

腾讯云无服务器云函数

API 文档

产品文档



腾讯云

【版权声明】

©2013-2017 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

文档声明.....	2
API 文档.....	4
API 简介.....	4
API 概览.....	6
更新历史.....	7
调用方式.....	8
请求结构.....	8
请求结构简介.....	8
公共请求参数.....	10
接口请求参数.....	13
最终请求形式.....	16
返回结果.....	18
正确返回结果.....	18
错误返回结果.....	19
错误码.....	20
签名方法.....	24
接口调试工具.....	33
函数相关接口.....	34
获取函数列表.....	34
获取函数详细信息.....	38
创建函数.....	45
删除函数.....	50
更新函数.....	52
获取函数运行日志.....	57
获取函数监控数据.....	63
运行函数.....	69
触发器相关接口.....	75
设置函数触发器.....	75
删除函数触发器.....	83

API 文档

API 简介

欢迎使用腾讯云无服务器云函数(Serverless Cloud Function

)。无服务器云函数通过托管用户代码使用用户配置的触发器，根据触发器的触发事件自动执行用户应用，为用户提供 FaaS（Function as a Service）类型服务。

无服务器云函数（Serverless Cloud Function）是腾讯云提供的无服务器（Serverless）执行环境，您无需购买和管理服务器，而只需使用平台支持的语言编写核心代码并设置代码运行的条件，代码即可在腾讯云基础设施上弹性、安全地运行。腾讯云完全管理底层计算资源，包括服务器 CPU、内存、网络和其他配置/资源维护、代码部署、弹性伸缩、负载均衡等。代码按需运行，空闲时不收费。

术语表

术语	全称	中文名称	说明
SCF	Serverless Cloud Function	无服务器云函数	无服务器云函数通过托管用户代码，使用用户配置的触发器，根据触发器的触发事件，自动执行用户应用，为用户提供 FaaS（Function as a Service）类型服务。
Trigger	SCF Trigger	函数触发器	SCF 云函数由事件触发执行，事件来源即触发器，通常是由其他产品资源对象作为触发器，例如 COS Bucket，CMQ Topic 队列，定时器等

使用方式

通过 API 使用无服务器云函数，按如下步骤完成配置即可开始使用云函数：

1. 创建函数

您可以使用 [创建函数](#) 接口来创建一个云函数，其中函数名作为函数的唯一ID，在同一地域内不可冲突。

2. 创建函数触发器

创建完函数之后，您可以通过使用 [设置函数触发器](#) 接口来配置函数对某个触发器的某种事件做出响应和执行。

3. 测试函数

您可以通过使用 [运行函数](#)

接口来直接调用函数用于测试函数的执行正确与否，或通过触发器对象内操作以产生事件，并通过 [获取函数运行日志](#) 检验函数是否正确执行。

经过以上三个步骤，您的函数就可以提供服务了。

在本文档

的接口说明部分，

凡出现任何参数可选范围等方面与腾讯云官网上给出的数值发生矛盾时，均以官网上给出的值为准。

API 概览

函数相关接口

接口功能	Action ID	功能描述
获取函数列表	ListFunctions	用于查询获取当前地域下函数列表
获取函数详情	GetFunction	用于查询获取函数详细信息
创建函数	CreateFunction	用于创建函数
删除函数	DeleteFunction	用于删除指定函数
更新函数	UpdateFunction	用于更新指定函数配置信息，或更新指定函数代码
获取函数运行日志	GetFunctionLogs	用于查询获取指定函数的运行日志
获取函数监控数据	GetMonitorData	用于查询获取指定函数的监控信息
运行函数	InvokeFunction	用于运行指定函数

触发器相关接口

接口功能	Action ID	功能描述
设置函数触发器	SetTrigger	用于设置指定函数的触发器
删除函数触发器	DeleteTrigger	用于删除指定函数的指定触发器

