活度、不同使用难度的配置功能（EdgeScript的配置灵活度最高，但是上手难度会相对大一些）。
3. 什么是EdgeScript？
4. EdgeScript提供的能力。
5. EdgeScript的技术原理。
6. EdgeScript的使用场景。

3.EdgeScript快速入门

3.1. 通过控制台使用EdgeScript

在CDN控制台上，您可以根据边缘脚本定义的代码规则，创建边缘脚本规则，并发布到生产环境，实现对CDN产品的定制化管理。本文为您介绍在CDN控制台上，如何配置使用边缘脚本。

配置流程

1. 登录CDN控制台。
2. 在左侧导航栏，单击**域名管理**。
3. 在**域名管理**页面，单击目标域名对应的**管理**。
4. 单击**EdgeScript自定义策略**。
5. 在模拟环境中添加规则。
   i. 在**模拟环境**页签，单击**添加规则**，配置边缘脚本规则。

> **说明** 目前单个域名仅支持添加一条边缘脚本规则。如果您需要添加多条规则，请提交工单申请。

配置信息及说明，请参见下表。

配置信息	是否必填	说明
规则名称	是	规则名称。仅支持英文字母、数字和下划线（_）。
规则代码	是	规则代码。 ■ 您可以直接输入规则代码，也可以单击**使用代码模板**，选择对应的场景代码模板。 ■ 您可以按照使用场景编写规则代码。更多详情，请参见EdgeScript场景示例。
优先级	是	规则优先级。取值范围0~999，数值越大，优先级越高。

配置信息	是否必填	说明
执行位置	是	规则执行位置。更多详情，请参见执行位置与优先级。
启用状态	是	规则启用状态。
Break	否	本规则命中情况下，是否终止本阶段剩余规则的执行。开启后，匹配该规则，停止执行其余的ESL规则。
扩展项	否	当前支持扩展。目前仅支持 `_es_dbg=signature` 字段，该字段表示开启调试模式。

ii. 单击**发布到模拟环境**。

6. 在模拟环境中，测试规则。

   边缘脚本需要在模拟环境先进行测试，测试通过后再发布到生产环境。模拟环境的测试IP可通过解析域名staging.myalicdn.com来获取，客户端绑定测试IP的host为加速域名即可测试模拟环境的EdgeScript规则。

   将模拟环境测试IP地址与加速域名的对应关系添加到*hosts*文件中，*hosts*文件的路径如下：

   - 终端为Windows系统，路径为*C:\Windows\System32\drivers\etc\hosts*。
   - 终端为MAC系统，路径为*/etc/hosts*。

7. 测试完成后，单击**发布所有规则到生产环境**，将模拟环境规则发布至生产环境。

   > **注意** 模拟环境规则发布到生产环境后，模拟环境的规则自动被清空。

8. 如果您需要基于最新发布的规则进行增改，则需要将发布到生产环境的规则同步到模拟环境后再执行操作。 单击**从生产环境复制规则**，将发布到生产环境的规则同步到模拟环境。

   生产环境的规则同步到模拟环境后，即可在模拟环境进行修改或新增规则，请参见下图。

如果您喜欢CLI命令行界面，可以通过CLI工具使用边缘脚本。更多详情，请参见通过CLI工具使用边缘脚本。

视频讲解

本视频以配置域名自定义限速策略为例，演示通过CDN控制台给域名配置EdgeScript边缘脚本的整个流程，包括添加边缘脚本、发布边缘脚本到模拟环境、测试模拟环境功能、发布边缘脚本到生产环境和测试生产环境功能。

3.2. 通过CLI工具使用边缘脚本

阿里云为您提供边缘脚本CLI工具，可以将您本地编写的边缘脚本规则发布到模拟环境进行测试，并发布至生产环境，也可以对模拟环境和生产环境的边缘脚本规则进行查询、修改和删除。本文为您介绍边缘脚本CLI工具的使用方法。

1. 下载CLI工具。更多信息，请参见下载CLI工具。
2. 配置AccessKey ID和AccessKey Secret。

```
python ./es.py config --id=AK_ID --secret=AK_SECRET
cat aliyun.ini
[Credentials]
accesskeyid = 更新后的AccessKey ID
accesskeysecret = 更新后的AccessKey Secret
```

3. 发布边缘脚本规则至模拟环境或生产环境。

```
./es.py action=push_test_env domain=<domain> rule='{"pos":"<head|foot>","pri":"0-999","
rule_path":"<the es code path>","enable":"<on|off>"}'
./es.py action=push_product_env domain=<domain> rule='{"pos":"<head|foot>","pri":"0-999
","rule_path":"<the es code path>","enable":"<on|off>","configid":"<configid>"}'
```

规则字段说明，请参见下表：

字段名称	是否必选	说明
enable	是	是否生效。取值： ○ on ○ off
pos	是	执行位置。取值： ○ head ○ foot
pri	是	执行优先级。不同执行位置的优先级各自独立。取值范围：0~999。 ○ 0表示优先级最高 ○ 999表示优先级最低。
rule	是	规则内容。

字段名称	是否必选	说明
brk	否	本规则命中情况下，是否终止本阶段剩余规则的执行。取值： ○ on ○ off
testip	否	测试客户IP。默认为空。如果配置了该参数，则只有客户端IP地址可触发本条规则执行。
option	否	扩展项。当前支持扩展。`_es_dbg=signature`用于增加调试响应头。

说明

- 新增规则不需要指定configid。
- 修改规则需要指定configid，使用查询接口可获取到对应规则的configid。
- 您可以指定多条rule。

4. 查询模拟环境或生产环境下的边缘脚本规则。

```
./es.py action=query_test_env domain=<domain>
./es.py action=query_product_env domain=<domain>
```

5. 删除模拟环境或生产环境下的边缘脚本规则。

```
./es.py action=del_test_env domain=<domain> configid=<configid>
./es.py action=del_product_env domain=<domain> configid=<configid>
```

6. 模拟环境的边缘脚本规则执行正式发布或回滚。

```
./es.py action=publish_test_env domain=<domain>
./es.py action=rollback_test_env domain=<domain>
```

基于m3u8.es规则拦截所有.m3u8请求做示例，为您详细介绍边缘规则的编写、保存、测试和发布的操作方法。具体操作，请参见示例：通过CLI工具使用边缘脚本。

实时调试方式

1. 配置实时调试。

   您可以通过控制台的WebIDE图形化操作页面进行`_es_dbg`配置，也可以使用如下命令进行配置。

```
 ./es.py action=push_test_env domain=<domain> rule='{"pos":"<head|foot>","pri":"0-999",
"rule_path":"<the es code path>","enable":"<on|off>","configid":"<configid>", "option":
"_es_dbg=123"}'
```

2. 测试请求。

   测试时请求参数中携带`_es_dbg`参数，参数值为配置的option中`_es_dbg`的值，关注如下响应头信息。

Trace信息：`X-DEBUG-ES-TRACE-RULE-{规则ID}`，查看对应规则的执行流，格式为`_行号_函数名(入参):返回值{_执行耗时}`。

4.EdgeScript语法

本文为您介绍EdgeScript语法中注释、标识符、数据类型、变量、运算符、语句和函数的使用规则。

EdgeScript语法使用规则，请参见下表：

语法	规则
注释	以#开头的当前行后续内容，均为注释。例如：# this is annotation。
标识符规则	• 由字母、数字、下划线组成，数字不能开头，区分大小写。 • 变量名（内置 、自定义）和函数名（内置 、自定义）均遵守标识符规则。
数据类型	• 字符串 字面常量：使用单引号括起来，例如：`'hello, EdgeScript'`。 • 数字 字面常量：十进制数字，例如：10、-99、1.1。 • 布尔值 字面常量：true、false。 • 字典 字面常量如下： ○ []：空 ○ `['key1', 'key2', 100]`： ▪ `1 -> 'key1'` ▪ `2 -> 'key2'` ▪ `3 -> 'key3'` ○ `['key1' = 'value1', 'key2' = 1000]` ▪ `'key1' -> 'value1'` ▪ `'key2' -> 1000`
变量	• 定义 赋值即定义。 • 使用 ○ 内置和自定义变量，均由变量名进行引用 。 ▪ 引用内置变量：`host`。 ▪ 引用自定义变量：`seckey`。 ○ 为强调变量的内置属性，可通过`$`进行引用。 引用内置变量：`$host`。 ○ 自定义变量的名称不能与内置变量同名。 内置变量，请参见EdgeScript内置变量表。

语法	规则

运算符	• = ：赋值运算符
	  ○ 例如：`seckey = 'ASDLFJ234dxvf34sDF'`
	  ○ 例如：`seckeys = ['key1', 'key2']`
	• - ：负号运算符
	  例如：`inum = -10`
	• 对各数据类型的操作，不再另行支持运算符，均由内置函数支持，请参见条件判断相关。
	  ○ 各数据类型内置函数支持
	  ■ 字符串类型内置处理函数。
	  ■ 数字类型内置处理函数。
	  ■ 字典类型内置处理函数。
	  ○ 示例
	  ■ `sval = concat(sval, 'trail')`
	  ■ `len(arrvar)`

	• 条件判断语句
	`if condition {`
	`  ...`
	`}`
	`if condition1 {`
	`  if conditon2 {`
	`  ...`
	`  }`
	`}`
	`if condition {`
	`  ...`
	`} else {`
	`  ...`
	`}`
	• 语句解释
	  ○ `condition` 可由如下语法元素组成：
	  ■ 字面值
	  ■ 变量
	  ■ 函数调用
	  ○ body部分
	  ■ 允许空body。
	  ■ 允许多语句：一行一条语句。
	  ○ 支持多层嵌套
	  ○ CodingStyle
	  语法强制要求左大扩号跟随在 `if condition` 之后，且同行。
	• for循环

语法	规则
语句	

```
a = ['a', 'b', 'c', 'd']
def for_func () {
    for k, v in a {
        if eq(v, 'c') {
            return true
        }
    }
}
for_func()
#######################################################################
#######################
a = ['a' = 1, 'b' = 2, 'c' = 3, 'd' = 4, 'e' = 5, 'f' = 6]
def for_func () {
    for k, v in a {
        if eq(k, 'c') {
            return true
        }
    }
}
for_func()
#######################################################################
#######################
num = 0
def for_func () {
    a = [0,1,2,3,4,5,6,7,8,9]
    for k ,v in a {
        b = [0,1,2,3,4,5,6,7,8,9]
        for k1 ,v1 in b {
            c = [0,1,2,3,4,5,6,7,8,9]
            for k2 ,v2 in c {
                num = add(num, 1)
                if and(eq(v, 3), eq(v1, 5), eq(v2, 7)) {
                    return true
                }
            }
        }
    }
}
for_func()
```

- 注意事项：
  - for循环只能用来遍历字典或数组类型数据。
  - 没有break等关键字，建议配合自定义函数使用，在需要的时候进行return来打破循环。
  - 支持多层嵌套。
  - CodingStyle

    语法强制要求左大扩号跟随在 `for` ...之后，且同行。

语法	规则

函数	• 定义语法
	`def` **函数名（参数列表）** `{`
	`...`
	`}`
	• 定义说明
	○ 形参列表
	▪ 允许无参。
	▪ 允许多参：由逗号分隔。
	○ 函数体部分
	▪ 允许空body。
	▪ 允许多语句：一行一条语句。
	▪ 返回值：支持return语句。
	○ CodingStyle
	语法强制要求左大扩号跟随在 `def` 函数名（参数列表）之后，且同行。
	• 函数调用
	无论内置、自定义函数，均通过 *函数名*`()` 进行调用。

其他	EdgeScript全文不允许出现任何双引号。

5.EdgeScript内置变量表

本文为您介绍EdgeScript脚本中所有内置变量的含义和对应nginx原生变量。

EdgeScript内置变量如下表所示。

内置变量名	含义	对应nginx原生变量
$arg_{name}	*Query String* 中的参数 `name` 值。`Query String` 表示HTTP请求中的请求参数。	$arg_ **说明** `{name}` 中出现的连接号（-），需要使用 `req_uri_arg` 函数来取值（不使用变量 `$arg_` ），例如：用户请求 `http://example.com/1.jpg?example-demo=123` ，取值函数式为 `req_uri_arg('example-demo')` 。
$http_{name}	请求头中的name值。	$http_ **说明** `{name}` 中出现的连接号（-），需要使用下划线（_）替代，例如： `X-USER-ID` 对应为 `$http_x_user_id` 。
$cookie_{name}	请求cookie头中的name值。	$cookie_ **说明** `{name}` 中出现的连接号（-），需要使用 `req_cookie` 函数来取值（不使用变量 `$cookie_` ），例如：用户请求 `cookie:example-demo=123` ，取值函数式为 `req_cookie('example-demo')` 。
$scheme	协议类型。	$scheme
$server_protocol	协议版本。	$server_protocol
$host	原始host。	$host
$uri	原始URI。	无

内置变量名	含义	对应nginx原生变量
$args	`$args` 表示当前HTTP请求的全部请求参数，但不包含问号（ `?` ）。例如： `http://example.aliyundoc.com/1k.file?k1=v1&k2=v2` 。 • `$arg_k1` 可以获得对应的 `v1` 值。 • `$args` 可以获得整个请求参数字符串，即 `k1=v1&k2=v2` ，不包括问号（ `?` ）。	$args
$request_method	请求方法。	$request_method
$request_uri	*uri*`+'?'+args` 的内容。	$request_uri
$remote_addr	客户的IP地址（即x-forwarded-for请求头携带的第一个IP）。	$remote_addr

说明

- 内置变量名前的美元符号（ `$` ）仅为强调其内置变量属性，删除后不影响使用。
- 内置变量不允许担当左值，即内置变量不允许被赋值。
- 每条ES规则中最多支持使用200个全局变量，局部变量不限。如果全局变量超过200个请自定义函数，并在函数中以局部变量的形式使用全局变量。

6.EdgeScript内置函数库

6.1. 概述

本文为您介绍EdgeScript内置函数的分类和各个分类包含的函数。

函数分类	函数
条件判断相关	条件判断相关函数包括：and、or、not、eq、ne、null。
数字类型相关	数字类型相关函数包括：add、sub、mul、div、mod、gt、ge、lt、le、floor、ceil。
字符串类型相关函数	字符串类型相关函数包括：substr、concat、upper、lower、len、byte、match_re、capture_re、gsub_re、split、split_as_key、tohex、tostring、tochar、reverse、find、format、tobin、trim。
字典类型相关	字典类型相关函数包括：set、get、foreach、del。
请求处理相关	请求处理相关函数包括：add_req_header、del_req_header、add_rsp_header、del_rsp_header、encode_args、decode_args、rewrite、say、print、exit、get_rsp_header、add_rsp_cookie。
限速相关函数	限速相关函数包括：limit_rate_after、limit_rate。
缓存相关	缓存相关函数为：set_cache_ttl。
时间相关	时间相关函数包括：today、time、now、localtime、utctime、cookie_time、http_time、parse_http_time、unixtime。
密码算法相关	密码算法相关函数包括：aes_new、aes_enc、aes_dec、sha1、sha2、hmac、hmac_sha1、md5、md5_bin。
JSON相关	JSON相关函数包括：json_enc、json_dec。
Misc相关	Misc相关函数包括：base64_enc、base64_dec、url_escape、url_unescape、rand、rand_hit、crc、tonumber、base64_enc_safe、base64_dec_safe、randomseed。
数组类型相关函数	数组类型相关函数包括：arr_concat、arr_insert、arr_remove、arr_sort、arr_len。
请求判断相关函数	请求判断相关函数包括：server_addr、server_port、client_addr、client_country、client_region、client_isp、ip_country、ip_region、ip_isp、req_uri、req_uri_basename、req_uri_ext、req_uri_seg、req_uri_arg、req_uri_query_string、req_scheme、req_method、req_host、req_user_agent、req_referer、req_cookie、req_first_x_forwarded、req_header、req_id。

6.2. 条件判断相关

本文为您介绍条件判断相关函数的语法、说明、参数、返回值和示例。

and

函数详细信息，请参见下表：

项目	描述
语法	`and(arg, ...)` 。
说明	• 逻辑与运算符。 • 支持短路语义，即某个参数为假时，后续参数不再进行求值。
参数	一个或多个参数，参数类型不限。
返回值	全部参数为真时返回 `true` ，任一参数为假时返回 `false` 。

示例	`if and($arg_mode, eq($arg_mode, 'set_header')) {`
	`  add_rsp_header('USER-DEFINED-1','path1')`
	`}`
	
	• 当请求携带mode参数且mode参数等于set_header时，设置响应头USER-DEFINED-1。
	• 当请求不携带mode参数，短路语义生效，不再执行后续的eq比较；由于and()为假，不会设置响应头USER-DEFINED-1。

or

函数详细信息，请参见下表：

项目	描述
语法	`or(arg, ...)` 。
说明	• 逻辑或运算符。 • 支持短路语义，即某个参数为真时，后续参数不再进行求值。
参数	一个或多个参数，参数类型不限。
返回值	任一参数为真时返回 `true` ，全部参数为假返时 `false` 。

示例	`if and($http_from, or(eq($http_from, 'wap'), eq($http_from, 'comos'))) {`
	`  rewrite(concat('http://example.com.cn/zt_d/we2015/', $http_from), 'enhance_redirect')`
	`}`
	
	• 当请求头from存在，且其值为[wap|comos]时，302重写向至http://example.com.cn/zt_d/we2015/[wap|comos]。
	• 当请求头from存在，且其值为wap时，短路语义生效，不再执行后续eq comos比较，同时or()返回true。

not

函数详细信息，请参见下表：

项目	描述
语法	`not(arg)` 。
说明	逻辑运算符取反。参数 `undef` 和 `false` 为假，其余为真。
参数	仅接受1个参数，参数类型不限。
返回值	• true • false

示例	`if not($arg_key) {`
	`  exit(403)`
	`}`
	`if not($cookie_user) {`
	`  exit(403, 'not cookie user')`
	`}`
	`if not(0) {`
	`  exit(403)`
	`}`
	`if not(false) {`
	`  exit(403)`
	`}`
	
	• 如果请求未携带参数key时，403拒绝请求。
	• 当请求未携带 `cookie user` 时，403拒绝请求，响应body为 `'not cookie user'` 。
	• `not(0)` 的结果为false。
	• `not(false)` 的结果为true。

eq

函数详细信息，请参见下表：

项目	描述
语法	`eq(arg1, arg2)` 。
说明	比较2个参数是否相等。
参数	• arg1：任意类型。 • arg2：应与 `arg1` 类型相同。
返回值	参数相等返回 `true` ，否则返回 `false` 。