更新历史

日期	更新内容
2017.09.25	ListFunctions 接口说明更新 GetFunction 接口说明更新 CreateFunction 接口说明更新 DeleteFunction 接口说明更新 UpdateFunction 接口说明更新 GetFunctionLogs 接口说明更新 GetMonitorData 接口说明更新 InvokeFunction 接口说明更新 SetTrigger 接口说明更新 DeleteTrigger 接口说明更新

调用方式

请求结构

请求结构简介

对腾讯云的 API 接口的调用是通过向腾讯云 API 的服务端地址发送请求，并按照接口说明在请求中加入相应的请求参数来完成的。腾讯云 API 的请求结构由：服务地址，通信协议，请求方法，请求参数和字符编码组成。具体描述如下：

服务地址

腾讯云 API 的服务接入地址与具体模块相关，详细请参见各接口相关描述。

通信协议

腾讯云 API 的大部分接口都通过 HTTPS 进行通信，为您提供高安全性的通信通道。

请求方法

腾讯云 API 同时支持 POST 和 GET 请求。

注意：

1. POST 和 GET 请求不能混合使用，若使用 GET 方式，则参数均从 Querystring 取得；若使用 POST 方式，则参数均从 Request Body 中取得，而 Querystring 中的参数将忽略。两种请求方式的参数格式规则相同，一般情况下使用 GET 请求，当参数字符串过长时推荐使用 POST。
2. 如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码，若为 POST，则无需对参数编码。
3. GET 请求的最大长度根据不同的浏览器和服务器设置有所不同，比如，传统 IE 浏览器限制为 2K，Firefox 限制为 8K；对于一些参数较多、长度较长的 API 请求，建议您使用 POST 方法以免在请求过程中会由于字符串超过最大长度而导致请求失败。
4. 对于 POST 请求，您需要使用

x-www-form-urlencoded

的形式传参，因为云 API 侧是从 \$_POST 中取出请求参数的。

请求参数

腾讯云 API 的每个请求都需要指定两类参数：公共请求参数以及接口请求参数。其中公共请求参数是每个接口都要用到的请求参数，具体可参见 [公共请求参数](#)，而接口请求参数是各个接口所特有的，具体见各个接口的“请求参数”描述。

字符编码

腾讯云 API 的请求及返回结果均使用 UTF-8 字符集进行编码。

公共请求参数

一个完整的腾讯云 API 请求需要两类请求参数：公共请求参数和接口请求参数。本文将介绍腾讯云 API 请求需要用到的 6 个公共请求参数，有关接口请求参数的详细说明请参见 [接口请求参数](#) 章节。

公共请求参数是每个接口都需要使用到的请求参数，开发者每次使用腾讯云 API 发送请求时都需要携带这些公共请求参数，否则会导致请求失败。公共请求参数的首字母均为大写，以此区分于接口请求参数。

公共请求参数具体列表如下：

注意：

文章中的接口实例以腾讯云 CVM 为例，各腾讯云产品的具体使用方式请对应实际产品。

参数名称	描述	类型	必选
Action	具体操作的指令接口名称，例如腾讯云 CVM 用户调用 查询实例列表 接口，则 Action 参数即为 DescribeInstances。	String	是
Region	地域参数，用来标识希望操作哪个地域的实例。详细信息可参见 地域和可用区 列表，或使用 查询地域列表 API 接口查看。 注意：1. 正常情况下此参数是必须的，如无需传入，则会在相应接口中进行说明。 2. 部分区域正在内测中，目前仅面向部分用户开放。	String	否
Timestamp	当前 UNIX 时间戳，可记录发起 API 请求的时间。	UInt	是
Nonce	随机正整数，与	UInt	是

参数名称	描述	类型	必选
	Timestamp 联合起来，用于防止重放攻击。		
SecretId	在 云API密钥 上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey，而 SecretKey 会用来生成请求签名 Signature。具体可参考 签名方法 章节。	String	是
Signature	请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。计算方法可参考 签名方法 章节。	String	是
SignatureMethod	签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。详细签名计算方法可参考 签名方法 章节。	String	否
Token	临时证书所用的 Token，需要结合临时密钥一起使用。长期密钥不需要 Token。	String	否