项目	描述

示例	`key1 = 'value1'`
	`key2 = 'value2'`
	`if and($arg_k1, $arg_k2, eq(key1, $arg_k1), ne(key2, $arg_k2)) {`
	`  say('match condition')`
	`}`
	• 请求参数k1和k2都存在时，执行后续的比较操作。
	• 请求参数k1或k2不存在时，短路语义生效，不再执行后续的比较操作。
	• eq：请求参数k1的值是否等于value1。
	• ne：请求参数k2的值不等于value2。
	• 当请求参数k1和k2均存在，且k1等于value1，k2不等于value2时，输出响应体match condition。

ne

函数详细信息，请参见下表：

项目	描述
语法	`ne(arg1, arg2)` 。
说明	比较2个参数是否不等。
参数	• arg1：任意类型。 • arg2：应与 `arg1` 类型相同。
返回值	参数不等返回 `true` ，否则返回 `false` 。

示例	`key1 = 'value1'`
	`key2 = 'value2'`
	`if and($arg_k1, $arg_k2, eq(key1, $arg_k1), ne(key2, $arg_k2)) {`
	`  say('match condition')`
	`}`
	• 请求参数k1和k2都存在时，执行后续的比较操作。
	• 请求参数k1或k2不存在时，短路语义生效，不再执行后续的比较操作。
	• eq：请求参数k1的值是否等于value1。
	• ne：请求参数k2的值不等于value2。
	• 当请求参数k1和k2均存在，且k1等于value1，k2不等于value2时，输出响应体match condition。

null

函数详细信息，请参见下表：

项目	描述
语法	`null(v)` 。
说明	判断ES数据类型是否为空。
参数	v：需要传入的参数，类型为数组、字典和字符串，其他类型均返回false。
返回值	返回值为bool类型 • v是数组和字典，如果为空，返回true。 • v是字符串，如果值为空串，返回true。 • 其他情况均返回false。

示例	`d = []`
	`say(null(d))`
	`set(d, 1, 'v1')`
	`say(null(d))`
	`say(tostring(null('x')))`
	`say(tostring(null('')))`
	
	输出：
	`true`
	`false`
	`false`
	`true`

6.3. 数字类型相关

本文为您介绍数字类型相关函数的语法、说明、参数、返回值和示例。

add

函数详细信息，请参见下表：

项目	描述
语法	`add(n1, n2)` 。
说明	加法操作。
参数	• n1：数字类型。 • n2：数字类型。
返回值	返回 `n1 + n2` 的结果。

项目	描述
示例	``` n1 = add(10, 20) n2 = sub(10, 20) n3 = mul(10, 20) n4 = div(10, 20) n5 = mod(35, 20) say(concat('n1=', n1)) say(concat('n2=', n2)) say(concat('n3=', n3)) say(concat('n4=', n4)) say(concat('n5=', n5)) ``` 输出： n1=30 n2=-10 n3=200 n4=0.5 n5=15

sub

函数详细信息，请参见下表：

项目	描述
语法	`sub(n1, n2)` 。
说明	减法操作。
参数	• n1：数字类型。 • n2：数字类型。
返回值	返回 `n1 - n2` 的结果。

项目	描述

示例	`n1 = add(10, 20)`
	`n2 = sub(10, 20)`
	`n3 = mul(10, 20)`
	`n4 = div(10, 20)`
	`n5 = mod(35, 20)`
	`say(concat('n1=', n1))`
	`say(concat('n2=', n2))`
	`say(concat('n3=', n3))`
	`say(concat('n4=', n4))`
	`say(concat('n5=', n5))`
	
	输出：
	n1=30
	n2=-10
	n3=200
	n4=0.5
	n5=15

mul

函数详细信息，请参见下表：

项目	描述
语法	`mul(n1, n2)` 。
说明	乘法操作。
参数	• n1：数字类型。 • n2：数字类型。
返回值	返回 `n1 * n2` 的结果。

项目	描述

示例	`n1 = add(10, 20)`
	`n2 = sub(10, 20)`
	`n3 = mul(10, 20)`
	`n4 = div(10, 20)`
	`n5 = mod(35, 20)`
	`say(concat('n1=', n1))`
	`say(concat('n2=', n2))`
	`say(concat('n3=', n3))`
	`say(concat('n4=', n4))`
	`say(concat('n5=', n5))`
	
	输出：
	n1=30
	n2=-10
	n3=200
	n4=0.5
	n5=15

div

函数详细信息，请参见下表：

项目	描述
语法	`div(n1, n2)` 。
说明	除法操作。
参数	• n1：数字类型。 • n2：数字类型。
返回值	返回 `n1/n2` 的结果。

项目	描述

示例	`n1 = add(10, 20)`
	`n2 = sub(10, 20)`
	`n3 = mul(10, 20)`
	`n4 = div(10, 20)`
	`n5 = mod(35, 20)`
	`say(concat('n1=', n1))`
	`say(concat('n2=', n2))`
	`say(concat('n3=', n3))`
	`say(concat('n4=', n4))`
	`say(concat('n5=', n5))`
	
	输出：
	n1=30
	n2=-10
	n3=200
	n4=0.5
	n5=15

mod

函数详细信息，请参见下表：

项目	描述
语法	`mod(n1, n2)` 。
说明	求余操作。
参数	• n1：数字类型。 • n2：数字类型。
返回值	返回 `n1 % n2` 的结果（余数）。

项目	描述

示例	
	`n1 = add(10, 20)`
	`n2 = sub(10, 20)`
	`n3 = mul(10, 20)`
	`n4 = div(10, 20)`
	`n5 = mod(35, 20)`
	`say(concat('n1=', n1))`
	`say(concat('n2=', n2))`
	`say(concat('n3=', n3))`
	`say(concat('n4=', n4))`
	`say(concat('n5=', n5))`
	
	输出：
	n1=30
	n2=-10
	n3=200
	n4=0.5
	n5=15

gt

函数详细信息，请参见下表：

项目	描述
语法	`gt(n1, n2)` 。
说明	大于比较。
参数	• n1：数字类型。 • n2：数字类型。
返回值	若 `n1 > n2` ，返回 `true` ，否则返回 `false` 。

项目	描述

示例	
	`if and($arg_num, gt(tonumber($arg_num), 10)) {`
	`  say('num > 10')`
	`}`
	`if and($arg_num, ge(tonumber($arg_num), 10)) {`
	`  say('num >= 10')`
	`}`
	`if and($arg_num, lt(tonumber($arg_num), 10)) {`
	`  say('num < 10')`
	`}`
	`if and($arg_num, le(tonumber($arg_num), 10)) {`
	`  say('num <= 10')`
	`}`
	
	• 请求：/path1/path2/file?num=10，响应：num <= 10 num >= 10。
	• 请求：/path1/path2/file?num=11，响应：num > 10 num >= 10。
	• 请求：/path1/path2/file?num=9， 响应：num < 10 num <= 10。

ge

函数详细信息，请参见下表：

项目	描述
语法	`ge(n1, n2)` 。
说明	大于等于比较。
参数	• n1：数字类型。 • n2：数字类型。
返回值	若 `n1 >= n2` ，返回 `true` ，否则返回 `false` 。

项目	描述

示例	
	`if and($arg_num, gt(tonumber($arg_num), 10)) {`
	`  say('num > 10')`
	`}`
	`if and($arg_num, ge(tonumber($arg_num), 10)) {`
	`  say('num >= 10')`
	`}`
	`if and($arg_num, lt(tonumber($arg_num), 10)) {`
	`  say('num < 10')`
	`}`
	`if and($arg_num, le(tonumber($arg_num), 10)) {`
	`  say('num <= 10')`
	`}`
	
	• 请求：/path1/path2/file?num=10，响应：num <= 10 num >= 10。
	• 请求：/path1/path2/file?num=11，响应：num > 10 num >= 10。
	• 请求：/path1/path2/file?num=9， 响应：num < 10 num <= 10。

lt

函数详细信息，请参见下表：

项目	描述
语法	`lt(n1, n2)` 。
说明	小于比较。
参数	• n1：数字类型。 • n2：数字类型。
返回值	若 `n1 < n2` ，返回 `true` ，否则返回 `false` 。

项目	描述

示例	
	`if and($arg_num, gt(tonumber($arg_num), 10)) {`
	`  say('num > 10')`
	`}`
	`if and($arg_num, ge(tonumber($arg_num), 10)) {`
	`  say('num >= 10')`
	`}`
	`if and($arg_num, lt(tonumber($arg_num), 10)) {`
	`  say('num < 10')`
	`}`
	`if and($arg_num, le(tonumber($arg_num), 10)) {`
	`  say('num <= 10')`
	`}`
	
	• 请求：/path1/path2/file?num=10，响应：num <= 10 num >= 10。
	• 请求：/path1/path2/file?num=11，响应：num > 10 num >= 10。
	• 请求：/path1/path2/file?num=9， 响应：num < 10 num <= 10。

le

函数详细信息，请参见下表：

项目	描述
语法	`le(n1, n2)` 。
说明	小于等于比较。
参数	• n1：数字类型。 • n2：数字类型。
返回值	若 `n1 <= n2` ，返回 `true` ，否则返回 `false` 。

项目	描述

示例	`if and($arg_num, gt(tonumber($arg_num), 10)) {`
	`  say('num > 10')`
	`}`
	`if and($arg_num, ge(tonumber($arg_num), 10)) {`
	`  say('num >= 10')`
	`}`
	`if and($arg_num, lt(tonumber($arg_num), 10)) {`
	`  say('num < 10')`
	`}`
	`if and($arg_num, le(tonumber($arg_num), 10)) {`
	`  say('num <= 10')`
	`}`
	
	• 请求：/path1/path2/file?num=10，响应：num <= 10 num >= 10。
	• 请求：/path1/path2/file?num=11，响应：num > 10 num >= 10。
	• 请求：/path1/path2/file?num=9， 响应：num < 10 num <= 10。

floor

函数详细信息，请参见下表：

项目	描述
语法	`floor(n)` 。
说明	向下取整。
参数	n：数字类型。
返回值	返回 `n` 的向下取整。

示例	`if $arg_num {`
	`  say(concat('ceil: ', ceil(tonumber($arg_num))))`
	`  say(concat('floor: ', floor(tonumber($arg_num))))`
	`}`
	
	请求：/path1/path2/file?num=9.3，响应：ceil: 10 floor: 9。

ceil

函数详细信息，请参见下表：

项目	描述
语法	`ceil(n)` 。
说明	向上取整。

项目	描述
参数	n：数字类型。
返回值	返回 `n` 的向上取整。

示例	`if $arg_num {`
	`  say(concat('ceil: ', ceil(tonumber($arg_num))))`
	`  say(concat('floor: ', floor(tonumber($arg_num))))`
	`}`
	
	请求：/path1/path2/file?num=9.3，响应：ceil: 10 floor: 9。

6.4. 字符串类型相关函数

本文为您介绍字符串类型相关函数的语法、说明、参数、返回值和示例。

substr

函数详细信息，请参见下表。

项目	描述
语法	`substr(s, i, j)` 。
说明	字符串截取操作。
参数	• s：目标字符串。 • i：整型，截取起始下标。从1开始，-1表示字符串最尾字符。 • j：整型，截取终止下标。从1开始，-1表示字符串最尾字符。
返回值	返回 `s` 的子字符串 `s[i, j]` 。

示例	`//说明：判断文件类型是否为.m3u8的两种方法：`
	`if eq(substr($uri, -5, -1), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`
	`uri_len = len($uri)`
	`if eq(substr($uri, -5, uri_len), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`

concat

函数详细信息，请参见下表。

项目	描述
语法	`concat(s1, ...)` 。
说明	字符串连接操作。
参数	一个或多个参数，参数类型允许为数字字符串。
返回值	将多个参数连接为一个字符串，并返回该字符串。

示例	`//说明：判断文件类型是否为.m3u8的两种方法：`
	`if eq(substr($uri, -5, -1), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`
	`uri_len = len($uri)`
	`if eq(substr($uri, -5, uri_len), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`

upper

函数详细信息，请参见下表。

项目	描述
语法	`upper(s)` 。
说明	将字符串中所有的小写字母转换成大写字母。
参数	s：目标字符串。
返回值	返回大写 `s` 。

示例	`mystr = 'Hello, Dsl'`
	`say(upper(mystr))`
	`say(lower(mystr))`
	`//输出：`
	`//HELLO,DSL`
	`//hello, dsl`

lower

函数详细信息，请参见下表。

项目	描述
语法	`lower(s)` 。
说明	将字符串中所有的大写字母转换成小写字母。

项目	描述
参数	s：目标字符串。
返回值	返回小写 `s` 。

示例	`mystr = 'Hello, Dsl'`
	`say(upper(mystr))`
	`say(lower(mystr))`
	`//输出：`
	`//HELLO,DSL`
	`//hello, dsl`

len

函数详细信息，请参见下表。

项目	描述
语法	`len(s)` 。
说明	获取字符串的长度。
参数	s：目标字符串。
返回值	返回 `s` 的长度，整型。

示例	`//说明：判断文件类型是否为.m3u8的两种方法：`
	`if eq(substr($uri, -5, -1), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`
	`uri_len = len($uri)`
	`if eq(substr($uri, -5, uri_len), '.m3u8') {`
	`  say(concat($uri, ' is .m3u8'))`
	`}`

byte

函数详细信息，请参见下表。

项目	描述
语法	`byte(c)` 。
说明	获取字符的ASCII码。
参数	c：目标字符，必须为单个字符。
返回值	返回对应的ASCII码，数字类型。

项目	描述

示例	`say(byte('a'))`
	`say(byte('A'))`
	`//输出：97`
	`//65`

match_re

函数详细信息，请参见下表。

项目	描述
语法	`match_re(s, p [, o])` 。
说明	使用PCRE正则引擎，进行正则匹配，判断字符串是否匹配对应的正则表达式。详细信息，请参见PCRE正则语法。

参数	• s：目标字符串，字符类型。
	• p：正则表达式，字符类型。
	• o：正则引擎参数，字符类型，可选填。
	  ○ 不填写：则区分大小写。
	  ○ 填写为i：则不区分大小写。

返回值	匹配成功返回 `true` ，否则返回 `false` 。

示例	`url = concat('http://', $host, $uri)`
	`m1 = match_re(url, 'http://.*\.example\.com/.*', 'i')`
	`m2 = match_re(url,`
	`'^http://.*\.aliyundoc\.com\.cn/.*\.d\\.html(\?.*)?$')`
	`m3 = match_re(url, '^http://.*.demo.aliyundoc/.*\.d\.html(\?.*)?`
	`$')`
	`m4 = match_re(url, '^http://.*\.aliyundoc\.com\.cn/zt_d/')`
	`m5 = match_re(url, '^http://learn.aliyundoc.com.cn/zt_d/we2015/?`
	`$')`
	`m6 = match_re($args, 'from=wap1$')`
	`m7 = match_re($args, 'from=comos1$')`
	`if and(m1, or(m2, m3), not(m4), not(m5), or(not(m6), not(m7))) {`
	`  add_rsp_header('USER-DEFINED-1', 'hit1')`
	`  add_rsp_header('USER-DEFINED-2', 'hit2')`

capture_re

函数详细信息，请参见下表。

项目	描述
语法	`capture_re(s, p [,init])` 。
说明	正则捕获，并返回捕获结果。使用PCRE正则引擎，详细信息，请参见PCRE正则语法。
参数	• s：目标字符串，字符类型。 • p：正则表达式，字符类型。 • init：指定匹配开始位置，下标从1开始，整型。
返回值	匹配成功的若干子串通过字典类型返回，匹配失败返回空字典。

示例	`pcs = capture_re($request_uri,'^/([^/]+)/([^/]+)([^?]+)\?(.*)')`
	`sec1 = get(pcs, 1)`
	`sec2 = get(pcs, 2)`
	`sec3 = get(pcs, 3)`
	`if or(not(sec1), not(sec2), not(sec3)) {`
	`  add_rsp_header('X-TENGINE-ERROR', 'auth failed - missing necessary uri set')`
	`  exit(403)`
	`}`
	`digest = md5(concat(sec1, sec3))`
	`if ne(digest, sec2) {`
	`  add_rsp_header('X-TENGINE-ERROR', 'auth failed - invalid digest')`
	`  exit(403)`
	`}`

gsub_re

函数详细信息，请参见下表。

项目	描述
语法	`gsub_re(subject, regex, replace [,option])` 。
说明	正则替换，并返回替换后的副本。使用PCRE正则引擎，详细信息，请参见PCRE正则语法。

参数	• subject：目标字符串，字符类型。
	• regex：正则表达式，字符类型。
	• replace：替换字符，字符类型。
	`replace` 部分可以引用匹配部分，即：
	  ○ $0：表示 `regex` 整体匹配的部分。
	  ○ $N：表示 `regex` 第N个 `()` 匹配的部分。
	• option：正则引擎参数，字符类型，可选。

项目	描述
返回值	`subject` 中所有的符合参数 `regex` 的子串都将被参数 `replace` 所指定的字符串所替换，并返回替换后的副本。

示例	`subject = 'Hello, Es'`
	`regex = '([a-zA-Z])[a-z]+'`
	`replace = '[$0,$1]'`
	`add_rsp_header('X-DEBUG-GSUB-RE', gsub_re(subject, regex, replace))`
	`//输出:`
	`//X-DEBUG-GSUB-RE: [Hello,H], [Es,E]`

split

函数详细信息，请参见下表。

项目	描述
语法	`split(s [,sep])` 。
说明	分隔字符串，并返回分隔结果。
参数	• s：目标字符串，字符类型。 • sep：字符类型。
返回值	分隔元素包含在字典类型中返回，由数字下标作 `key` ，起始下标为1，例如：[1]=xx，[2]=yy；若 `sep` 为空，则默认以任意空白字符分隔。默认空白字符包含：空格、Tab。

示例	`if $arg_from {`
	`  t = split($arg_from, ',')`
	`  if get(t, 1) {`
	`  say(concat('[1]=', get(t, 1)))`
	`  }`
	`  if get(t, 2) {`
	`  say(concat('[2]=', get(t, 1)))`
	`  }`
	`}`
	`//请求: ?from=xx1,xx2,xx3`
	`//响应: [1]=xx1`
	`//[2]=xx1`

split_as_key

函数详细信息，请参见下表。

项目	描述
语法	`split_as_key(s [,sep])` 。
说明	分隔字符串，并返回分隔结果。
参数	• s：目标字符串，字符类型。 • sep：分隔符，字符类型。
返回值	同 `split()` ，区别在于 `key` ： *[分割元素]* -> *[分割元素]* 。

示例	`def echo_each(k, v, u) {`
	`  s = concat(k, '=', v, ' u=', get(u, 1))`
	`  say(s)`
	`}`
	`if $arg_from {`
	`  t = split_as_key($arg_from, ',')`
	`  foreach(t, echo_each, ['hi,dsl'])`
	`}`
	`//请求：?from=xx1,xx2,xx3`
	`//响应：xx2=xx2 u=hi,dsl`
	`//xx1=xx1 u=hi,dsl`
	`//xx3=xx3 u=hi,dsl`

tohex

函数详细信息，请参见下表。

项目	描述
语法	`tohex(s)` 。
说明	十六进制转换。
参数	s：字符串。
返回值	返回 `s` 的十六进制可读形式。

示例	`digest = sha1('xxxx')`
	`add_rsp_header('X-DSL-TOHEX', tohex(digest))`
	`//说明：增加响应头`
	`//X-DSL-TOHEX: 4ad583af22c2e7d40c1c916b2920299155a46464`

tostring

函数详细信息，请参见下表。

项目	描述
语法	`tostring(a)` 。
说明	字符串类型转换。
参数	a：任意类型。
返回值	返回参数 `a` 转换后的字符串。

示例	`s = tostring(123)`
	`add_rsp_header('X-DSL-TOSTRING', s)`
	`//说明：增加响应头`
	`//X-DSL-TOSTRING: 123`

tochar

函数详细信息，请参见下表。

项目	描述
语法	`tochar(n1, n2, ...)` 。
说明	• 接受1个或多个整型参数，返回对应整型参数的内部数值表示的字符串，例如：48对应于字符“0”。 • 返回字符串的长度为参数个数。
参数	nX：整型参数。
返回值	返回转换后的字符串。

示例	`add_rsp_header('X-DSL-TOCHAR', tochar(97))`
	`add_rsp_header('X-DSL-TOCHAR', tochar(97, 98), true)`
	`//输出：增加响应头`
	`//X-DSL-TOCHAR: a`
	`//X-DSL-TOCHAR: ab`
	`if $arg_filename {`
	`  hn = 'Content-Disposition'`
	`  add_rsp_header('Content-Disposition',`
	`concat('attachment;filename=', tochar(34), filename, tochar(34)))`
	`  add_rsp_header(hn, hv)`
	`}`
	`//输出：增加响应头`
	`//Content-Disposition: attachment;filename="请求参数filename的值"`

reverse

函数详细信息，请参见下表。

项目	描述
语法	`reverse(str)` 。
说明	字符串反转。
参数	str：待反转的字符串。
返回值	返回字符类型，返回反转后的字符串。

示例	`say(reverse('hello'))`
	输出：
	`#olleh`

find

函数详细信息，请参见下表。

项目	描述
语法	`string.find (s, substr, pos)` 。
说明	在目标字符串中搜索指定的字符串。
参数	• s：待查找的字符串。 • substr：需要查找的子串。 • pos（可选参数）：该参数为索引，类型为数值，指定查找的起始位置。可以为负数，默认起始位置为1。

返回值	• 成功：返回数组类型。
	  ○ 索引1，返回查找的起始位置。
	  ○ 索引2，返回查找的截止位置。
	• 失败：返回空数组。

项目	描述
示例	``` str = 'hello dsl' add_rsp_header('string-find()-start', tostring(get(find(str, 'dsl'), 1))) str = 'hello dsl 12' add_rsp_header('string-find()-end', tostring(get(find(str, 'dsl'), 2))) str = 'hello dsl' add_rsp_header('string-find()-tail-start', tostring(get(find(str, 'dsl', -6), 1))) str = 'hello dsl 12' add_rsp_header('string-find()-tail-end', tostring(get(find(str, 'dsl', -6), 2))) ``` 输出： ``` string-find()-start:7 string-find()-end:9 string-find()-tail-start:7 string-find()-tail-end:9 ```

format

函数详细信息，请参见下表。

项目	描述
语法	`format(fmt, ···)` 。
说明	返回不定数量参数的格式化版本，格式化串为第一个参数（必须是一个字符串）。格式化字符串遵循ISO C函数sprintf的规则。 fmt规则格式为：%[指定参数][标识符][宽度][.精度]指示符 • %%：印出百分比符号，不转换。 • %c：整数转换成对应的ASCII字元。 • %d：整数转换成十进位。 • %f：倍精确度数字转换成浮点数。 • %o：整数转换成八进位。 • %s：整数转换成字符串。 • %x：整数转换成小写十六进位。 • %X：整数转换成大写十六进位。
参数	• fmt：string类型，格式化字符串。 • 可变参数：可以为任意类型。
返回值	返回值为string类型。

项目	描述

示例	`say(concat('format:', format('%%%s$%.2s$%s$%c$%d$%2.2f$%.2o$%x$%X', 'format', 3.1415926, true, 95, 3.1415926, 3.1415926, 3.1415926, 10, 10)))`
	
	输出：
	`format:%format$3.$true$_$3$3.14$03$a$A`

tobin

函数详细信息，请参见下表。

项目	描述
语法	`tobin(str)` 。
说明	16进制转ASCII字符串。
参数	str：双字节16进制字符串，不区分大小写。
返回值	返回值为string类型。

示例	`say(concat('tobin:', tobin('2F2F')))`
	
	输出：
	`tobin://`

trim

函数详细信息，请参见下表。

项目	描述
语法	`trim(s, [, loc])` 。
说明	剔除s两端或指定端的全部空白字符，并返回剔除后的字符串。

参数	• s：目标字符串。
	• loc（可选参数）：默认为both，可用值如下：
	  ○ both：剔除两端。
	  ○ left：仅剔除左端。
	  ○ right：仅剔除右端。

返回值	返回剔除后的字符串。

项目	描述

示例	`say(concat('trim():', trim(' abcd ')))`
	`say(concat('trim(left):', trim(' abcd ', 'left')))`
	`say(concat('trim(right):', trim(' abcd ', 'right')))`
	
	输出：
	`trim():abcd`
	`trim(left):abcd`
	`trim(right): abcd`

6.5. 字典类型相关

本文为您介绍字典类型相关函数的语法、参数、示例和返回值。

set

使用set在字典d中设置k和v，函数的详细信息见下表。

项目	描述
语法	set(d, k, v)
参数	• d：目标字典。 • k：key值，可以为任意类型。 • v：value值，可以为任意类型。

项目	描述

示例	• 示例1
	`outer_keys=['e66fd4aa-f281-472f-b919-fc7e7474****', '66fee78d-1887-42ec-9119-a9b50b7f****']`
	`say(concat('keys[1]=', get(outer_keys, 1)))`
	`say(concat('keys[2]=', get(outer_keys, 2)))`
	`inner_keys=[]`
	`set(inner_keys, 'dev', '243390eb-00b7-4551-a6b8-021bb34d****')`
	`set(inner_keys, 'zeus', '4747d33b-12b0-45e6-ac10-a8e191d6****')`
	`def echo_each(k, v, u) {`
	`  s = concat('keys[', k, ']=', v)`
	`  say(s)`
	`}`
	`foreach(inner_keys, echo_each, [])`
	• 示例2
	`d_inner = []`
	`set(d_inner, 'name', 'inner dsl')`
	`d_outer = []`
	`set(d_outer, 'dictA', d_inner)`
	`v = get(d_outer, 'dictA')`
	`if v {`
	`  v = get(v, 'name')`
	`  if v {`
	`  add_rsp_header('X-DSL-NESTED-DICT', v)`
	`  }`
	`}`

返回值	始终返回true。本示例的返回值如下：
	• 示例1的返回值
	`keys[1]=e66fd4aa-f281-472f-b919-fc7e7474****`
	`keys[2]=66fee78d-1887-42ec-9119-a9b50b7f****`
	`keys[dev]=243390eb-00b7-4551-a6b8-021bb34d****`
	`keys[zeus]=4747d33b-12b0-45e6-ac10-a8e191d6****`
	• 示例2的返回值：增加响应头。
	`X-DSL-NESTED-DICT: inner dsl`

get

使用get获取字典d中k对应的v，函数的详细信息见下表。

项目	描述
语法	get(d, k)

项目	描述
参数	• d：目标字典。 • k：key值，可以为任意类型。

示例	`outer_keys=['e66fd4aa-f281-472f-b919-fc7e7474****', '66fee78d-1887-42ec-9119-a9b50b7f****']`
	`say(concat('keys[1]=', get(outer_keys, 1)))`
	`say(concat('keys[2]=', get(outer_keys, 2)))`
	`inner_keys=[]`
	`set(inner_keys, 'dev', '243390eb-00b7-4551-a6b8-021bb34d****')`
	`set(inner_keys, 'zeus', '4747d33b-12b0-45e6-ac10-a8e191d6****')`
	`def echo_each(k, v, u) {`
	`  s = concat('keys[', k, ']=', v)`
	`  say(s)`
	`}`
	`foreach(inner_keys, echo_each, [])`

返回值	获取成功返回对应的值，获取失败返回false。本示例的返回值如下：
	`keys[1]=e66fd4aa-f281-472f-b919-fc7e7474****`
	`keys[2]=66fee78d-1887-42ec-9119-a9b50b7f****`
	`keys[dev]=243390eb-00b7-4551-a6b8-021bb34d****`
	`keys[zeus]=4747d33b-12b0-45e6-ac10-a8e191d6****`

foreach

foreach函数的说明如下：

- 遍历字典d中的元素，依次回调函数f。
- f原型要求为 `f(key, value, user_data)` 。
- 当 `f()` 返回false时， `foreach()` 循环终止。

函数的详细信息见下表。

项目	描述
语法	foreach(d, f, user_data)
参数	• d：目标字典。 • f：回调函数。 • user_data：用于传递用户数据，为字典类型。

项目	描述

示例	• 示例1
	`outer_keys=['e66fd4aa-f281-472f-b919-fc7e7474****', '66fee78d-1887-42ec-9119-a9b50b7f****']`
	`say(concat('keys[1]=', get(outer_keys, 1)))`
	`say(concat('keys[2]=', get(outer_keys, 2)))`
	`inner_keys=[]`
	`set(inner_keys, 'dev', '243390eb-00b7-4551-a6b8-021bb34d****')`
	`set(inner_keys, 'zeus', '4747d33b-12b0-45e6-ac10-a8e191d6****')`
	`def echo_each(k, v, u) {`
	`  s = concat('keys[', k, ']=', v)`
	`  say(s)`
	`}`
	`foreach(inner_keys, echo_each, [])`
	• 示例2
	输出 `m3u8` 的前2个分片，演示终止foreach循环。
	`def echo_each(k, v, u) {`
	`  say(v)`
	`  if match_re(v, '.*ts') {`
	`  ts_cnt = get(u, 'ts_cnt')`
	`  ts_cnt = add(ts_cnt, 1)`
	`  set(u, 'ts_cnt', ts_cnt)`
	`  if ge(ts_cnt, 2) {`
	`  return false`
	`  }`
	`  }`
	`}`
	`m3u8 = ''`
	`m3u8 = concat(m3u8, '#EXTM3U8', '\n')`
	`m3u8 = concat(m3u8, '#EXT-X-MEDIA-SEQUENCE:14065****\n')`
	`m3u8 = concat(m3u8, '#EXT-X-TARGETDURATION:10\n')`
	`m3u8 = concat(m3u8, '#EXTINF:8,\n')`
	`m3u8 = concat(m3u8, 'http://***.cn/cache/289_/seg0/index140651514_1406****.ts\n')`
	`m3u8 = concat(m3u8, '#EXTINF:9,\n')`
	`m3u8 = concat(m3u8, 'http://***.cn/cache/289_/seg0/index140651514_1406****.ts\n')`
	`m3u8 = concat(m3u8, '#EXTINF:10,\n')`
	`m3u8 = concat(m3u8, 'http://***.cn/cache/289_/seg0/index140651514_1406****.ts\n')`
	`lines = split(m3u8, '\n')`
	`u = []`
	`set(u, 'ts_cnt', 0)`
	`foreach(lines, echo_each, u)`

项目	描述

返回值	始终返回true。本示例的返回值如下：
	• 示例1的返回值
	`keys[1]=e66fd4aa-f281-472f-b919-fc7e7474****`
	`keys[2]=66fee78d-1887-42ec-9119-a9b50b7f****`
	`keys[dev]=243390eb-00b7-4551-a6b8-021bb34d****`
	`keys[zeus]=4747d33b-12b0-45e6-ac10-a8e191d6****`
	• 示例2的返回值
	`#EXTM3U8`
	`#EXT-X-MEDIA-SEQUENCE:140651513`
	`#EXT-X-TARGETDURATION:10`
	`#EXTINF:8,`
	`http://***.cn/cache/289_/seg0/index140651514_1406****.ts`
	`#EXTINF:9,`
	`http://***.cn/cache/289_/seg0/index140651514_1406****.ts`

del

使用del删除字典d中的k/v对，函数的详细信息见下表。

项目	描述
语法	del(d, k)
参数	• d：目标字典。 • k：key值，可以为任意类型。

示例	`var_a = []`
	`set(var_a, 'note_a', 'note a info')`
	`if get(var_a, 'note_a') {`
	`  add_rsp_header('X-RESPOND-OUTPUT', 'found var_a key', true)`
	`} else {`
	`  add_rsp_header('X-RESPOND-OUTPUT', 'del var_a key', true)`
	`}`
	`del(var_a, 'note_a')`
	`if get(var_a, 'note_a') {`
	`  add_rsp_header('X-RESPOND-OUTPUT', 'found var_a key', true)`
	`} else {`
	`  add_rsp_header('X-RESPOND-OUTPUT', 'del var_a key', true)`
	`}`

返回值	始终返回true。本示例的返回值如下：
	`X-RESPOND-OUTPUT: found var_a key`
	`X-RESPOND-OUTPUT: del var_a key`

6.6. 请求处理相关

本文为您介绍请求处理相关函数的语法、说明、参数、返回值和示例。

add_req_header

函数详细信息，请参见下表：

项目	描述
语法	`add_req_header(name, value [, append])` 。
说明	添加请求头，即回源请求头。
参数	• name：待添加的请求头 `name` ，字符类型。 • value：待添加的请求头 `value` ，字符类型。 • append：若请求头已添加， `append` 决定是否追加 `value` ，默认覆盖（即默认不追加），布尔类型。
返回值	默认返回 `true` ，无效请求头返回 `false` 。

示例	`add_req_header('USER-DEFINED-REQ-1', '1')`
	`add_req_header('USER-DEFINED-REQ-1', 'x', true)`
	`add_req_header('USER-DEFINED-REQ-2', '2')`
	`del_req_header('USER-DEFINED-REQ-2')`
	说明：添加2个请求头，分别为
	`USER-DEFINED-REQ-1: 1`
	`USER-DEFINED-REQ-1: x`
	`USER-DEFINED-REQ-2`**先添加、后删除，故回源请求头中无**`USER-DEFINED-REQ-2`

del_req_header

函数详细信息，请参见下表：

项目	描述
语法	`del_req_header(name)` 。
说明	删除请求头，即回源请求头。
参数	name：待删除的请求头 `name` ，字符类型。
返回值	默认返回 `true` ，无效请求头返回 `false` 。

项目	描述

示例	`add_req_header('USER-DEFINED-REQ-1', '1')`
	`add_req_header('USER-DEFINED-REQ-1', 'x', true)`
	`add_req_header('USER-DEFINED-REQ-2', '2')`
	`del_req_header('USER-DEFINED-REQ-2')`
	说明：添加2个请求头，分别为
	`USER-DEFINED-REQ-1: 1`
	`USER-DEFINED-REQ-1: x`
	`USER-DEFINED-REQ-2`**先添加、后删除，故回源请求头中无**`USER-DEFINED-REQ-2`

add_rsp_header

函数详细信息，请参见下表：

项目	描述
语法	`add_rsp_header(name, value [, append])` 。
说明	添加响应头。
参数	• name：待添加的响应头 `name` ，字符类型。 • value：待添加的响应头 `value` ，字符类型。 `value` 可包含如下标记，用于在响应阶段执行动态替换： ○ ${x} ：将替换为 `ngx.var.x` 的值。 ○ @{y}：将替换为 *响应头*`y` 的值。 • append：若响应头已添加， `append` 决定是否追加 `value` ，默认覆盖，布尔类型。 。
返回值	默认返回 `true` ，无效响应头返回 `false` 。

示例	`add_rsp_header('USER-DEFINED-RSP-1', '1')`
	`add_rsp_header('USER-DEFINED-RSP-1', 'x', true)`
	`add_rsp_header('USER-DEFINED-RSP-2', '2')`
	`del_rsp_header('USER-DEFINED-RSP-2')`
	说明：添加2个响应头，分别为
	`USER-DEFINED-RSP-1: 1`
	`USER-DEFINED-RSP-1: x`
	`USER-DEFINED-RSP-2`**先添加、后删除，故响应头中无**`USER-DEFINED-RSP-2`

del_rsp_header

函数详细信息，请参见下表：

项目	描述
语法	`del_rsp_header(name)` 。
说明	删除响应头。
参数	name：待删除的响应头 `name` ，字符类型。
返回值	默认返回 `true` ，无效响应头返回 `false` 。

示例	`add_rsp_header('USER-DEFINED-RSP-1', '1')`
	`add_rsp_header('USER-DEFINED-RSP-1', 'x', true)`
	`add_rsp_header('USER-DEFINED-RSP-2', '2')`
	`del_rsp_header('USER-DEFINED-RSP-2')`
	说明：添加2个响应头，分别为
	`USER-DEFINED-RSP-1: 1`
	`USER-DEFINED-RSP-1: x`
	`USER-DEFINED-RSP-2`**先添加、后删除，故响应头中无**`USER-DEFINED-RSP-2`

encode_args

函数详细信息，请参见下表：

项目	描述
语法	`encode_args(d)` 。
说明	将字典 `d` 中的 `k/v` ，转换为URI编码的k1=v1&k2=v2格式的字符串。
参数	d：字典类型。
返回值	返回URI编码格式的字符串。

项目	描述

示例	`my_args = []`
	`set(my_args, 'signature', 'da9dc4b7-87ae-4330-aaaf-e5454e2c2af1')`
	`set(my_args, 'algo', 'private sign1')`
	`my_args_str = encode_args(my_args)`
	`add_rsp_header('X-DSL-ENCODE-ARGS', my_args_str)`
	`to_args = decode_args(my_args_str)`
	`if get(to_args, 'algo') {`
	`  add_rsp_header('X-DSL-DECODE-ARGS-ALGO', get(to_args, 'algo'))`
	`}`
	`if get(to_args, 'signature') {`
	`  add_rsp_header('X-DSL-DECODE-ARGS-SIGN', get(to_args, 'signature'))`
	`}`
	输出：添加3个响应头，分别为
	`X-DSL-ENCODE-ARGS: signature=da9dc4b7-87ae-4330-aaaf-e5454e2c2af1&algo=private%20sign1`
	`X-DSL-DECODE-ARGS-ALGO: private sign1`
	`X-DSL-DECODE-ARGS-SIGN: da9dc4b7-87ae-4330-aaaf-e5454e2c2af1`

decode_args

函数详细信息，请参见下表：

项目	描述
语法	`decode_args(s)` 。
说明	将URI编码的k1=v1&k2=v2格式的字符串，转换为字典类型。
参数	s：目标字符串。
返回值	返回转换后的字典对象。