使用示例

腾讯各云产品 API 请求链接中，公共请求参数的形式如下，以腾讯云 CVM 为例，假设用户想要查询广州地域的云服务器实例列表，则其请求链接的形式为：

<https://cvm.api.qcloud.com/v2/index.php?>

Action=DescribeInstances

&SecretId=xxxxxxx

&Region=ap-guangzhou

&Timestamp=1465055529

&Nonce=59485

&Signature=mysignature

&SignatureMethod=HmacSHA256

&

接口请求参数

一个完整的腾讯云 API 请求需要两类请求参数：公共请求参数和接口请求参数。本文将介绍腾讯云 API 请求需要用到的接口请求参数，有关公共请求参数的详细说明请参见 [公共请求参数](#) 章节。

接口请求参数与具体的接口有关，不同的接口支持的接口请求参数也不一样。接口请求参数的首字母均为小写，以此区分于公共请求参数。

注意：

文章中的参数以腾讯云 CVM 为例，各腾讯云产品的实际参数请对应实际产品 API 参数说明。

以下参数列表以腾讯云 CVM [查询实例列表](#) 接口（DescribeInstances）为例，该 API 支持的接口请求参数如下：

参数名称	描述	类型	必选
instanceIds.n	要查询的 CVM 实例 ID 数组，数组下标从 0 开始。可以使用 instanceId 和 unInstanceId，建议使用统一资源 ID:unInstanceId。	String	否
lanIps.n	要查询的云服务器器的内网 IP 数组。	String	否
searchWord	用户设定的主机别名。	String	否
offset	偏移量，默认为 0。	Int	否
limit	一次最多可查询的服务器数量，默认为 20，最大为 100。	Int	否
status	待查询的主机状态。	Int	否
projectId	项目 ID，不传则查询全部项目的 CVM 实例。0 表示默认项目，如需指定其他项目，可	String	否

参数名称	描述	类型	必选
	调用 查询项目列表 (DescribeProject) 接口查询。		
simplify	获取非实时数据，当传参添加 simplify=1 时获取非实时数据。	Int	否
zoneId	可用区 ID，不传则查询所有可用区的 CVM 实例。如需指定可用区，可调用 查询可用区 (DescribeAvailabilityZones) 接口查询。	Int	否

其中各字段的说明如下：

参数名称：该接口支持的请求参数名，用户可以在使用此接口时将其作为接口请求参数。如果参数名称以

“.n”

结尾，则表明此参数为一个数组，使用时需要依次传入数组参数。

必选：

标志此参数是否是必须的，若为“是”，则表明调用该接口必须传入此参数；若为“否”，表示可以不传入。

类型：此接口参数的数据类型。

描述：简要描述了此接口请求参数的内容。

使用示例

腾讯各云产品 API 请求链接中，接口请求参数的形式如下，以腾讯云 CVM

为例，假设用户想要查询伸缩组列表，则其请求链接的形式为：

https://cvm.api.qcloud.com/v2/index.php?

&

[&instanceIds.0=ins-0hm4gvho](#)

[&instanceIds.1=ins-8oby8q00](#)

[&offset=0](#)

[&limit=20](#)

[&status=2](#)

[&zoneId=100003](#)

最终请求形式

拼接规则

腾讯云 API 接口请求 URL 的拼接规则为：

`https:// + 请求域名 + 请求路径 + ? + 最终请求参数串`

组成部分说明：

- 请求域名：

请求域名由接口所属的产品或所属产品的模块决定，不同产品或不同产品的模块的请求域名会有不同，如腾讯云 CVM 的查询实例列表（DescribeInstances）的请求域名为：