项目	描述

示例	`my_args = []`
	`set(my_args, 'signature', 'da9dc4b7-87ae-4330-aaaf-e5454e2c2af1')`
	`set(my_args, 'algo', 'private sign1')`
	`my_args_str = encode_args(my_args)`
	`add_rsp_header('X-DSL-ENCODE-ARGS', my_args_str)`
	`to_args = decode_args(my_args_str)`
	`if get(to_args, 'algo') {`
	`  add_rsp_header('X-DSL-DECODE-ARGS-ALGO', get(to_args, 'algo'))`
	`}`
	`if get(to_args, 'signature') {`
	`  add_rsp_header('X-DSL-DECODE-ARGS-SIGN', get(to_args, 'signature'))`
	`}`
	输出：添加3个响应头，分别为
	`X-DSL-ENCODE-ARGS: signature=da9dc4b7-87ae-4330-aaaf-e5454e2c2af1&algo=private%20sign1`
	`X-DSL-DECODE-ARGS-ALGO: private sign1`
	`X-DSL-DECODE-ARGS-SIGN: da9dc4b7-87ae-4330-aaaf-e5454e2c2af1`

rewrite

函数详细信息，请参见下表：

项目	描述
语法	`rewrite(url, flag, code)` 。
说明	改写操作或重定向操作。

项目	描述

参数	• url：目标URL，字符类型。
	  ○ 当flag=redirect或flag=break时，仅改写URI，则参数URL表示改写后的目标URI。
	  ○ 当flag=enhance_redirect或flag=enhance_break时，修改整个URI和参数，参数URL表示改写后的目标URI和参数。
	• flag：重写模式，字符类型。
	  ○ redirect ：仅修改URI，不修改参数；默认执行302跳转至URL；如果指定参数 `code` ，则响应码可自定义为：301、302（默认）、303、307或308。
	  ○ break ：仅修改URI，不修改参数，将URI修改为URL。
	  ○ enhance_redirect：与 `redirect` 类似，但是修改整个URI和参数。
	  ○ enhance_break：与 `break` 类似，但是修改整个URI和参数。
	• code：为响应码，数字类型 。
	当flag=redirect或flag=enhance_redirect时，可自定义响应码。

返回值	• 对于改写操作，默认返回 `true` 。 • 对于重定向操作，默认不返回。

项目	描述

示例	`if and($arg_mode, eq($arg_mode, 'rewrite:enhance_break')) {`
	`  rewrite('/a/example/examplefile.txt?k=v', 'enhance_break')`
	`}`
	说明：回源和缓存的`uri`**+参数，修改为**`/a/example/examplefile.txt?k=v`
	`if and($arg_mode, eq($arg_mode, 'rewrite:enhance_redirect')) {`
	`  rewrite('/a/example/examplefile.txt?k=v', 'enhance_redirect')`
	`}`
	`if and($arg_mode, eq($arg_mode, 'rewrite:enhance_redirect_301')) {`
	`  rewrite('/a/example/examplefile.txt?k=v', 'enhance_redirect', 301)`
	`}`
	说明：`302`**或**`301`**跳转至**`/a/example/examplefile.txt?k=v`
	`if and($arg_mode, eq($arg_mode, 'rewrite:break')) {`
	`  rewrite('/a/example/examplefile.txt', 'break')`
	`}`
	说明：回源和缓存的`uri`**，修改为**`/a/example/examplefile.txt`**，保持原参数不变**
	`if and($arg_mode, eq($arg_mode, 'rewrite:redirect')) {`
	`  rewrite('/a/example/examplefile.txt', 'redirect')`
	`}`
	`if and($arg_mode, eq($arg_mode, 'rewrite:redirect_301')) {`
	`  rewrite('/a/example/examplefile.txt', 'redirect', 301)`
	`}`
	说明：`302`**或**`301`**跳转至**`/a/example/examplefile.txt`**，保持原参数不变**

say

函数详细信息，请参见下表：

项目	描述
语法	`say(arg)` 。
说明	输出响应体，并在行尾追加换行符。
参数	arg：任意类型。
返回值	无。

示例	`say('hello')`
	`print('byebye')`
	`print('byebye')`
	输出：
	`hello`
	`byebyebyebye`

print

函数详细信息，请参见下表：

项目	描述
语法	`print(arg)` 。
说明	输出响应体与 `say()` 相同，但不会在行尾追加换行符。
参数	arg：任意类型。
返回值	无。

示例	`say('hello')`
	`print('byebye')`
	`print('byebye')`
	输出：
	`hello`
	`byebyebyebye`

exit

函数详细信息，请参见下表：

项目	描述
语法	`exit(code [, body])` 。
说明	以状态码 `code` 结束当前请求。若有 `body` ，则为响应体。

项目	描述
参数	• code：响应状态码。 • body：响应体。
返回值	无。

项目	描述

示例	• 示例1
	`if not($arg_key) {`
	`  exit(403)`
	`}`
	说明：如果请求未携带参数`key`**时，**`403`**拒绝请求。**
	`if not($cookie_user) {`
	`  exit(403, 'not cookie user')`
	`}`
	说明：当请求未携带`cookie user`**时，**`403`**拒绝请求，响应**`body`**为**`'not cookie user'`
	`if not(0) {`
	`  exit(403)`
	`}`
	说明：`not(0)`**的结果为**`false`
	`if not(false) {`
	`  exit(403)`
	`}`
	说明：`not(false)`**的结果为**`true`
	• 示例2
	`pcs = capture_re($request_uri,'^/([^/]+)/([^/]+)([^?]+)\?(.*)')`
	`sec1 = get(pcs, 1)`
	`sec2 = get(pcs, 2)`
	`sec3 = get(pcs, 3)`
	`if or(not(sec1), not(sec2), not(sec3)) {`
	`  add_rsp_header('X-TENGINE-ERROR', 'auth failed - missing necessary uri set')`
	`  exit(403)`
	`}`
	`digest = md5(concat(sec1, sec3))`
	`if ne(digest, sec2) {`
	`  add_rsp_header('X-TENGINE-ERROR', 'auth failed - invalid digest')`
	`  exit(403)`
	`}`

6.7. 限速相关函数

本文为您介绍限速相关函数的语法、说明、参数、返回值和示例。

limit_rate_after

函数详细信息，请参见下表。

项目	描述
语法	`limit_rate_after(n, unit)` 。
说明	设置不限速大小。 说明 该函数只是针对单请求设置不限速大小，而非全网的带宽限制。
参数	• n：整型。如果不配置 `unit` ，则默认单位为Byte。 • （可选）unit：字符类型。如果配置 `unit` ，则可用值为 `K` 、 `M` 或 `G` ，不区分大小写。
返回值	设置成功返回 `true` ，设置失败返回 `false` 。

示例	`limit_rate_after(10, 'k')`
	`limit_rate(1, 'm')`
	`//说明：设置前10 KB不限速，之后限速为1 MB/s。`

limit_rate

函数详细信息，请参见下表。

项目	描述
语法	`limit_rate(n, unit)` 。
说明	设置限速值。限速值默认下限为4 KB/s，当您设置的限速值低于4 KB/s时，默认生效4 KB/s。 说明 该函数只是针对单请求进行限速，而非全网的带宽限制。
参数	• n：整型。如果不配置 `unit` ，则默认单位为Byte/s。 • （可选）unit：字符类型。如果配置 `unit` ，则可用值为 `K` 、 `M` 或 `G` ，不区分大小写。
返回值	成功设置返回 `true` ，失败设置返回 `false` 。

项目	描述

示例	`limit_rate_after(10, 'k')`
	`limit_rate(1, 'm')`
	`//说明：设置前10 KB不限速，之后限速为1 MB/s。`

6.8. 缓存相关

本文为您介绍缓存相关函数的语法、说明、参数、返回值和示例。

set_cache_ttl

函数详细信息，请参见下表：

项目	描述
语法	`set_cache_ttl(type, ttl)` 。
说明	设置资源缓存时长。
参数	• type 缓存类型，可用值 `'path|code'` ，字符类型。 • ttl ○ 当type=path时，表明针对某类资源设置缓存时间，则 `ttl` 为缓存时长，要求数字类型。 ○ 当type=code时，表明针对响应码设置缓存时间，则为 `ttl` 各响应码的缓存时长，要求字符类型。 ○ 缓存时长默认单位：秒。
返回值	设置成功返回 `true` ，设置失败返回 `false` 。

项目	描述

示例	`if match_re($uri, '^/image') {`
	`  set_cache_ttl('code', '301=10,302=5')`
	`}`
	`if eq(substr($uri, -4, -1), '.mp4') {`
	`  set_cache_ttl('path', 5)`
	`}`
	`if match_re($uri, '^/201801/mp4/') {`
	`  set_cache_ttl('path', 50)`
	`}`
	`if match_re($uri, '^/201802/flv/') {`
	`  set_cache_ttl('path', 10)`
	`}`
	
	说明
	• /image开头的uri，针对响应码设置缓存时长，301缓存10s，302缓存5s。
	• 针对文件名、uri路径，设置各种不同的缓存时长

6.9. 时间相关

本文为您介绍时间相关函数的语法、说明、参数、返回值和示例。

today

函数详细信息，请参见下表：

项目	描述
语法	`today()` 。
说明	返回当前时间（本地时间）字符串，格式：yyyy-mm-dd。
参数	无。
返回值	返回当前时间字符串，格式：yyyy-mm-dd。

示例	`say(concat('today:', today()))`
	
	输出：
	`today:2019-05-23`

time

函数详细信息，请参见下表：

项目	描述
语法	`time()` 。
说明	返回当前的UNIX时间戳（不包含毫秒的小数部分），单位：秒。 说明 UNIX时间戳不区分时区，表示的是从1970年1月1日（UTC/GMT的午夜）开始所经过的秒数。
参数	无。
返回值	返回当前的UNIX时间戳。

示例	`say(concat('time:', time()))`
	输出：
	`time:1559109666`

now

函数详细信息，请参见下表：

项目	描述
语法	`now()` 。
说明	返回当前的UNIX时间戳（数值类型，包含毫秒的小数部分），单位：秒。 说明 UNIX时间戳不区分时区，表示的是从1970年1月1日（UTC/GMT的午夜）开始所经过的秒数。
参数	无。
返回值	返回当前的UNIX时间戳（数值类型，包含毫秒的小数部分），单位：秒。

示例	`say(concat('now:', now()))`
	输出：
	`now:1559109666.644`

localtime

函数详细信息，请参见下表：

项目	描述
语法	`localtime()` 。
说明	返回当前时间（本地时间）字符串，格式：yyyy-mm-dd hh:mm:ss。
参数	无。
返回值	返回当前时间字符串，格式：yyyy-mm-dd hh:mm:ss。

示例	`say(concat('localtime:', localtime()))`
	输出：
	`localtime:2019-05-29 14:02:41`

utctime

函数详细信息，请参见下表：

项目	描述
语法	`utctime()` 。
说明	返回当前时间字符串（UTC时间），格式：yyyy-mm-dd hh:mm:ss。
参数	无。
返回值	返回当前时间字符串，格式：yyyy-mm-dd hh:mm:ss。

示例	`say(concat('utctime:', utctime()))`
	输出：
	`utctime:2019-05-29 06:02:41`

cookie_time

函数详细信息，请参见下表：

项目	描述
语法	`cookie_time(sec)` 。
说明	生成cookie格式的GMT时间字符串。
参数	sec：UNIX时间戳。例如：调用 `time()` 获取。
返回值	基于 `sec` 表示的UNIX时间戳，返回cookie格式的时间字符串。

项目	描述

示例	`say(concat('cookie_time:', cookie_time(time())))`
	
	输出：
	`cookie_time:Wed, 29-May-19 06:02:41 GMT`

http_time

函数详细信息，请参见下表：

项目	描述
语法	`http_time(sec)` 。
说明	生成HTTP header格式的时间字符串。例如：Last-Modified。 注意 该函数只能生成GMT标准的时间。
参数	sec：UNIX时间戳。例如：调用 `time()` 获取。
返回值	基于 `sec` 表示的UNIX时间戳，返回HTTP header格式的时间字符串，用于HTTP头的时间。

示例	`say(concat('http_time:', http_time(time())))`
	
	输出
	`http_time:Wed, 29 May 2019 06:02:41 GMT`

parse_http_time

函数详细信息，请参见下表：

项目	描述
语法	`parse_http_time(str)` 。
说明	解析HTTP header格式的时间字符串，并返回对应的UNIX时间戳。 注意 该函数不识别时区，因此需要把本地时间先转换为GMT标准的时间以后再传递给该函数。
参数	str：待转换的HTTP header格式的时间字符串。格式：`Wed, 29 May 2019 06:02:41 GMT` 。调用 `http_time()` 获取。

项目	描述
返回值	成功返回UNIX时间戳，失败返回 `false` 。

示例	`say(concat('parse_http_time:', parse_http_time(http_time(time()))))`
	输出
	`parse_http_time:1559109761`

unixtime

函数详细信息，请参见下表：

项目	描述
语法	`unixtime(year, month, day, hour, min, sec)` 。
说明	根据本地时间的年、月、日、时、分、秒，生成UNIX时间戳并返回。 **说明** UNIX时间戳不区分时区，表示的是从1970年1月1日（UTC/GMT的午夜）开始所经过的秒数。
参数	• year：指定年。 • month：指定月。 • day：指定日。 • hour：指定小时。 • min：指定分钟。 • sec：指定秒。
返回值	返回转换后的UNIX时间戳。

项目	描述

示例	• 示例一：
	`t = unixtime(1970, 1, 1, 8, 0, 0)`
	`say(concat('unixtime()=', t))`
	输出
	`unixtime()=0`
	• 示例二：
	`t = unixtime(2021,12,23,0,0,0)`
	`say(concat('unixtime()=', t))`
	输出
	`unixtime()=1640188800`

6.10. 密码算法相关

本文为您介绍密码算法相关函数的语法、说明、参数、返回值和示例。

aes_new

函数详细信息，请参见下表：

项目	描述
语法	`aes_new(config)` 。
说明	创建AES对象，用于后续的 `aes_enc()` 加密和 `aes_dec()` 解密。
参数	`config` 参数为字典类型，包含如下参数： • （必选）key：密钥，字符串类型。 • （可选）salt：盐值，字符串类型。 • （必选）cipher_len：密码器长度。取值范围：128、192和256。 • （必选）cipher_mode：密码器模式。取值范围：ecb、cbc、ctr、cfb和ofb。 • （可选）iv：初始向量，字符串类型。
返回值	成功返回AES对象（字典类型），失败返回 `false` 。

项目	描述

示例	<pre>aes_conf = []
	plaintext = ''
	if and($http_mode, eq($http_mode, 'ecb-128')) {
	    set(aes_conf, 'key', 'ab8bfd9f-a1af-4ba2-bbb0-1ee520e3d8bc')
	    set(aes_conf, 'salt', '1234567890')
	    set(aes_conf, 'cipher_len', 128)
	    set(aes_conf, 'cipher_mode', 'ecb')
	    plaintext = 'hello aes ecb-128'
	}
	if and($http_mode, eq($http_mode, 'cbc-256')) {
	    set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')
	    set(aes_conf, 'cipher_len', 256)
	    set(aes_conf, 'cipher_mode', 'cbc')
	    set(aes_conf, 'iv', '0123456789abcdef')
	    plaintext = 'hello aes cbc-256'
	}
	if and($http_mode, eq($http_mode, 'ofb-256')) {
	    set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')
	    set(aes_conf, 'cipher_len', 256)
	    set(aes_conf, 'cipher_mode', 'ofb')
	    set(aes_conf, 'iv', tochar(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))
	    plaintext = 'hello aes ofb-256'
	}
	aes_obj = aes_new(aes_conf)
	if not(aes_obj) {
	    say(concat('aes obj failed'))
	    exit(400)
	}
	ciphertext = aes_enc(aes_obj, plaintext)
	plaintext_reverse = aes_dec(aes_obj, ciphertext)
	say(concat('plain: ', plaintext))
	say(concat('cipher: ', tohex(ciphertext)))
	say(concat('plain_reverse: ', plaintext_reverse))
	if ne(plaintext, plaintext_reverse) {
	    say('plaintext ~= plaintext_reverse')
	    exit(400)
	}</pre>

aes_enc

函数详细信息，请参见下表：

项目	描述
语法	`aes_enc(o, s)` 。
说明	AES加密。

项目	描述
参数	• s：明文或密文。 • o：表示 `aes_new` 返回的AES对象。
返回值	返回对 `s` 加密后的密文。

示例	`aes_conf = []`
	`plaintext = ''`
	`if and($http_mode, eq($http_mode, 'ecb-128')) {`
	`  set(aes_conf, 'key', 'ab8bfd9f-a1af-4ba2-bbb0-1ee520e3d8bc')`
	`  set(aes_conf, 'salt', '1234567890')`
	`  set(aes_conf, 'cipher_len', 128)`
	`  set(aes_conf, 'cipher_mode', 'ecb')`
	`  plaintext = 'hello aes ecb-128'`
	`}`
	`if and($http_mode, eq($http_mode, 'cbc-256')) {`
	`  set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')`
	`  set(aes_conf, 'cipher_len', 256)`
	`  set(aes_conf, 'cipher_mode', 'cbc')`
	`  set(aes_conf, 'iv', '0123456789abcdef')`
	`  plaintext = 'hello aes cbc-256'`
	`}`
	`if and($http_mode, eq($http_mode, 'ofb-256')) {`
	`  set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')`
	`  set(aes_conf, 'cipher_len', 256)`
	`  set(aes_conf, 'cipher_mode', 'ofb')`
	`  set(aes_conf, 'iv', tochar(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))`
	`  plaintext = 'hello aes ofb-256'`
	`}`
	`aes_obj = aes_new(aes_conf)`
	`if not(aes_obj) {`
	`  say(concat('aes obj failed'))`
	`  exit(400)`
	`}`
	`ciphertext = aes_enc(aes_obj, plaintext)`
	`plaintext_reverse = aes_dec(aes_obj, ciphertext)`
	`say(concat('plain: ', plaintext))`
	`say(concat('cipher: ', tohex(ciphertext)))`
	`say(concat('plain_reverse: ', plaintext_reverse))`
	`if ne(plaintext, plaintext_reverse) {`
	`  say('plaintext ~= plaintext_reverse')`
	`  exit(400)`
	`}`

aes_dec

函数详细信息，请参见下表：

项目	描述
语法	`aes_dec(o, s)` 。
说明	AES解密。
参数	• s：密文。 • o：表示 `aes_new` 返回的AES对象。
返回值	返回对 `s` 解密后的明文。

示例	`aes_conf = []`
	`plaintext = ''`
	`if and($http_mode, eq($http_mode, 'ecb-128')) {`
	`  set(aes_conf, 'key', 'ab8bfd9f-a1af-4ba2-bbb0-1ee520e3d8bc')`
	`  set(aes_conf, 'salt', '1234567890')`
	`  set(aes_conf, 'cipher_len', 128)`
	`  set(aes_conf, 'cipher_mode', 'ecb')`
	`  plaintext = 'hello aes ecb-128'`
	`}`
	`if and($http_mode, eq($http_mode, 'cbc-256')) {`
	`  set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')`
	`  set(aes_conf, 'cipher_len', 256)`
	`  set(aes_conf, 'cipher_mode', 'cbc')`
	`  set(aes_conf, 'iv', '0123456789abcdef')`
	`  plaintext = 'hello aes cbc-256'`
	`}`
	`if and($http_mode, eq($http_mode, 'ofb-256')) {`
	`  set(aes_conf, 'key', '146ebcc8-392b-4b3a-a720-e7356f62')`
	`  set(aes_conf, 'cipher_len', 256)`
	`  set(aes_conf, 'cipher_mode', 'ofb')`
	`  set(aes_conf, 'iv', tochar(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))`
	`  plaintext = 'hello aes ofb-256'`
	`}`
	`aes_obj = aes_new(aes_conf)`
	`if not(aes_obj) {`
	`  say(concat('aes obj failed'))`
	`  exit(400)`
	`}`
	`ciphertext = aes_enc(aes_obj, plaintext)`
	`plaintext_reverse = aes_dec(aes_obj, ciphertext)`
	`say(concat('plain: ', plaintext))`
	`say(concat('cipher: ', tohex(ciphertext)))`
	`say(concat('plain_reverse: ', plaintext_reverse))`
	`if ne(plaintext, plaintext_reverse) {`
	`  say('plaintext ~= plaintext_reverse')`
	`  exit(400)`
	`}`

sha1

函数详细信息，请参见下表：

项目	描述
语法	`sha1(s)` 。
说明	计算SHA1摘要。
参数	s：待计算摘要的字符串。
返回值	返回SHA1摘要的二进制形式。

示例	`digest = sha1('hello sha')`
	`say(concat('sha1:', tohex(digest)))`
	输出：
	`sha1:853789bc783a6b573858b6cc9f913afe82962956`

sha2

函数详细信息，请参见下表：

项目	描述
语法	`sha2(s, l)` 。
说明	计算SHA2摘要。
参数	• s：待计算摘要的字符串。 • l：指明SHA2长度。取值范围：224、256、384和512。
返回值	使用SHA2摘要的二进制形式。

项目	描述

示例	`digest = sha2('hello sha2', 224)`
	`say(concat('sha2-224:', tohex(digest)))`
	`digest = sha2('hello sha2', 256)`
	`say(concat('sha2-256:', tohex(digest)))`
	`digest = sha2('hello sha2', 384)`
	`say(concat('sha2-384:', tohex(digest)))`
	`digest = sha2('hello sha2', 512)`
	`say(concat('sha2-512:', tohex(digest)))`
	输出：
	`sha2-224:b24b7effcf53ce815ee7eb73c7382613aba1c334e2a1622655362927`
	`sha2-256:af0425cee23c236b326ed1f008c9c7c143a611859a11e87d66d0a4c3217c7792`
	`sha2-384:bebbdde9efabd4b9cf90856cf30e0b024dd13177d9367d2dcf8d7a04e059f92260f16b21e261358c2271be32086ef35b`
	`sha2-512:a1d1aef051c198c0d26bc03500c177a315fa248cea815e04fbb9a75e5be5061617daab311c5e3d0b215dbfd4e83e73f23081242b0143dcdfce5cd92ec51394f7`

hmac

函数详细信息，请参见下表：

项目	描述
语法	`hmac(k, s, v)` 。
说明	计算HMAC类算法摘要。
参数	• k：算法密钥。 • s：待计算摘要的字符串。 • v：算法版本。取值范围： md5、sha256和sha512。
返回值	使用对应算法HMAC摘要的二进制形式。

项目	描述

示例	`k = '146ebcc8-392b-4b3a-a720-e7356f62f87b'`
	`v = 'hello mac'`
	`say(concat('hmac(md5): ', tohex(hmac(k, v, 'md5'))))`
	`say(concat('hmac(sha1): ', tohex(hmac(k, v, 'sha1'))))`
	`say(concat('hmac(sha256): ', tohex(hmac(k, v, 'sha256'))))`
	`say(concat('hmac(sha512): ', tohex(hmac(k, v, 'sha512'))))`
	`say(concat('hmac_sha1(): ', tohex(hmac_sha1(k, v))))`
	输出：
	`hmac(md5): 358cbfca8ad663b547c83748de2ea778`
	`hmac(sha1): 5555633cef48c3413b68f9330e99357df1cc3d93`
	`hmac(sha256):`
	`7a494543cad3b92ce1e7c4bbc86a8f5212b53e4d661f7830f455847540a85771`
	`hmac(sha512):`
	`59d7c07996ff675b45bd5fd40a6122bb5f40f597357a9b4a9e29da6f5c7cb80679`
	`8c016fe09cb46457b6df9717d26d0af19896f72eaf4296be03e3681fea59ad`
	`hmac_sha1(): 5555633cef48c3413b68f9330e99357df1cc3d93`

hmac_sha1

函数详细信息，请参见下表：

项目	描述
语法	`hmac_sha1(k, s)` 。
说明	计算HMAC-SHA-1摘要。
参数	• s：待计算摘要的字符串。 • k：HMAC-SHA-1密钥。
返回值	返回HMAC-SHA-1摘要的二进制形式。

项目	描述

示例	`k = '146ebcc8-392b-4b3a-a720-e7356f62f87b'`
	`v = 'hello mac'`
	`say(concat('hmac(md5): ', tohex(hmac(k, v, 'md5'))))`
	`say(concat('hmac(sha1): ', tohex(hmac(k, v, 'sha1'))))`
	`say(concat('hmac(sha256): ', tohex(hmac(k, v, 'sha256'))))`
	`say(concat('hmac(sha512): ', tohex(hmac(k, v, 'sha512'))))`
	`say(concat('hmac_sha1(): ', tohex(hmac_sha1(k, v))))`
	输出：
	`hmac(md5): 358cbfca8ad663b547c83748de2ea778`
	`hmac(sha1): 5555633cef48c3413b68f9330e99357df1cc3d93`
	`hmac(sha256):`
	`7a494543cad3b92ce1e7c4bbc86a8f5212b53e4d661f7830f455847540a85771`
	`hmac(sha512):`
	`59d7c07996ff675b45bd5fd40a6122bb5f40f597357a9b4a9e29da6f5c7cb80679`
	`8c016fe09cb46457b6df9717d26d0af19896f72eaf4296be03e3681fea59ad`
	`hmac_sha1(): 5555633cef48c3413b68f9330e99357df1cc3d93`

md5

函数详细信息，请参见下表：

项目	描述
语法	`md5(s)` 。
说明	计算MD5摘要。
参数	s：待计算摘要的字符串。
返回值	返回MD5摘要的十六进制形式。

示例	`say(concat('md5: ', md5('hello md5')))`
	输出：
	`md5: 741fc6b1878e208346359af502dd11c5`

md5_bin

函数详细信息，请参见下表：

项目	描述
语法	`md5_bin(s)` 。
说明	计算MD5摘要。
参数	s：待计算摘要的字符串。

项目	描述
返回值	返回MD5摘要的二进制形式。

示例	`say(concat('md5_bin: ', tohex(md5_bin('hello md5))))`
	输出：
	`md5_bin: 741fc6b1878e208346359af502dd11c5`

6.11. JSON相关

本文为您介绍JSON相关函数的语法、说明、参数、返回值和示例。

json_enc

函数详细信息，请参见下表：

项目	描述
语法	`json_enc(d)` 。
说明	JSON编码。
参数	d：待编码的字典对象。
返回值	成功返回编码后的字符串，失败返回 `false` 。

示例	`var_a = []`
	`var_b = ['v1', 'v2']`
	`set(var_a, 'k1', 'v1')`
	`set(var_a, 'k2', var_b)`
	`var_c = '{"k1":"v1","k2":["v1","v2"]}'`
	`say(concat('json_enc=', json_enc(var_a)))`
	`say(concat('json_dec=', get(json_dec(var_c), 'k1')))`
	输出：
	`json_enc={"k1":"v1","k2":["v1","v2"]}`
	`json_dec=v1`

json_dec

函数详细信息，请参见下表：

项目	描述
语法	`json_dec(s)` 。
说明	JSON解码。
参数	s：待解码的JSON格式字符串。

项目	描述
返回值	成功返回解码后的字典，失败返回 `false` 。

示例	`var_a = []`
	`var_b = ['v1', 'v2']`
	`set(var_a, 'k1', 'v1')`
	`set(var_a, 'k2', var_b)`
	`var_c = '{"k1":"v1","k2":["v1","v2"]}'`
	`say(concat('json_enc=', json_enc(var_a)))`
	`say(concat('json_dec=', get(json_dec(var_c), 'k1')))`
	输出：
	`json_enc={"k1":"v1","k2":["v1","v2"]}`
	`json_dec=v1`

6.12. Misc相关

本文为您介绍Misc相关函数的语法、说明、参数、返回值和示例。

base64_enc

函数详细信息，请参见下表：

项目	描述
语法	`base64_enc(s [, no_padding])` 。
说明	base64编码。
参数	• s：待编码的字符串。 • no_padding： `true` 表示无填充，默认 `false` 。
返回值	base64编码后的字符串。

示例	`if $http_data {`
	`  decdata = base64_dec($http_data)`
	`  say(concat('base64_decdata=', decdata))`
	`  say(concat('base64_encdata=', base64_enc('hello, dsl')))`
	`}`
	请求`header: "data: aGVsbG8sIGRzbA=="`
	响应：`base64_decdata=hello, dsl`
	`base64_encdata=aGVsbG8sIGRzbA==`

base64_dec

函数详细信息，请参见下表：

项目	描述
语法	`base64_dec(s)` 。
说明	base64解码。
参数	s：待解码的字符串。
返回值	base64解码后的字符串。

示例	`if $http_data {`
	`  decdata = base64_dec($http_data)`
	`  say(concat('base64_decdata=', decdata))`
	`  say(concat('base64_encdata=', base64_enc('hello, dsl')))`
	`}`
	请求header: "data: aGVsbG8sIGRzbA=="
	响应： base64_decdata=hello, dsl
	base64_encdata=aGVsbG8sIGRzbA==

url_escape

函数详细信息，请参见下表：

项目	描述
语法	`url_escape(s)` 。
说明	URL编码。
参数	s：待编码的字符串。
返回值	URL编码后的字符串。

示例	`raw = '/abc/123/ dd/file.m3u8'`
	`esdata = url_escape(raw)`
	`dsdata = url_unescape(esdata)`
	`if eq(raw, dsdata) {`
	`  say(concat('raw=', raw))`
	`  say(concat('dsdata=', dsdata))`
	`}`
	输出： raw=/abc/123/ dd/file.m3u8
	esdata=%2Fabc%2F123%2F%20dd%2Ffile.m3u8
	dsdata=/abc/123/ dd/file.m3u8

url_unescape

函数详细信息，请参见下表：

项目	描述
语法	`url_unescape(s)` 。
说明	URL解码。
参数	s：待解码的字符串。
返回值	URL解码后的字符串。

示例	`raw = '/abc/123/ dd/file.m3u8'`
	`esdata = url_escape(raw)`
	`dsdata = url_unescape(esdata)`
	`if eq(raw, dsdata) {`
	`  say(concat('raw=', raw))`
	`  say(concat('dsdata=', dsdata))`
	`}`
	
	输出：
	`raw=/abc/123/ dd/file.m3u8`
	`esdata=%2Fabc%2F123%2F%20dd%2Ffile.m3u8`
	`dsdata=/abc/123/ dd/file.m3u8`

rand

函数详细信息，请参见下表：

项目	描述
语法	`rand(n1, n2)` 。
说明	生成随机数，随机数范围：n1 <= 返回值 <= n2。
参数	• n1：随机数下限。 • n2：随机数上限。
返回值	返回生成的随机数。
示例	`r = rand(1,100)`

rand_hit

函数详细信息，请参见下表：

项目	描述
语法	`rand_hit(ratio)` 。
说明	按指定概率返回真假。
参数	ratio：为真概率，有效值范围为[0-100]。
返回值	按ratio概率返回 `true` 。例如：当ratio为100时，返回 `true` ，当ratio为0时，返回 `false` 。
示例	`rand_hit(80)`

crc

函数详细信息，请参见下表：

项目	描述
语法	`crc(s)` 。
说明	计算crc摘要。
参数	s：待计算摘要的字符串。
返回值	返回 `s` 的crc摘要。
示例	`crc('hello edgescript')`

tonumber

函数详细信息，请参见下表：

项目	描述
语法	`tonumber(s [, base])` 。
说明	类型转换，将字符串类型转换为数字类型。
参数	• s：待转换的字符串。 • base：可指定待转换目标的进制，可用值：10和16，默认10进制。

项目	描述

示例	`n = tonumber('100')`
	`say(concat('tonumber()=', n))`
	输出： `tonumber()=100`

base64_enc_safe

函数详细信息，请参见下表：

项目	描述
语法	`base64_enc_safe(str)` 。
说明	对输入的字符串进行Base64安全编码。安全编码后输出时，需要将“+”替换成“-”、“/”替换成“_”，同时去掉编码后的“=”。
参数	str：待加密的字符串。
返回值	返回字符串类型

示例	`add_rsp_header('X-RESPOND-OUTPUT', concat('base64_enc_safe=', base64_enc_safe('hello, dsl')), true)`
	
	输出响应头：
	`X-RESPOND-OUTPUT: base64_enc_safe=aGVsbG8sIGRzbA`

base64_dec_safe

函数详细信息，请参见下表：

项目	描述
语法	`base64_dec_safe(str)` 。
说明	对输入的字符串进行Base64安全解码。安全解码后输出时，需要将“-”替换成“+”、“_”替换成“/”，末尾用“=”按照4的余数补齐。
参数	str：Base64加密后的内容。
返回值	返回字符串类型。

项目	描述

示例	`add_rsp_header('X-RESPOND-OUTPUT', concat('base64_dec_safe=', base64_dec_safe(base64_enc_safe('hello, dsl'))), true)`
	
	输出响应头：
	`X-RESPOND-OUTPUT:base64_dec_safe=hello, dsl`

randomseed

函数详细信息，请参见下表：

项目	描述
语法	`randomseed()` 。
说明	指定生成随机数种子。
参数	无。
返回值	无。
示例	`randomseed()` `r = rand(1,100)`

6.13. 数组类型相关函数

本文为您介绍数组类型相关函数的语法、参数、示例和返回值。

arr_concat

使用arr_concat将字段表中的字符串按照指定的字符进行连接，函数的详细信息见下表。

项目	描述
语法	arr_concat(tbl, sep)
参数	• tbl：数组变量。 • sep（可选参数）：拼接符，默认为空串拼接。
示例	`d = ['t1','t2','t3']` `say(arr_concat(d, '&'))`

项目	描述
返回值	返回拼接后的字符串，返回值为字符类型。本示例的返回值为 `t1&t2&t3` 。

arr_insert

使用arr_insert向数组中插入值，函数的详细信息见下表。

项目	描述
语法	arr_insert(list, value, [pos])
参数	• list：数组类型。 • value：任意类型。 • pos：number类型，不能为0。list索引默认从1开始，在list的pos处插入元素value，并将元素pos+1后移至尾部元素；如果未指定pos，默认从末尾插入。

示例	`tbl_1 = []`
	`arr_insert(tbl_1, '1')`
	`arr_insert(tbl_1, '3')`
	`arr_insert(tbl_1, '5')`
	`arr_insert(tbl_1, '2')`
	`arr_insert(tbl_1, '6', 1)`
	`str = arr_concat(tbl_1, '')`
	`say(concat('arr_insert:', str))`

返回值	始终返回 `true` 。本示例的返回值为 `arr_insert:61352` 。

arr_remove

使用arr_remove移除list中指定位置的元素并返回被移除元素的值，无pos时返回尾部位置的元素对应的值，函数的详细信息见下表。

项目	描述
语法	arr_remove(list, [pos])
参数	• list：数组类型。 • pos：number类型。

示例	`tbl_1 = []`
	`arr_insert(tbl_1, '1')`
	`arr_insert(tbl_1, '3')`
	`arr_insert(tbl_1, '5')`
	`arr_insert(tbl_1, '2')`
	`say(concat('arr_remove:', arr_remove(tbl_1, 2)))`

项目	描述
返回值	返回值为被移除位置的数据。本示例的返回值为 `arr_remove:3` 。

arr_sort

使用arr_sort对list元素从索引的头部到索引的尾部按指定次序排序。

- 如果提供了comp，comp必须是一个可以接收两个列表内元素为参数的函数。当第一个元素需要排在第二个元素之前时，返回true。
- 如果未提供comp，则按字符的ANSII码从小到大排序。这种排序算法不稳定，即当两个元素次序相等时，这两个元素在排序后的相对位置可能会改变。

函数的详细信息见下表。

项目	描述
语法	arr_sort(list, [comp])
参数	• list：数组类型。 • comp：自定义排序算法函数，该参数的类型为函数。

示例	`tbl_1 = []`
	`arr_insert(tbl_1, '1')`
	`arr_insert(tbl_1, '3')`
	`arr_insert(tbl_1, '5')`
	`arr_insert(tbl_1, '2')`
	`say(concat('remove:', arr_remove(tbl_1, 2)))`
	`str = arr_concat(tbl_1, '')`
	`say(concat('insert:', str))`
	`arr_sort(tbl_1)`
	`str = arr_concat(tbl_1, '')`
	`say(concat('sort:', str))`
	`def my_comp(a, b){`
	`  a = tonumber(a)`
	`  b = tonumber(b)`
	`  if gt(a, b) {`
	`  return true`
	`  }`
	`  return false`
	`}`
	`arr_sort(tbl_1, my_comp)`
	`str = arr_concat(tbl_1, '')`
	`say(concat('sort_comp:', str))`

项目	描述

返回值	始终返回 `true` 。本示例的返回值如下：
	`remove:3`
	`insert:152`
	`sort:125`
	`sort_comp:521`

arr_len

使用arr_len计算数组的元素个数，函数的详细信息见下表。

项目	描述
语法	arr_len(arr)
参数	arr：数组变量。

示例	`d = []`
	`set(d, 1, 'v1')`
	`say(arr_len(d))`

返回值	返回值为数字类型，本示例的返回值为 `1` 。

6.14. 请求判断相关函数

本文为您介绍请求判断相关函数的语法、参数、示例和返回值。

server_addr

使用server_addr返回接收了当前请求的服务器地址，函数的详细信息见下表。

项目	描述
语法	server_addr()
参数	无
示例	`s_addr = server_addr()` `say(concat('s_addr:', s_addr))`
返回值	返回服务器地址，返回值为字符串类型。

server_port

使用server_port返回接收了当前请求的服务器端口，函数的详细信息见下表。

项目	描述
语法	server_port()
参数	无
示例	`s_addr = server_addr()` `say(concat('s_addr:', s_addr))`
返回值	返回服务器端口，返回值为数字类型。

client_addr

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用client_addr返回客户端IP地址，函数的详细信息见下表。

项目	描述
语法	client_addr()
参数	无

示例	`c_addr = client_addr()`
	`c_port = client_port()`
	`say(concat('c_addr:', c_addr))`
	`say(concat('c_port:', tostring(c_port)))`