`cvm.api.qcloud.com`

。具体产品请求域名详见各接口说明。

- 请求路径：腾讯云 API 对应产品的请求路径，一般是一个产品对应一个固定路径（如腾讯云 CVM 请求路径固定为

`/v2/index.php`

）。

- 最终请求参数串：接口的请求参数串包括公共请求参数和接口请求参数。

使用示例

腾讯云 API 接口最终的请求 URL 结构形式如下：

以腾讯云 CVM [查询实例列表](#) 接口（DescribeInstances）为例，其中前 6 个参数为公共请求参数，后 6 个参数为接口请求参数。

`https://cvm.api.qcloud.com/v2/index.php?`

Action=DescribeInstances

&SecretId=xxxxxxx

&Region=gz

&Timestamp=1465055529

&Nonce=59485

&Signature=mysignature //公共请求参数

&instanceIds.0=ins-0hm4gvho

&instanceIds.1=ins-8oby8q00

&offset=0

&limit=20

&status=2

&zoneId=100003 //接口请求参数

返回结果

正确返回结果

若 API 调用成功，则最终返回结果中的错误码 code 为 0，错误信息 message 为空，并且会显示返回的结果数据。

示例如下：

```
{  
  "code": 0,  
  "message": "",  
  <返回结果数据>  
}
```

错误返回结果

若 API 调用失败，则最终返回结果中的错误码 code 不为 0，显示错误代码，message 字段会显示详细错误信息。用户可以根据 code 和 message 在 [错误码](#) 页面查询具体的错误信息。

错误返回示例如下：

```
{
  "code": 5100,
  "message": "(100004)projectId不正确",
}
```

错误码

错误码

message字段表示与模块相关的错误。

其示例如下：

"message": "(100004)projectId不正确"

它由两部分组成，()内的为模块错误代码，()后的为具体的错误描述。

不同模块可能产生的

错误情况是不一样的，用户可以根据具体

的错误描述确定错误所在。

下面的表都是公共错误码，如果您要查找的错误码不在其中，请到接口文档中具体接口说明处查找。

公共错误码

错误代码	模块错误代码	英文提示	描述
4000	10000	invalid request parameters	无效（非法）请求参数。通常是参数没有按照CMQ云API的接口说明导致的错误。
4000	10010	lacked of required parameters	缺少必填参数。应参照CMQ云API接口的说明看缺少哪些必填参数。
4100	10030	authentication failed	鉴权失败。可能的原因有 1) 生成签名算法的secret key错误； 2) 签名算法 有误（没用官方sdk，自己实现签名算法）。
4300	10040	charge overdue	账户欠费。
6000	10050	server internal error	服务器内部错误。一般会有requestId返回给用户，用户可根据requestId提工单给技术支持人员定位问题

			。
4100	10080	secret id status error	secret id状态错误。secret id可能处于禁用状态。
4000	10110	request parameters error	请求参数错误。通常是参数没有按照CMQ云API的接口说明导致的错误。
4420	10250	qps throttling	qps达到最大值，因此被限制。为了保证后台的稳定性和调用时延，目前实际的限制是每秒的消息数最大5k。对于有特殊要求的客户，可以提工单申请。
4100	10270	secret id is not existed	secret id 不存在。
4000	10280	action is not existed	参数Action不存在。
4000	10310	error: parameter key format error"	参数格式错误。
4000	10320	no such parameter	不存在这样的参数。
4000	10330	parameter is NOT a repeatable parameter	参数不是一个可以重复的参数。在批量接口中，例如BatchSendMessage，msgBody是可重复的参数。在BatchDeleteMessage中，receiptHandle是可重复参数。把每个接口规定的可重复参数之外的参数以可重复参数的形式传入都视为错误。
4000	10350	parameter value or length is out of range	参数取值或参数长度超过了范围限制。通常对于整型而言，是取值错误。对String类型参数而言，是长度错误。
4000	10360	parameter error type	参数类型错误。例如把整

			型填成了字符串类型。
4000	10370	parameter batch size is more than 16	批量接口的批量值超过了16。
4000	10380	parameter is not consequent	可重复参数的下标不连续。
4000	10390	lacked of required parameter	缺少必填的参数。
4000	10400	cannot find parameter in uri	uri中找不到参数。
4000	10410	unexpected http method, only GET or POST is supported	不支持的http方法。目前只支持GET和POST方法。由于GET有长度的限制，建议使用POST。
4000	10420	cannot parse	无法解析http报文。
4000	10430	action name is not existed	Action所指定的接口名不存在。
4000	10440	account illegal, it may be an assistant account	帐号不合法。请检查是不是协助者帐号，目前CMQ不支持协助者帐号的操作。
4000	10450	secret id dosen't begin with AKID account	secret id不是以AKID开头。
4480	10460	exceed interface frequency limit, please slow down	为保护后端系统，控制类接口（例如 CreateQueue, DeleteQueue等）的允许调用频率比 QPS 要低很多，所以看到该错误时，请减少控制类接口调用频率。
4200	10461	no cam authorization	子账号没有被根账号授权访问该接口或者资源。

SCF 错误码

错误代码	模块错误代码	英文提示	描述
4400	9003	InvalidParameter	参数错误

5100	9000	SystemError	系统错误
4102	9002	SecretidNotAuthAccess Resource	未授权访问资源
4000	9305	InsufficientBalance	余额不足

签名方法

腾讯云 API 会对每个访问的请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature），以验证用户身份。签名信息由用户所执有的安全凭证生成，安全凭证包括 SecretId 和 SecretKey，若用户还没有安全凭证，则需要在腾讯云官网上自主申请，否则无法调用云 API 接口。

申请安全凭证

在第一次使用腾讯云 API 之前，用户需要在【腾讯云控制台】>【[API 密钥管理](#)】上申请安全凭证。安全凭证包括 SecretId 和 SecretKey，其中：

- SecretId：用于标识 API 调用者身份；
- SecretKey：用于加密签名字符串和服务器端验证签名字符串的密钥。

注意：

API 密钥是构建腾讯云 API 请求的重要凭证，使用腾讯云 API 可以操作您名下的所有腾讯云资源，为了您的财产和服务安全，请妥善保管并定期更换密钥，当您更换密钥后，请及时删除旧密钥。

申请安全凭证步骤：

1. 登录 [腾讯云控制台](#)。
2. 单击【云产品】，选择【管理工具】栏下的【云 API 密钥】，进入云 API 密钥管理页面。



3. 在 [API 密钥管理](#) 页面，单击【新建密钥】即可以创建一对 SecretId/SecretKey。

注意：

- 开发商帐号最多可以拥有两对 SecretId / SecretKey。
- 被开发商添加为子用户的 QQ 帐号，在不同开发商控制台，可以申请不同的安全凭证。
- 子用户的安全凭证，目前仅可调用部分接口的云 API。

生成签名串

有了安全凭证 SecretId 和 SecretKey 后，就可以生成签名串了。生成签名串的详细过程如下：



假设用户的 SecretId 和 SecretKey 分别是：

SecretId：AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA

SecretKey：Gu5t9xGARNpq86cd98joQYCN3Cozk1qA

注意：

这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 和请求参数进行后续操作。

以腾讯云 CVM 为例，当用户调用腾讯云 CVM 的 [查看实例列表](#) (DescribeInstances)接口时，其请求参数为：

参数名称	描述	参数值
Action	方法名	DescribeInstances
SecretId	密钥 ID	AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA
Timestamp	当前时间戳	1465185768
Nonce	随机正整数	11886
Region	实例所在区域	ap-guangzhou
SignatureMethod	签名方式	HmacSHA256
InstanceIds.0	待查询的实例 ID	ins-09dx96dg