返回值	返回客户端IP地址，返回值为字符串类型。

client_port

使用client_port返回客户端的端口，函数的详细信息见下表。

项目	描述
语法	client_port()
参数	无

示例	`c_addr = client_addr()`
	`c_port = client_port()`
	`say(concat('c_addr:', c_addr))`
	`say(concat('c_port:', tostring(c_port)))`

项目	描述
返回值	返回客户端的端口，返回值为数字类型。

client_country

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用client_country返回客户端IP的国家编码，函数的详细信息见下表。

项目	描述
语法	client_country()
参数	无

示例	`c_country = client_country()`
	`c_region = client_region()`
	`c_city = client_city()`
	`c_isp = client_isp()`
	`if c_country {`
	`  say(concat('client_country:', c_country))`
	`}`
	`if c_region {`
	`  say(concat('client_region:', c_region))`
	`}`
	`if c_city {`
	`  say(concat('client_city:', c_city))`
	`}`
	`if c_isp {`
	`  say(concat('client_isp:', c_isp))`
	`}`

返回值	返回客户端IP的国家编码，返回值为字符串类型。国家编码信息，请参见国家编码表。

client_region

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用client_region返回客户端的地区编码，函数的详细信息见下表。

项目	描述
语法	client_region()
参数	无

项目	描述

示例	<pre>c_country = client_country()
	c_region = client_region()
	c_city = client_city()
	c_isp = client_isp()
	if c_country {
	    say(concat('client_country:', c_country))
	}
	if c_region {
	    say(concat('client_region:', c_region))
	}
	if c_city {
	    say(concat('client_city:', c_city))
	}
	if c_isp {
	    say(concat('client_isp:', c_isp))
	}</pre>

返回值	返回客户端的地区编码，返回值为字符串类型。地区编码信息，请参见地区编码表。

client_city

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用client_city返回客户端的城市编码，函数的详细信息见下表。

项目	描述
语法	client_city()
参数	无

项目	描述

示例	<pre>c_country = client_country()
	c_region = client_region()
	c_city = client_city()
	c_isp = client_isp()
	if c_country {
	  say(concat('client_country:', c_country))
	}
	if c_region {
	  say(concat('client_region:', c_region))
	}
	if c_city {
	  say(concat('client_city:', c_city))
	}
	if c_isp {
	  say(concat('client_isp:', c_isp))
	}</pre>

返回值	返回客户端的城市编码，返回值为字符串类型。城市编码信息，请参见城市编码表。

client_isp

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用client_isp返回客户端的运营商编码，函数的详细信息见下表。

项目	描述
语法	client_isp()
参数	无

项目	描述

示例	<pre>c_country = client_country()
	c_region = client_region()
	c_city = client_city()
	c_isp = client_isp()
	if c_country {
	  say(concat('client_country:', c_country))
	}
	if c_region {
	  say(concat('client_region:', c_region))
	}
	if c_city {
	  say(concat('client_city:', c_city))
	}
	if c_isp {
	  say(concat('client_isp:', c_isp))
	}</pre>

返回值	返回客户端的运营商编码，返回值为字符串类型。运营商编码信息，请参见运营商编码表。

ip_country

注意 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用ip_country返回指定IP的国家编码，函数的详细信息见下表。

项目	描述
语法	ip_country(ipaddr)
参数	ipaddr：点分十进制的IP字符串。

项目	描述

示例	
	`c_country = ip_country('192.168.0.1')`
	`c_region = ip_region('192.168.0.1')`
	`c_city = ip_city('192.168.0.1')`
	`c_isp = ip_isp('192.168.0.1')`
	`if c_country {`
	`  say(concat('ip_country:', c_country))`
	`}`
	`if c_region {`
	`  say(concat('ip_region:', c_region))`
	`}`
	`if c_city {`
	`  say(concat('ip_city:', c_city))`
	`}`
	`if c_isp {`
	`  say(concat('ip_isp:', c_isp))`
	`}`

返回值	返回指定IP的国家编码，返回值为字符串类型。国家编码信息，请参见国家编码表。

ip_region

> **注意** 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用ip_region返回指定IP的地区（省市）编码，函数的详细信息见下表。

项目	描述
语法	ip_region(ipaddr)
参数	ipaddr：点分十进制的IP字符串。

项目	描述

示例	<pre>c_country = ip_country('192.168.0.1')
	c_region = ip_region('192.168.0.1')
	c_city = ip_city('192.168.0.1')
	c_isp = ip_isp('192.168.0.1')
	if c_country {
	  say(concat('ip_country:', c_country))
	}
	if c_region {
	  say(concat('ip_region:', c_region))
	}
	if c_city {
	  say(concat('ip_city:', c_city))
	}
	if c_isp {
	  say(concat('ip_isp:', c_isp))
	}</pre>

返回值	返回指定IP的地区（省市）编码，返回值为字符串类型。地区编码信息，请参见地区编码表。。

ip_city

注意 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用ip_city返回指定IP的城市编码，函数的详细信息见下表。

项目	描述
语法	ip_city(ipaddr)
参数	ipaddr：点分十进制的IP字符串。

项目	描述

示例	
	`c_country = ip_country('192.168.0.1')`
	`c_region = ip_region('192.168.0.1')`
	`c_city = ip_city('192.168.0.1')`
	`c_isp = ip_isp('192.168.0.1')`
	`if c_country {`
	`  say(concat('ip_country:', c_country))`
	`}`
	`if c_region {`
	`  say(concat('ip_region:', c_region))`
	`}`
	`if c_city {`
	`  say(concat('ip_city:', c_city))`
	`}`
	`if c_isp {`
	`  say(concat('ip_isp:', c_isp))`
	`}`

返回值	返回客户端的城市编码，返回值为字符串类型。城市编码信息，请参见城市编码表。

ip_isp

注意 风险提示：因运营商网络NAT策略导致客户端地址被修改，或者IP不在CDN地址库的情况下，会影响该接口真实的返回值，请谨慎使用。

使用ip_isp返回指定IP的运营商编码，函数的详细信息见下表。

项目	描述
语法	ip_isp(ipaddr)
参数	ipaddr：点分十进制的IP字符串。

项目	描述

示例	<pre>c_country = ip_country('192.168.0.1')
	c_region = ip_region('192.168.0.1')
	c_city = ip_city('192.168.0.1')
	c_isp = ip_isp('192.168.0.1')
	if c_country {
	  say(concat('ip_country:', c_country))
	}
	if c_region {
	  say(concat('ip_region:', c_region))
	}
	if c_city {
	  say(concat('ip_city:', c_city))
	}
	if c_isp {
	  say(concat('ip_isp:', c_isp))
	}</pre>

返回值	返回指定IP的运营商编码，返回值为字符串类型。运营商编码信息，请参见运营商编码表。

req_uri

req_uri函数的说明如下：

- 如果无pattern参数，则返回请求URI，不包含参数部分。
- 如果有pattern参数，则针对请求URI进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_uri([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

示例	<pre># req_uri
	say(concat('req_uri: ', req_uri()))
	if req_uri('/path1/path2') {
	  say('req_uri: plain match')
	}
	if req_uri('re:/path[0-9]/path[0-9]') {
	  say('req_uri: regex match')
	}</pre>

项目	描述

返回值	• 无pattern参数，返回请求URI。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `/path1/path2?mode=ip`
	响应：
	`req_uri: /path1/path2`
	`req_uri: plain match`
	`req_uri: regex match`

req_uri_basename

req_uri_basename函数的说明如下：

- 如果无pattern参数，则返回请求URI中的文件名部分。
- 如果有pattern参数，则针对请求URI中的文件名部分进行匹配判断。

文件名部分示例如下：

- 示例1：对于/document_detail/30360.html，文件名部分为30360。
- 示例2：对于/M604/guopei_mp4/ZYJY2017BJGL0101/2-1_g.mp4，文件名部分为2-1_g。
- 示例3：对于/tarball/foo.tar.bz2，文件名部分为foo。

函数的详细信息见下表。

项目	描述
语法	req_uri_basename([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

示例	`# req_uri_basename`
	`basename = req_uri_basename()`
	`say(concat('req_uri_basename: ', basename, ' ', len(basename)))`
	`if req_uri_basename('foo') {`
	`  say('req_uri_basename: plain match')`
	`}`
	`if req_uri_basename('re:^f.*') {`
	`  say('req_uri_basename: regex match')`
	`}`

项目	描述

返回值	• 无pattern参数，返回请求URI中的文件名部分。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `/path1/path2/foo.tar.bz2`
	响应：
	`req_uri_basename: foo 3`
	`req_uri_basename: plain match`
	`req_uri_basename: regex match`

req_uri_ext

req_uri_ext函数的说明如下：

- 如果无pattern参数，则返回请求URI中的扩展名部分。
- 如果有pattern参数，则针对请求URI中的扩展名部分进行匹配判断。

扩展名部分示例如下：

- 示例1：对于/document_detail/30360.html，扩展名部分为.html。
- 示例2：对于/M604/guopei_mp4/ZYJY2017BJGL0101/2-1_g.mp4，扩展名部分为.mp4。
- 示例3：对于/tarball/foo.tar.bz2，扩展名部分为.tar.bz2。

函数的详细信息见下表。

项目	描述
语法	req_uri_ext([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

示例	`# req_uri_ext`
	`ext = req_uri_ext()`
	`say(concat('req_uri_ext: ', ext, ' ', len(ext)))`
	`if req_uri_ext('.tar.bz2') {`
	`  say('req_uri_ext: plain match')`
	`}`
	`if req_uri_ext('re:\.tar\.bz[0-2]') {`
	`  say('req_uri_ext: regex match')`
	`}`

项目	描述

返回值	• 无pattern参数，返回请求URI中的扩展名部分。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `/path1/path2/foo.tar.bz2`
	响应：
	`req_uri_ext: .tar.bz2 8`
	`req_uri_ext: plain match`
	`req_uri_ext: regex match`

req_uri_seg

req_uri_seg函数的说明如下：

- 使用正斜线（/）对uri进行分隔并返回所有段落。
  - 如果无idx参数，返回所有段落。
  - 如果有idx参数，返回指定索引后的所有段落，包含对应的索引。
- 段落索引：段落索引从头部递增，依次从左向右。
- 段落上限：段落上限为128个字符，超出上限的段落会被丢弃。

函数的详细信息见下表。

项目	描述
语法	req_uri_seg([idx])
参数	idx（可选参数）：允许指定起始索引。

项目	描述
示例	<pre>```
# req_uri_seg
def echo_each(k, v, u) {
say(concat(get(u, 'msg'), ' : segs[', k, ']=', v))
}
# fetch all segments
segs = req_uri_seg()
foreach(segs, echo_each, ['msg'='req_uri_seg()'])
# fetch segments from idx 3
segs = req_uri_seg(3)
if get(segs, 3) {
say(concat('req_uri_seg(3): segs[3]=', get(segs, 3)))
}
if get(segs, 4) {
say(concat('req_uri_seg(3): segs[4]=', get(segs, 4)))
}
if get(segs, 5) {
say(concat('req_uri_seg(3): segs[5]=', get(segs, 5)))
}
``</pre>

返回值	返回值为字典类型，包含相应段落。
	说明 从返回字典中获取指定索引段落时，必须判断是否为空。
	本示例的返回值如下：
	请求： `/path1/path2/path3/path4?mode=req2`
	响应：
	`req_uri_seg() : segs[1]=path1`
	`req_uri_seg() : segs[2]=path2`
	`req_uri_seg() : segs[3]=path3`
	`req_uri_seg() : segs[4]=path4`
	`req_uri_seg(3): segs[3]=path3`
	`req_uri_seg(3): segs[4]=path4`

req_uri_arg

使用req_uri_arg默认返回指定参数的值，如果有pattern参数，则对指定参数的值进行匹配判断，函数的详细信息见下表。

项目	描述
语法	req_uri_arg(name, [pattern])
```

项目	描述

参数	• name：参数名称。
	• pattern：使用该参数进行匹配，支持以下两种模式。
	  ○ 简单匹配：相等判断，默认为简单匹配。
	  ○ 正则匹配：`re:` 前导的正则表达式。

示例	`# req_uri_arg`
	`uid = req_uri_arg('uid')`
	`if uid {`
	`  say(concat('found uid ', uid))`
	`} else {`
	`  say('not found uid')`
	`}`
	`uid_chk = req_uri_arg('uid', '058334')`
	`if uid_chk {`
	`  say('check uid ok. plain mode')`
	`} else {`
	`  say('check uid fail. plain mode')`
	`}`
	`uid_chk = req_uri_arg('uid', 're:[0-9]+')`
	`if uid_chk {`
	`  say('check uid ok. regex mode')`
	`} else {`
	`  say('check uid fail. regex mode')`
	`}`

项目	描述

返回值	• 如果无pattern参数
	  ○ 参数存在：返回name指定参数的字符串值。
	  ○ 参数不存在：返回false。
	• 如果有pattern参数
	  ○ 参数存在：进行匹配，返回true，表示匹配成功；返回false，表示匹配失败。
	  ○ 参数不存在：返回false。
	本示例的返回值如下：
	请求： `/path1/path2/path3/path4?mode=req4&uid`
	响应：
	`not found uid`
	`check uid fail. plain mode`
	`check uid fail. regex mode`
	请求： `/path1/path2/path3/path4?mode=req4&uid=`
	响应：
	`found uid`
	`check uid fail. plain mode`
	`check uid fail. regex mode`
	请求： `/path1/path2/path3/path4?mode=req4&uid=12345`
	响应：
	`found uid 12345`
	`check uid fail. plain mode`
	`check uid ok. regex mode`

req_uri_query_string

req_uri_query_string函数的说明如下：

- 如果无pattern参数，则返回请求中的参数部分，不包含问号（？）。
- 如果有pattern参数，则针对请求中的参数部分进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_uri_query_string([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述

示例	
	``
	# req_uri_query_string
	say(concat('req_uri_query_string: ', req_uri_query_string()))
	if req_uri_query_string('mode=') {
	    say('check uri query string ok. plain mode')
	} else {
	    say('check uri query string fail. plain mode')
	}
	if req_uri_query_string('re:mode=[0-9a-z]+') {
	    say('check uri query string ok. regex mode')
	} else {
	    say('check uri query string fail. regex mode')
	}
	``

返回值	• 无pattern参数，返回请求中的参数部分。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `/path1/path2/path3/path4?mode=req5&token=34Deasd#243`
	响应：
	`req_uri_query_string: mode=req5&token=34Deasd`
	`check uri query string fail. plain mode`
	`check uri query string ok. regex mode`

req_scheme

req_scheme函数的说明如下：

- 如果无pattern参数，则返回请求scheme。
- 如果有pattern参数，则针对请求scheme进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_scheme([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述

示例	
	``
	# req_scheme
	say(concat('req_scheme: ', req_scheme()))
	if req_scheme('https') {
	    say('check scheme ok. plain mode')
	} else {
	    say('check scheme fail. plain mode')
	}
	if req_scheme('re:https?') {
	    say('check scheme ok. regex mode')
	} else {
	    say('check scheme fail. regex mode')
	}
	``

返回值	• 无pattern参数，返回请求scheme。返回值为字符串类型。 • 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。 本示例的返回值如下： ``` 请求：http://xx.. req_scheme: http check scheme fail. plain mode check scheme ok. regex mode ```

req_method

req_method函数的说明如下：

- 如果无pattern参数，则返回请求method。
- 如果有pattern参数，则针对请求method进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_method([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述

示例	`# req_method`
	`say(concat('req_method: ', req_method()))`
	`if req_method('GET') {`
	`  say('check method ok. plain mode')`
	`} else {`
	`  say('check method fail. plain mode')`
	`}`
	`if req_method('re:(GET|POST)') {`
	`  say('check method ok. regex mode')`
	`} else {`
	`  say('check method fail. regex mode')`
	`}`

返回值	• 无pattern参数，返回请求method。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `POST /xxxx/xxx`
	响应：
	`req_method: POST`
	`check method fail. plain mode`
	`check method ok. regex mode`

req_host

req_host函数的说明如下：

- 如果无pattern参数，则返回请求头Host的值。
- 如果有pattern参数，则针对请求头Host的值进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_host([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述

示例	
	`# req_host`
	`say(concat('req_host: ', req_host()))`
	`if req_host('image.developer.aliyundoc.com') {`
	`  say('check host ok. plain mode')`
	`} else {`
	`  say('check host fail. plain mode')`
	`}`
	`if req_host('re:.+\.y\.z\.com') {`
	`  say('check host ok. regex mode')`
	`} else {`
	`  say('check host fail. regex mode')`
	`}`

返回值	• 无pattern参数，返回请求头Host的值。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `Host: image.developer.aliyundoc.com`
	响应：
	`req_host: image.developer.aliyundoc.com`
	`check host fail. plain mode`
	`check host ok. regex mode`

req_user_agent

req_user_agent函数的说明如下：

- 如果无pattern参数，则返回请求头User-Agent的值。
- 如果有pattern参数，则针对请求头User-Agent的值进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_user_agent([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述

示例	
	``
	# req_user_agent
	say(concat('req_user_agent: ', req_user_agent()))
	if req_user_agent('Mozilla') {
	    say('check user_agent ok. plain mode')
	} else {
	    say('check user_agent fail. plain mode')
	}
	if req_user_agent('re:^Mozilla') {
	    say('check user_agent ok. regex mode')
	} else {
	    say('check user_agent fail. regex mode')
	}
	``

返回值	• 无pattern参数，返回请求头User-Agent的值。返回值为字符串类型。 • 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。 本示例的返回值如下： **请求：** `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)` **响应：** ``` req_user_agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) check user_agent fail. plain mode check user_agent ok. regex mode ```

req_referer

req_referer函数的说明如下：

- 如果无pattern参数，则返回请求头Referer的值。
- 如果有pattern参数，则针对请求头Referer的值进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_referer([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

项目	描述
示例	<pre>```
# req_referer
say(concat('req_referer: ', req_referer()))
if req_referer('https://example.aliyundoc.com/******00003') {
say('check referer ok. plain mode')
} else {
say('check referer fail. plain mode')
}
if req_referer('re:https://foo\.bar\.cn/*+[0-9]+') {
say('check referer ok. regex mode')
} else {
say('check referer fail. regex mode')
}
``</pre>

返回值	• 无pattern参数，返回请求头Referer的值。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `Referer: https://example.aliyundoc.com/******00003`
	响应：
	`req_referer: https://example.aliyundoc.com/******00003`
	`check referer ok. plain mode`
	`check referer ok. regex mode`

req_cookie

使用req_cookie默认返回指定cookie的值，如果有pattern参数，则对指定cookie的值进行匹配判断，函数的详细信息见下表。

项目	描述
语法	req_cookie(name, [pattern])

参数	• name：cookie名称。
	• pattern：使用该参数进行匹配，支持以下两种模式。
	  ○ 简单匹配：相等判断，默认为简单匹配。
	  ○ 正则匹配：`re:` 前导的正则表达式。