1. 对参数排序

首先对所有请求参数按参数名做字典序升序排列。（所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推。）您可以借助编程语言中的相关排序函数来实现这一功能，如 php 中的 ksort 函数。上述示例参数的排序结果如下：

```
{
  "Action": "DescribeInstances",
  "Nonce": 11886,
  "Region": "ap-guangzhou",
  "SecretId": "AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA",
  "SignatureMethod": "HmacSHA256",
  "Timestamp": 1465185768,
  "InstanceIds.0": "ins-09dx96dg"
}
```

使用其它程序设计语言开发时, 可对上面示例中的参数进行排序, 得到的结果一致即可。

2. 拼接请求字符串

此步骤将生成请求字符串。

将把上一步排序好的请求参数格式化

“参数名称” = “参数值”

的形式, 如对 Action 参数, 其参数名称为

"Action"

, 参数值为

"DescribeInstances"

, 因此格式化后就为

Action=DescribeInstances

。

注意：

1. “参数值” 为原始值而非 url 编码后的值。
2. 若输入参数的 Key 中包含下划线, 则需要将其转换为 “.”, 但是 Value 中的下划线则不用转换。如

Placement_Zone=CN_GUANGZHOU

, 则需要将其转换成

Placement.Zone=CN_GUANGZHOU

。

然后将格式化后的各个参数用

"&"

拼接在一起，最终生成的请求字符串为（请忽略文中的换行）：

```
Action=DescribeInstances
&InstanceIds.0=ins-09dx96dg
&Nonce=11886
&Region=ap-guangzhou
&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA
&SignatureMethod=HmacSHA256
&Timestamp=1465185768
```

3. 拼接签名原文字符串

此步骤将生成签名原文字符串。

签名原文字符串的拼接规则为：

请求方法 + 请求主机 + 请求路径 + ? + 请求字符串

参数构成说明：

- 请求方法：支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
- 请求主机：
即主机域名，请求域名由接口所属的产品或所属产品的模块决定，不同产品或不同产品的模块的请求域名会有不同。如腾讯云 CVM 的查询实例列表（DescribeInstances）的请求域名为：

cvm.api.qcloud.com

，具体产品请求域名详见各接口说明。

- 请求路径：腾讯云 API 对应产品的请求路径，一般是一个产品对应一个固定路径，如腾讯云 CVM 请求路径固定为

/v2/index.php

- 请求字符串：即上一步生成的请求字符串。

因此，上述示例的拼接签名原字符串结果为（请忽略文中的换行）：

```
GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances
&InstanceIds.0=ins-09dx96dg
&Nonce=11886
&Region=ap-guangzhou
&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA
&SignatureMethod=HmacSHA256
&Timestamp=1465185768
```

4. 生成签名串

此步骤生成签名串。

注意：

计算签名的方法有两种：HmacSHA256 和 HmacSHA1 这里要根据您指定的签名算法（即 SignatureMethod 参数）生成签名串。当指定 SignatureMethod 为 HmacSHA256 时，需要使用

HmacSHA256 计算签名，其他情况请使用 HmacSHA1 计算签名。

首先使用签名算法（HmacSHA256 或 HmacSHA1）对上一步中获得的 签名原字符串进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例，由于本例中所用的签名算法为 HmacSHA256，因此生成签名串的代码如下（使用其它程序设计语言开发时，可用上述示例中的原字符串进行签名验证，得到的签名串与例子中的一致即可）：

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3Cozk1qA';  
$srcStr =  
'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Non  
ce=11886&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA&Signatu  
reMethod=HmacSHA256&Timestamp=1465185768';  
$signStr = base64_encode(hash_hmac('sha256', $srcStr, $secretKey, true));  
echo $signStr;
```

最终得到的签名串为:

0EEem/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s=

同理，当您指定签名算法为 HmacSHA1 时，生成签名串的代码如下：

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3Cozk1qA';  
$srcStr =  
'GETcvm.api.qcloud.com/v2/index.php?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Non  
ce=11886&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA&Signatu  
reMethod=HmacSHA1&Timestamp=1465185768';  
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
```

```
echo $signStr;
```

最终得到的签名串为：

```
nPVnY6njQmwQ8ciqbPl5Qe+Oru4=
```

签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为

```
0EEem/HtGRr/VJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s=
```

，则其编码后为

```
0EEem%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D
```

。因此，最终得到的签名串请求参数 (Signature) 为：

```
0EEem%2FHtGRr%2FVJXTAD9tYMth1Bzm3lLHz5RCDv1GdM8s%3D
```

，它将用于生成最终的请求URL。

注意：

如果用户的请求方法是 GET，则对所有请求参数的参数值均需要做 URL

编码；此外，部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

鉴权失败

当鉴权不通过时，可能出现如下表的错误：

错误代码	错误类型	错误描述
4100	身份认证失败	身份验证失败，请确保您请求参数中的 Signature 按照上述步骤计算正确，特别注意 Signature 要做 url 编码后再发起请求。
4101	未被开发商授权访问本接口	该子用户未被授权调用此接口。请联系开发商授权，详情请查阅 授权策略 。
4102	未被开发商授权访问本接口中所操作的资源	

接口调试工具

请前往 [云API接口调试工具](#) 进行接口测试。

函数相关接口

获取函数列表

接口描述

功能描述

ListFunctions 根据传入的查询参数返回相关函数信息。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

语法示例：

GET https://scf.api.qcloud.com/v2/index.php?Action=ListFunctions

&<公共请求参数>

&limit=2

&order=asc

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 ListFunctions。

参数名称	描述	类型	必选
order	以升序还是降序的方式返回	String	否

参数名称	描述	类型	必选
	结果，可选值 asc 和 desc		
orderby	根据哪个字段进行返回结果排序,支持以下字段： addtime、modtime、function name	String	否
offset	数据偏移量，默认值为 0	Int	否
limit	返回数据长度，默认值为 20	Int	否
searchKey	模糊匹配 functionName 字段	String	否

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": {
    "functions": [
      {
        "modTime": "2017-09-08 17:42:53",
        "addTime": "2017-09-08 17:42:53",
        "runtime": "Python2.7",
        "functionName": "image"
      },
      {
        "modTime": "2017-09-18 14:06:57",
        "addTime": "2017-08-18 20:44:36",
        "runtime": "Nodejs6.10",
        "functionName": "testnode"
      }
    ]
  }
}
```

```

],
"total": 2
}
}

```

响应参数

参数名称	描述	类型
code	公共错误码，0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String
data	json 格式数据，返回属于该用户的 函数列表信息	Object

data 字段包含属于该用户的函数列表信息，其中的 total 字段表示该用户在该区域的函数总数，functions 为函数列表，每个函数包含的数据结构如下：

参数名称	描述	类型
modTime	函数最后一次修改时间	String
addTime	函数创建时间	String
functionName	函数名称	String
runtime	函数所使用的运行环境	String

实际案例

请求

```

GET https://scf.api.qcloud.com/v2/index.php?
Action=ListFunctions
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu
&Nonce=24693

```

&Timestamp=1505723955
&Region=gz
&Signature=gXlWZFfqRWZaaCPyX%2F4U0E92yPk%3D
&limit=2
&order=asc

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": {
    "functions": [
      {
        "modTime": "2017-06-08 17:36:38",
        "addTime": "2017-06-08 17:36:38",
        "runtime": "Python2.7",
        "functionName": "ziptest"
      },
      {
        "modTime": "2017-06-10 16:23:12",
        "addTime": "2017-06-10 16:20:00",
        "runtime": "Python2.7",
        "functionName": "zoepil"
      }
    ],
    "total": 7
  }
}
```