```

项目	描述
示例	<pre>```
# req_cookie
uid = req_cookie('uid')
if uid {
say(concat('found cookie uid ', uid))
} else {
say('not found cookie uid')
}
uid_chk = req_cookie('uid', '058334')
if uid_chk {
say('check cookie uid ok. plain mode')
} else {
say('check cookie uid fail. plain mode')
}
uid_chk = req_cookie('uid', 're:^[0-9]+')
if uid_chk {
say('check cookie uid ok. regex mode')
} else {
say('check cookie uid fail. regex mode')
}
``</pre>
```

项目	描述

返回值	• 如果无pattern参数
	  ○ 参数存在：返回name指定cookie的字符串值。
	  ○ 参数不存在：返回false。
	• 如果有pattern参数
	  ○ 参数存在：进行匹配，返回true，表示匹配成功；返回false，表示匹配失败。
	  ○
	本示例的返回值如下：
	请求： `Cookie: uid=123456; token=value2`
	响应：
	`found cookie uid 123456`
	`check cookie uid fail. plain mode`
	`check cookie uid ok. regex mode`

req_first_x_forwarded

req_first_x_forwarded函数的说明如下：

- 如果无pattern参数，则返回请求头X-Forwarded-For中的第一个地址。
- 如果有pattern参数，则针对请求头X-Forwarded-For中的第一个地址进行匹配判断。

函数的详细信息见下表。

项目	描述
语法	req_first_x_forwarded_addr([pattern])
参数	pattern：使用该参数进行匹配，支持以下两种模式。 • 简单匹配：相等判断，默认为简单匹配。 • 正则匹配：`re:` 前导的正则表达式。

示例	`# req_first_x_forwarded`
	`say(concat('req_first_x_forwarded: ', req_first_x_forwarded()))`
	`if req_first_x_forwarded('1.1.1.1') {`
	`  say('check first_x_forwarded ok. plain mode')`
	`} else {`
	`  say('check first_x_forwarded fail. plain mode')`
	`}`
	`if req_first_x_forwarded('re:1.1.1.[0-9]') {`
	`  say('check first_x_forwarded ok. regex mode')`
	`} else {`
	`  say('check first_x_forwarded fail. regex mode')`
	`}`

项目	描述

返回值	• 无pattern参数，返回请求头X-Forwarded-For中的第一个地址。返回值为字符串类型。
	• 有pattern参数，返回true，表示匹配成功；返回false，表示匹配失败。
	本示例的返回值如下：
	请求： `X-Forwarded-For: 1.1.1.1, 10.10.10.10, 172.16.0.1`
	响应：
	`req_first_x_forwarded: 1.1.1.1`
	`check first_x_forwarded ok. plain mode`
	`check first_x_forwarded ok. regex mode`

req_header

使用req_header默认返回指定请求头的值，如果有pattern参数，则对指定请求头的值进行匹配判断。函数的详细信息见下表。

项目	描述
语法	req_header(name, [pattern])

参数	• name：请求头名称。
	请求头名称中出现的短划线（-），需要使用下划线（_）替代。例如：X-USER-ID对应为x_user_id。
	• pattern：使用该参数进行匹配，支持以下两种模式。
	  ○ 简单匹配：相等判断，默认为简单匹配。
	  ○ 正则匹配：`re:` 前导的正则表达式。

项目	描述
示例	<pre>```
# req_header
uid = req_header('x_uid')
if uid {
say(concat('found header x-uid ', uid))
} else {
say('not found header x-uid')
}
uid_chk = req_header('x_uid', 'es developer')
if uid_chk {
say('check header x-uid ok. plain mode')
} else {
say('check header x-uid fail. plain mode')
}
uid_chk = req_header('x_uid', 're:es [a-z]+')
if uid_chk {
say('check header x-uid ok. regex mode')
} else {
say('check header x-uid fail. regex mode')
}
``</pre>

返回值	• 如果无pattern参数
	  ○ 参数存在：返回name指定请求头的字符串值。
	  ○ 参数不存在：返回false。
	• 如果有pattern参数
	  ○ 参数存在：进行匹配，返回true，表示匹配成功；返回false，表示匹配失败。
	  ○
	本示例的返回值如下：
	请求： `X-UID: es developer`
	响应：
	`found header x-uid es developer`
	`check header x-uid ok. plain mode`
	`check header x-uid ok. regex mode`

req_id

使用req_id获取每个请求唯一的标识（EagleeyeID），用于标识对应的请求。函数的详细信息见下表。

项目	描述
语法	req_id()
参数	无
```

项目	描述
示例	`# req_id` `say(concat('req_id: ', req_id()))`
返回值	返回字符串类型的请求ID。本示例的返回值为 `req_id: 6451c43d15815890089411000e` 。

地区编码表

6.15. 附录

本文汇总了国家、地区、运营商编码供您查询及使用。

国家编码表

```
AD 安道尔共和国
AE 阿拉伯联合酋长国
AF 阿富汗
AG 安提瓜和巴布达
AI 安圭拉
AL 阿尔巴尼亚
AM 亚美尼亚
AN 荷属安的列斯
AO 安哥拉
AQ 南极洲
AR 阿根廷
ASC 阿森松
AS 东萨摩亚
AT 奥地利
AU 澳大利亚
AW 阿鲁巴
AX 奥兰群岛
AZ 阿塞拜疆
BA 波黑
BB 巴巴多斯
BD 孟加拉国
BE 比利时
BF 布基纳法索
BG 保加利亚
BH 巴林
BI 布隆迪
BJ 贝宁
BL 圣巴泰勒米岛
BM 百慕大群岛
BN 文莱
BO 玻利维亚
BQ 博内尔岛
BR 巴西
BS 巴哈马
BT 不丹
```

```
BV 布维岛
BW 博茨瓦纳
BY 白俄罗斯
BZ 伯利兹
CA 加拿大
CC 科科斯群岛
CD 刚果民主共和国
CF 中非共和国
CG 刚果共和国
CH 瑞士
CI 科特迪瓦
CK 库克群岛
CL 智利
CM 喀麦隆
CN 中国（内地）
CO 哥伦比亚
CR 哥斯达黎加
CV 佛得角
CW 库拉索
CX 圣诞岛（英属）
CY 塞浦路斯
CZ 捷克
DE 德国
DJ 吉布提
DK 丹麦
DM 多米尼克
DO 多米尼加共和国
DQ 贾维斯岛
DZ 阿尔及利亚
EAZ 桑给巴尔
EC 厄瓜多尔
EE 爱沙尼亚
EG 埃及
EH 西萨摩亚
ER 厄立特里亚
ES 西班牙
ET 埃塞俄比亚
EU 欧洲
FI 芬兰
FJ 斐济
FK 福克兰群岛
FM 密克罗尼西亚
FO 法罗群岛
FR 法国
FX 法属美特罗波利坦
GA 加蓬
GBA 奥尔德尼岛
GB 英国
GD 格林纳达
GE 格鲁吉亚
GF 法属圭亚那
GG 根西岛
GH 加纳
GI 直布罗陀
```

```
GL 格陵兰岛
GM 冈比亚
GN 几内亚
GO 格洛里厄斯群岛
GP 瓜德罗普
GQ 赤道几内亚
GR 希腊
GS 南乔治亚岛和南桑威奇群岛（英属）
GT 危地马拉
GU 关岛
GW 几内亚比绍
GY 圭亚那
HM 赫德岛和麦克唐纳群岛
HN 洪都拉斯
HR 克罗地亚
HT 海地
HU 匈牙利
IANA 未分配或者内网IP
ID 印度尼西亚
IE 爱尔兰
IL 以色列
IM 英属马恩岛
IN 印度
IO 英属印度洋领地
IQ 伊拉克
IS 冰岛
IT 意大利
JE 泽西岛
JM 牙买加
JO 约旦
JP 日本
JQ 约翰斯顿岛
JU 新胡安岛（法属）
KE 肯尼亚
KG 吉尔吉斯斯坦
KH 柬埔寨
KI 基里巴斯
KM 科摩罗
KN 圣基茨和尼维斯
KR 韩国
KS 科索沃
KW 科威特
KY 开曼群岛
KZ 哈萨克斯坦
LA 老挝
LB 黎巴嫩
LC 圣卢西亚
LI 列支敦士登
LK 斯里兰卡
LR 利比里亚
LS 莱索托
LT 立陶宛
LU 卢森堡
LV 拉脱维亚
```

```
LY 利比亚
MA 摩洛哥
MC 摩纳哥
MD 摩尔多瓦
ME 黑山
MF 法属圣马丁
MG 马达加斯加
MH 马绍尔群岛
MK 马其顿
ML 马里
MM 缅甸
MN 蒙古
MP 北马里亚纳群岛
MQ 马提尼克
MR 毛里塔尼亚
MS 蒙特塞拉特岛
MT 马耳他
MU 毛里求斯
MV 马尔代夫
MW 马拉维
MX 墨西哥
MY 马来西亚
MZ 莫桑比克
NA 纳米比亚
NC 新喀里多尼亚群岛
NE 尼日尔
NF 诺福克岛
NG 尼日利亚
NI 尼加拉瓜
NL 荷兰
NO 挪威
NP 尼泊尔
NR 瑙鲁
NU 纽埃
NZ 新西兰
OM 阿曼
PA 巴拿马
PE 秘鲁
PF 法属波利尼西亚
PG 巴布亚新几内亚
PH 菲律宾
PK 巴基斯坦
PL 波兰
PM 圣皮埃尔和密克隆群岛
PN 皮特克恩岛
PR 波多黎各
PS 巴勒斯坦
PT 葡萄牙
PW 帕劳
PY 巴拉圭
QA 卡塔尔
RE 留尼汪岛
RO 罗马尼亚
RS 塞尔维亚
```

```
RU 俄罗斯
RW 卢旺达
SA 沙特阿拉伯
SB 所罗门群岛
SC 塞舌尔
SD 苏丹
SE 瑞典
SG 新加坡
SH 海伦娜
SI 斯洛文尼亚
SJ 斯瓦尔巴岛和扬马延岛
SK 斯洛伐克
SL 塞拉利昂
SM 圣马力诺
SN 塞内加尔
SO 索马里
SR 苏里南
SS 南苏丹
ST 圣多美和普林西比
SV 萨尔瓦多
SX 圣马丁岛
SZ 斯威士兰
TC 特克斯和凯科斯群岛
TD 乍得
TF 法属南半球领地
TG 多哥
TH 泰国
TJ 塔吉克斯坦
TK 托克劳
TL 东帝汶
TM 土库曼斯坦
TN 突尼斯
TO 汤加
TR 土耳其
TT 特立尼达和多巴哥
TV 图瓦卢
TZ 坦桑尼亚
UA 乌克兰
UG 乌干达
UM 美国本土外小岛屿
US 美国
UY 乌拉圭
UZ 乌兹别克斯坦
VA 梵蒂冈
VC 圣文森特和格林纳丁斯
VE 委内瑞拉
VG 英属维尔京群岛
VI 美属维尔京群岛
VN 越南
VU 瓦努阿图共和国
WF 瓦利斯和富图纳群岛
WQ 复活岛
WS 萨摩亚
YE 也门
YT 马约特
```

```
YT 马约特
YU 南斯拉夫
ZA 南非
ZM 赞比亚
ZR 扎伊尔
ZW 津巴布韦
```

地区编码表

```
中国:
HK_01 中国香港
MO_01 中国澳门
TW_01 中国台湾
370000 山东省
450000 广西壮族自治区
310000 上海市
230000 黑龙江省
540000 西藏自治区
440000 广东省
120000 天津市
150000 内蒙古自治区
340000 安徽省
330000 浙江省
530000 云南省
630000 青海省
320000 江苏省
620000 甘肃省
420000 湖北省
410000 河南省
360000 江西省
210000 辽宁省
640000 宁夏回族自治区
110000 北京市
460000 海南省
130000 河北省
510000 四川省
500000 重庆市
610000 陕西省
220000 吉林省
140000 山西省
650000 新疆维吾尔自治区
350000 福建省
430000 湖南省
520000 贵州省
海外:
ID_30 西爪哇
JM_17 京斯敦
TR_44 马拉蒂亚省
TR_39 克尔克拉雷利省
US_CA 加利福尼亚州
TR_52 奥尔杜省
ES_53 加那利群岛
TR_08 阿尔特温省
HU_22 沃什州
```

```
PL_83 西里西亚省
ID_35 邦加-勿里洞
US_NH 新罕布什尔州
PL_74 罗兹省
TR_70 哈卡里省
MX_06 奇瓦瓦州
FR_A2 布列塔尼半岛
US_OH 俄亥俄州
FR_B9 罗纳-阿尔卑斯
US_SC 南卡罗来纳州
AT_02 克恩顿州
RU_16 楚瓦什共和国
TR_19 乔鲁姆省
US_HI 夏威夷州
IN_03 阿萨姆邦
MX_27 塔巴斯科州
US_IA 艾奥瓦州
FR_99 下诺曼底
US_TN 田纳西州
TR_45 马尼萨省
US_MS 密西西比州
TR_76 巴特曼省
AZ_35 纳希切万自治共和国
MX_31 尤卡坦州
MY_17 布城
GL_04 库雅雷哥
TR_38 开塞利省
TR_26 埃斯基谢希尔省
TR_09 艾登省
TR_22 埃迪尔内省
US_IL 伊利诺伊州
UA_17 敖德萨州
US_WI 威斯康辛州
IT_03 卡拉布里亚
ST_01 普林西比岛
TR_31 哈塔伊省
US_MI 密歇根州
US_NV 内华达州
FR_A1 勃艮第
DE_09 萨尔
US_TX 得克萨斯州
PL_80 喀尔巴阡山省
US_WY 怀俄明州
TR_82 昌克勒省
DE_01 巴登-符腾堡
TR_48 穆拉省
AF_28 扎布尔省
US_KY 肯塔基州
NZ_G2 惠灵顿
US_MO 密蘇里州
ES_55 卡斯蒂利亚-莱昂
IN_32 达曼-第乌
MD_81 Raionul Ocnita
TR_79 克勒克卡莱省
```

```
TR_86 阿尔达汉省
FR_B6 皮卡第
US_MT 蒙大拿州
AR_07 布宜诺斯艾利斯
MX_22 克雷塔羅州
TR_46 卡赫拉曼马拉什省
US_CT 康乃狄克州
UY_02 卡内洛内斯省
FR_97 阿基坦
TR_28 吉雷松省
NZ_F1 吉斯伯恩大区
UY_01 阿蒂加斯省
US_VT 佛蒙特州
ID_26 北苏门答腊省
TR_05 阿马西亚省
US_NM 新墨西哥州
MX_05 恰帕斯州
TR_32 梅尔辛省
FR_B3 南部-比利牛斯
US_VA 弗吉尼亚州
US_DE 特拉华州
FR_A9 朗格多克-鲁西永
TR_49 穆什省
SJ_21 斯瓦尔巴群岛
TR_71 科尼亚省
TR_87 巴尔滕省
US_AL 亚拉巴马州
TR_72 马尔丁省
TR_78 卡拉曼省
IN_33 果阿邦
IN_28 西孟加拉邦
FR_A4 香槟-阿登
TR_33 伊斯帕尔塔省
RU_17 达吉斯坦共和国
IT_02 巴斯利卡塔
UY_03 塞罗拉尔戈省
US_UT 犹他州
MX_16 米却肯州
TR_34 伊斯坦布尔
ID_12 南加里曼丹
US_PA 宾夕法尼亚州
AF_07 法利亚布省
US_OR 俄勒冈州
TR_07 安塔利亚省
TR_81 阿达纳省
IR_34 中央省
TR_65 凡城省
UA_08 赫尔松州
TR_75 阿克萨赖省
TR_12 宾格尔省
SI_39 Obcina Ivancna Gorica
AF_36 帕克蒂亚省
PE_23 Region de Tacna
MX_20 瓦哈卡州
```

```
TR_40 克尔谢希尔省
US_WA 华盛顿州
US_ND 北达科他州
IT_16 托斯卡纳
US_NJ 新泽西州
US_ME 缅因州
US_NY 纽约州
US_NC 北卡罗来纳州
US_AR 阿肯色州
PL_75 卢布林省
MX_17 莫雷洛斯州
TR_04 阿勒省
FR_B8 普罗旺斯-阿尔卑斯-蓝色海岸
MX_21 普埃布拉州
ID_14 東加里曼丹
TR_13 比特利斯省
TR_41 科贾埃利省
MX_04 坎佩切州
TR_64 乌沙克省
FR_C1 阿尔萨斯
IT_15 西西里岛
UA_18 波尔塔瓦州
US_LA 路易斯安那州
FR_A6 弗朗什孔泰
AF_02 巴德吉斯省
MX_28 塔毛利帕斯州
TR_17 恰纳卡莱省
TR_43 屈塔希亚省
ID_07 中爪哇省
TR_90 基利斯省
US_AK 阿拉斯加州
US_MD 马里兰州
TR_88 厄德尔省
FR_B7 普瓦图-夏朗德
PL_77 小波兰省
RU_89 犹太自治州
AT_08 福拉尔贝格州
HU_24 佐洛州
HU_01 巴奇-基什孔州
TR_25 埃尔祖鲁姆省
NL_06 北布拉班特省
TR_92 亚洛瓦省
TR_37 卡斯塔莫努省
TR_68 安卡拉省
TR_83 加济安泰普省
PL_72 下西里西亚省
DE_02 巴伐利亚
MX_14 哈利斯科州
TR_10 巴勒克埃西尔省
TR_14 博卢省
DE_16 柏林
TR_15 布尔杜尔省
UA_13 基辅州
MX_15 墨西哥州
PL_86 大波兰省
```

```
PL_86 大波兰省
RU_49 摩爾曼斯克州
ID_11 西加里曼丹
ST_02 圣多美岛
US_FL 佛罗里达州
LV_32 文茨皮尔斯
US_ID 爱达荷州
IR_25 塞姆南省
TR_24 埃尔津詹省
CA_07 新斯科舍
US_CO 科罗拉多州
TR_11 比莱吉克省
TR_20 代尼兹利省
TR_66 约兹加特省
AT_01 布尔根兰州
```

城市编码表

```
331100 丽水市
220200 吉林市
230400 鹤岗市
141100 吕梁市
440200 韶关市
210100 沈阳市
411500 信阳市
652100 吐鲁番地区
533100 德宏傣族景颇族自治州
511900 巴中市
341800 宣城市
310100 上海市
542200 山南地区
450300 桂林市
542600 林芝地区
440400 珠海市
420100 武汉市
431000 郴州市
445100 潮州市
361000 抚州市
650100 乌鲁木齐市
411300 南阳市
320700 连云港市
341100 滁州市
410800 焦作市
411200 三门峡市
410600 鹤壁市
440800 湛江市
341200 阜阳市
321000 扬州市
370900 泰安市
450800 贵港市
340100 合肥市
110200 北京市
420600 襄樊市
220800 白城市
```

```
654000 伊犁哈萨克自治州
211300 朝阳市
542100 昌都地区
410100 郑州市
320900 盐城市
500200 重庆市
469000 省直辖县级行政区划
430800 张家界市
130200 唐山市
530900 临沧市
370500 东营市
340700 铜陵市
340600 淮北市
210700 锦州市
140700 晋中市
450400 梧州市
440500 汕头市
451200 河池市
321300 宿迁市
231200 绥化市
370600 烟台市
632500 海南藏族自治州
533400 迪庆藏族自治州
350400 三明市
410400 平顶山市
411000 许昌市
140600 朔州市
623000 甘南藏族自治州
532600 文山壮族苗族自治州
232700 大兴安岭地区
350900 宁德市
451300 来宾市
371100 日照市
430100 长沙市
340200 芜湖市
653100 喀什地区
610800 榆林市
422800 恩施土家族苗族自治州
640400 固原市
371600 滨州市
220300 四平市
361100 上饶市
131100 衡水市
510800 广元市
330200 宁波市
420300 十堰市
150600 鄂尔多斯市
220400 辽源市
321200 泰州市
340500 马鞍山市
632100 海东地区
430700 常德市
150300 乌海市
371300 临沂市
```

```
445200 揭阳市
211400 葫芦岛市
210500 本溪市
533300 怒江傈僳族自治州
522200 铜仁地区
430500 邵阳市
local 内网IP
430200 株洲市
131000 廊坊市
441600 河源市
220600 白山市
411600 周口市
330700 金华市
632200 海北藏族自治州
451400 崇左市
632700 玉树藏族自治州
140900 忻州市
450900 玉林市
341300 宿州市
341600 亳州市
350200 厦门市
360100 南昌市
330400 嘉兴市
621200 陇南市
430600 岳阳市
611000 商洛市
510700 绵阳市
130600 保定市
513400 凉山彝族自治州
450600 防城港市
222400 延边朝鲜族自治州
430900 益阳市
530800 普洱市
130300 秦皇岛市
441500 汕尾市
431300 娄底市
370100 济南市
320500 苏州市
442000 中山市
810000 中国香港
513300 甘孜藏族自治州
610300 宝鸡市
450500 北海市
150800 巴彦淖尔市
450100 南宁市
511800 雅安市
370300 淄博市
331000 台州市
321100 镇江市
520300 遵义市
640300 吴忠市
230600 大庆市
610200 铜川市
410200 开封市
```

```
410700 新乡市
371000 威海市
510100 成都市
440900 茂名市
350100 福州市
210900 阜新市
360700 赣州市
622900 临夏回族自治州
370200 青岛市
411100 漯河市
511600 广安市
340300 蚌埠市
210800 营口市
652900 阿克苏地区
440700 江门市
330100 杭州市
430400 衡阳市
420900 孝感市
140500 晋城市
532800 西双版纳傣族自治州
230900 七台河市
360400 九江市
621000 庆阳市
370700 潍坊市
610600 延安市
520400 安顺市
TW_1017 台北
TW_1012 新北
TW_1019 桃园
TW_1015 台中
TW_1016 台南
TW_1006 高雄
TW_1003 新竹县
TW_1010 苗栗县
TW_1000 彰化县
TW_1011 南投县
TW_1021 云林县
TW_1001 嘉义县
TW_1014 屏东县
TW_1020 宜兰县
TW_1005 花莲县
TW_1018 台东县
TW_1013 澎湖县
TW_1008 金门县
TW_1009 连江县
TW_1007 基隆市
TW_1004 新竹市
TW_1002 嘉义市
350700 南平市
360300 萍乡市
320200 无锡市
360600 鹰潭市
510500 泸州市
211200 铁岭市
```

```
120200 天津市
130400 邯郸市
140800 运城市
440300 深圳市
210300 鞍山市
210400 抚顺市
230100 哈尔滨市
511500 宜宾市
152200 兴安盟
420700 鄂州市
520200 六盘水市
330600 绍兴市
441200 肇庆市
150200 包头市
420500 宜昌市
370800 济宁市
610400 咸阳市
230300 鸡西市
420800 荆门市
371500 聊城市
632600 果洛藏族自治州
220500 通化市
330300 温州市
441300 惠州市
620600 武威市
653200 和田地区
542500 阿里地区
650200 克拉玛依市
500100 重庆市
421300 随州市
150400 赤峰市
431100 永州市
652700 博尔塔拉蒙古自治州
511300 南充市
340800 安庆市
350800 龙岩市
360900 宜春市
211000 辽阳市
420200 黄石市
522300 黔西南布依族苗族自治州
654200 塔城地区
540100 拉萨市
150100 呼和浩特市
150700 呼伦贝尔市
510400 攀枝花市
210200 大连市
360800 吉安市
530700 丽江市
450200 柳州市
450700 钦州市
530100 昆明市
152900 阿拉善盟
522600 黔东南苗族侗族自治州
330500 湖州市
371700 菏泽市
```

```
445300 云浮市
130100 石家庄市
141000 临汾市
610900 安康市
654300 阿勒泰地区
511100 乐山市
630100 西宁市
620100 兰州市
441800 清远市
652800 巴音郭楞蒙古自治州
640200 石嘴山市
632800 海西蒙古族藏族自治州
542400 那曲地区
120100 天津市
421100 黄冈市
510300 自贡市
620900 酒泉市
532900 大理白族自治州
130800 承德市
530400 玉溪市
130900 沧州市
530500 保山市
371400 德州市
340400 淮南市
341500 六安市
620500 天水市
350600 漳州市
320400 常州市
370400 枣庄市
210600 丹东市
231100 黑河市
510900 遂宁市
652200 哈密地区
220100 长春市
520100 贵阳市
513200 阿坝藏族羌族自治州
431200 怀化市
460100 海口市
451000 百色市
530600 昭通市
460200 三亚市
220700 松原市
620400 白银市
320100 南京市
440600 佛山市
621100 定西市
320800 淮安市
430300 湘潭市
211100 盘锦市
620300 金昌市
140400 长治市
640500 中卫市
610700 汉中市
441400 梅州市
```

```
230200 齐齐哈尔市
330900 舟山市
511000 内江市
530300 曲靖市
350300 莆田市
410900 濮阳市
441700 阳江市
419000 省直辖县级行政区划
532300 楚雄彝族自治州
330800 衢州市
360200 景德镇市
659000 自治区直辖县级行政区划
532500 红河哈尼族彝族自治州
410300 洛阳市
429000 省直辖县级行政区划
130500 邢台市
130700 张家口市
411700 驻马店市
610500 渭南市
150900 乌兰察布市
310200 上海市
433100 湘西土家族苗族自治州
150500 通辽市
511400 眉山市
512000 资阳市
152500 锡林郭勒盟
653000 克孜勒苏柯尔克孜自治州
140300 阳泉市
441900 东莞市
140200 大同市
620700 张掖市
371200 莱芜市
522400 毕节地区
231000 牡丹江市
451100 贺州市
640100 银川市
421000 荆州市
411400 商丘市
320600 南通市
421200 咸宁市
510600 德阳市
230500 双鸭山市
140100 太原市
440100 广州市
110100 北京市
652300 昌吉回族自治州
341700 池州市
230700 伊春市
610100 西安市
620800 平凉市
542300 日喀则地区
522700 黔南布依族苗族自治州
620200 嘉峪关市
410500 安阳市
```

```
632300 黄南藏族自治州
341000 黄山市
320300 徐州市
230800 佳木斯市
360500 新余市
350500 泉州市
511700 达州市
```

运营商编码表

```
100017 中国电信
100025 中国移动
100026 中国联通
```

7.EdgeScript场景示例

本文为您介绍EdgeScript的定制化鉴权逻辑、定制化请求头和响应头控制、定制化改写和重定向、定制化缓存控制和定制化限速的场景示例。

定制化鉴权逻辑

自定义鉴权算法场景示例如下：

- 需求
  - 请求URL格式： `/path/digest/?.ts?key=&t=` 。
  - 针对 `.ts` 类请求，自定义防盗链需求如下：
    - 规则1：参数 `t` 或参数 `key` 不存在，响应403，增加响应头 `X-AUTH-MSG` 标识鉴权失败原因。
    - 规则2：参数 `t` 表示过期时间，若参数 `t` 小于当前时间，则响应403，增加响应头 `X-AUTH-MSG` 标识鉴权失败原因。
    - 规则3： `md5` 与 `digest` 匹配。若 `md5` 与 `digest` 不匹配，响应403。

      md5取值格式为： `私钥 + path + 文件名.后缀` 。
- 对应的EdgeScript规则

```
if eq(substr($uri, -3, -1), '.ts') {
   if or(not($arg_t), not($arg_key)) {
       add_rsp_header('X-AUTH-MSG', 'auth failed - missing necessary arg')
       exit(403)
   }
   t = tonumber($arg_t)
   if not(t) {
       add_rsp_header('X-AUTH-MSG', 'auth failed - invalid time')
       exit(403)
   }
   if gt(now(), t) {
       add_rsp_header('X-AUTH-MSG', 'auth failed - expired url')
       exit(403)
   }
    pcs = capture_re($request_uri,'^/([^/]+)/([^/]+)/([^?]+)\?(.*)')
    sec1 = get(pcs, 1)
    sec2 = get(pcs, 2)
    sec3 = get(pcs, 3)
    if or(not(sec1), not(sec2), not(sec3)) {
        add_rsp_header('X-AUTH-MSG', 'auth failed - malformed url')
        exit(403)
    }
    key = 'b98d643a-9170-4937-8524-6c33514bbc23'
    signstr = concat(key, sec1, sec3)
    digest = md5(signstr)
    if ne(digest, sec2) {
        add_rsp_header('X-AUTH-DEBUG', concat('signstr: ', signstr))
        add_rsp_header('X-AUTH-MSG', 'auth failed - invalid digest')
        exit(403)
    }
}
```

定制化请求头和响应头控制

文件自动重命名场景示例如下：

示例：

```
add_rsp_header('Content-Disposition', concat('attachment;filename=', tochar(34), filename,
tochar(34)))
```

> **说明**
>
> - 通过在HTTP应答中添加响应头“Content-Disposition：attachment”来实现消息体的强制下载，并且有参数 `filename` 时，自动重命名为 `filename` ，无参数时，使用默认命名。
> - `filename` 增加双引号，34为双引号的ascii，可经tochar转回字符串。

输出：

```
Content-Disposition: attachment;filename="monitor.apk"
```

对应的EdgeScript规则：

```
if $arg_filename {
    hn = 'Content-Disposition'
    hv = concat('attachment;filename=', $arg_filename)
    add_rsp_header(hn, hv)
}
```

定制化改写和重定向

定制化改写和重定向场景示例如下：

- 精确URI改写
  - 需求

    将用户请求 `/hello` 在CDN内部改写成 `/index.html` ，回源和缓存的URI都将变成 `/index.html` ，参数保持原样。
  - 对应的EdgeScript规则

    ```
    if match_re($uri, '^/hello$') {
        rewrite('/index.html', 'break')
    }
    ```
- 文件后缀改写
  - 需求

    将用户请求的URI信息 `/1.txt` ，在CDN节点上改写成 `/1.<url参数type>` ，即，使用请求URL中携带的type参数值来替换URI中的文件后缀，例如： `/1.txt?type=mp4` 将会被改成 `/1.mp4?type=mp4` 然后回源获取文件，并缓存在CDN节点上。

- 对应的EdgeScript规则

```
if and(match_re($uri, '^/1.txt$'), $arg_type) {
     rewrite(concat('/1.', $arg_type), 'break')
}
```

- 文件后缀小写化
  - 需求

    将URI改成小写。
  - 对应的EdgeScript规则

```
pcs = capture_re($uri, '^(.+%.)([^.]+)')
section = get(pcs, 1)
postfix = get(pcs, 2)
if and(section, postfix) {
    rewrite(concat(section, lower(postfix)), 'break')
}
```

- 添加URI前缀
  - 需求

    将用户请求 `^/nn_live/(.*)` ，在CDN节点上改写为 `/3rd/nn_live/$1` 。
  - 对应的EdgeScript规则

```
pcs = capture_re($uri, '^/nn_live/(.*)')
sec = get(pcs, 1)
if sec {
     dst = concat('/3rd/nn_live/', sec)
     rewrite(dst, 'break')
}
```

- 302重定向
  - 需求

    将根目录 `/` ，302重定向到 `/app/movie/pages/index/index.html` 页面。
  - 对应的EdgeScript规则

```
if eq($uri, '/') {
    rewrite('/app/movie/pages/index/index.html', 'redirect')
}
```

- 302重定向HTTPS
  - 需求

    将如下URI（对根目录匹配， `^/$` ）跳转到 `https://demo.aliyundoc.com/index.html` ，跳转后的URI可按需填写。
    - `http://demo.aliyundoc.com`
    - `https://demo.aliyundoc.com`

- 对应的EdgeScript规则

```
if eq($uri, '/') {
    rewrite('https://demo.aliyundoc.com/index.html', 'redirect')
}
```

定制化缓存控制

自定义缓存时长的场景样例如下：

- 需求

  根据各类条件，自定义资源缓存时长。

- 对应的EdgeScript规则

```
if match_re($uri, '^/image') {
    set_cache_ttl('code', '301=10,302=5')
}
if eq(substr($uri, -4, -1), '.mp4') {
    set_cache_ttl('path', 5)
}
if match_re($uri, '^/201801/mp4/') {
    set_cache_ttl('path', 50)
}
if match_re($uri, '^/201802/flv/') {
    set_cache_ttl('path', 10)
}
```

> **说明** /image开头的URI，针对响应码设置缓存时长，301缓存10s，302缓存5s。

定制化限速

自定义限速值的场景样例如下：

- 需求

  如果有参数 `sp` 和 `unit` ，则实施限速。 `sp` 参数指明限速数值， `unit` 为参数单位，KB或MB。

- 对应的EdgeScript规则

```
if and($arg_sp, $arg_unit) {
    sp = tonumber($arg_sp)
    if not(sp) {
        add_rsp_header('X-LIMIT-DEBUG', 'invalid sp')
        return false
    }
    if and(ne($arg_unit, 'k'), ne($arg_unit, 'm')) {
        add_rsp_header('X-LIMIT-DEBUG', 'invalid unit')
        return false
    }
    add_rsp_header('X-LIMIT-DEBUG', concat('set on: ', sp, $arg_unit))
    limit_rate(sp, $arg_unit)
    return true
}
```

EdgeScript配置示例操作视频

本视频以配置域名自定义限速策略为例，演示通过CDN控制台给域名配置EdgeScript边缘脚本的整个流程，包括添加边缘脚本、发布边缘脚本到模拟环境、测试模拟环境功能、发布边缘脚本到生产环境和测试生产环境功能。

8.EdgeScript实践分享

阿里云CDN技术专家在2020年05月08日在线为大家带来了《可编程CDN - EdgeScript实践》主题技术分享。通过本文你可以回顾该分享。

《可编程CDN - EdgeScript实践》主题技术分享介绍了阿里云CDN的成长和挑战，以及如何应对挑战、EdgeScript为何而生、EdgeScript语言速览和EdgeScript控制台实操，帮助您快速上手EdgeScript。

- EdgeScript应用场景、语言速览和实操演示。
- EdgeScript实践（在线视频介绍）。

9.示例：通过CLI工具使用边缘脚本

本文通过样例为您介绍通过CLI工具使用边缘脚本的操作方法，包括在CLI工具下进行规则的编写、保存、测试和发布的基本操作。

> **说明** CLI工具的使用方法，请参见通过CLI工具使用边缘脚本。

边缘脚本的基本操作如下：

- 将编写的边缘脚本规则存储为本地文件

  例如：m3u8.es规则拦截所有.m3u8请求

```
$cat m3u8.es
if eq(substr($uri, -5, -1), '.m3u8') {
    add_rsp_header('X-DEBUG-DENY-REASON', 'block m3u8')
    exit(400)
}
```

- 发布规则至模拟环境

```
$./es.py action=push_test_env domain=<your domain> rule='{"pos":"head","pri":"0","rule_pa
th":"./m3u8.es","enable":"on"}'
Response Code:
=============
200 OK
Response Info:
==============
{
    "RequestId": "FB98CC67-8FBA-44CF-A98A-BCE3B19FE510"
}
```

- 查看模拟环境的规则列表

```
$./es.py action=query_test_env domain=<your domain>
Response Code:
=============
200 OK
Response Info:
==============
{
    "DomainConfigs": [
        {
            "Status": "success",    # Status为success时，表示规则已成功设置
            "ConfigId": 17432558,
            "FunctionArgs": [
                {
                    "ArgName": "enable",
                    "ArgValue": "on"
                },
                {
                    "ArgName": "pri",
                    "ArgValue": "0"
                },
                {
                    "ArgName": "pos",
                    "ArgValue": "head"
                },
                {
                    "ArgName": "rule",
                    "ArgValue": "if eq(substr($uri, -5, -1), '.m3u8') {\n    add_rsp_head
er('X-DEBUG-DENY-REASON', 'block m3u8')\n    exit(400)\n}\n"
                }
            ],
            "FunctionName": "dsl_ex"
        }
    ],
    "RequestId": "4DDBF3DB-BCAC-4074-AC1E-B6C1F1C6CBFB"
}
```

- 测试规则

```
$curl -x 模拟环境IP:80 -o /dev/null -v 'http://www.archnote.net/test.m3u8'
< HTTP/1.1 400 Bad Request
< Server: Tengine
< Date: Thu, 18 Jul 2019 09:40:41 GMT
< Content-Type: text/html
< Content-Length: 265
< Connection: close
< X-DEBUG-DENY-REASON: block m3u8
< Via: cache1.cn1191-1[,0]
< Timing-Allow-Origin: *
< EagleId: 2a7b771b15634428415537484e
```

- 发布规则至生产环境

```
$./es.py action=push_product_env domain=<your domain>
Response Code:
=============
200 OK
Response Info:
==============
{
    "RequestId": "F4B378F8-6AAE-457A-A70C-E856ED8341D8"
}
```

- 查看生产环境的规则列表

```
$./es.py action=query_product_env domain=<your domain>
Response Code:
=============
200 OK
Response Info:
==============
{
    "DomainConfigs": {
        "DomainConfig": [
            {
                "Status": "success",
                "ConfigId": 17432558,
                "FunctionArgs": {
                    "FunctionArg": [
                        {
                            "ArgName": "enable",
                            "ArgValue": "on"
                        },
                        {
                            "ArgName": "pri",
                            "ArgValue": "0"
                        },
                        {
                            "ArgName": "pos",
                            "ArgValue": "head"
                        },
                        {
                            "ArgName": "rule",
                            "ArgValue": "if eq(substr($uri, -5, -1), '.m3u8') {\n    add_
rsp_header('X-DEBUG-DENY-REASON', 'block m3u8')\n    exit(400)\n}\n"
                        }
                    ]
                },
                "FunctionName": "dsl_ex"
            }
        ]
    },
    "RequestId": "36D57C1D-C820-43DA-8E70-DADC4B8BD4DD"
}
```