获取函数详细信息

功能描述

GetFunction 获取某个函数的详细信息，包括名称、代码、处理方法、关联触发器和超时时间等字段。

接口访问域名：scf.api.qcloud.com

请求

语法示例：

GET https://scf.api.qcloud.com/v2/index.php?Action=GetFunction

&<公共请求参数>

&functionName=<FunctionName>

请求参数

以下请求参数列表仅列出了

接口请求参数，正式调用时需要加上公共请求参数，见[公共请求参数](#)页面。其中，此接口的Action字段为GetFunction。

参数名称	必选	类型	描述
functionName	是	String	需要获取详情的函数名称
code	否	Int	0表示获取函数详情时不返回代码字段，1表示返回代码字段，默认值为0

响应

响应示例：

```
{
```

```

"code": 0,
"message": "",
"codeDesc": "Success",
"data": {
  "modTime": "2017-06-08 17:36:38",
  "functionName": "ziptest",
  "triggers": [
    {
      "modTime": "2017-09-18 16:52:33",
      "type": "cos",
      "triggerDesc": "{\"event\":\"cos:ObjectCreated:*\"}",
      "triggerName": "testscfcode-1251762227.cn-south.myqcloud.com",
      "addTime": "2017-09-18 16:52:33"
    }
  ],
  "handler": "index.main_handler",
  "codeSize": 2600020,
  "memorySize": 128,
  "version": "LATEST",
  "timeout": 3,
  "runtime": "Python2.7",
  "description": ""
}

```

响应参数

参数名称	类型	描述
code	Int	公共错误码, 0表示成功, 其他值表示失败
message	String	模块错误信息描述, 与接口相关
codeDesc	String	英文错误码, 成功返回 Success, 失败有相应的英文说明

参数名称	类型	描述
data	Object	json数据，包含该函数的触发器名称、运行时memorySize、入口函数、函数描述、版本号、代码大小和代码等信息

其中data字段除了包含函数创建时的描述、处理程序名称、运行时内存大小等字段外，还包括关联该函数的触发器列表信息。

函数的数据结构如下：

参数名称	类型	描述
modtime	String	函数的最后一次修改时间
addtime	String	函数的创建时间
functionName	String	函数名称
handler	String	函数入口
codeSize	Long	函数代码大小
memorySize	Int	配置的内存使用大小，以MB为单位
version	String	函数版本，当前均为"LATEST"
timeout	String	函数运行超时时间，单位为秒（s）
runtime	String	函数运行环境
description	String	函数描述字段
triggers	Object	函数触发器列表

每个触发器包含的数据结构如下：

参数名称	类型	描述
modtime	String	触发器最后一次修改时间
addtime	String	触发器和函数的绑定时间
type	String	触发器类型
triggerDesc	String	触发器的相关参数
triggerName	String	触发器名称

实际案例

不返回代码示例

请求

GET https://scf.api.qcloud.com/v2/index.php?

Action=GetFunction

&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu

&Nonce=32501

&Timestamp=1505724757

&Region=gz

&Signature=eY%2Fmn5Cw%2Fjxnhb%2FQCneeQBJDXkI%3D

&functionName=ziptest

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": {
    "modTime": "2017-06-08 17:36:38",
    "functionName": "ziptest",
    "triggers": [
      {
        "modTime": "2017-09-18 16:52:33",
        "type": "cos",
        "triggerDesc": "{\"event\":\"cos:ObjectCreated:*\"}",
        "triggerName": "testscfcode-1251762227.cn-south.myqcloud.com",
        "addTime": "2017-09-18 16:52:33"
      }
    ],
    "handler": "index.main_handler",
    "codeSize": 2600020,
```

```
"memorySize": 128,  
"version": "LATEST",  
"timeout": 3,  
"runtime": "Python2.7",  
"description": ""  
}  
}
```

返回代码示例

请求

```
GET https://scf.api.qcloud.com/v2/index.php?  
Action=GetFunction  
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu  
&Nonce=52947  
&Timestamp=1505725241  
&Region=gz  
&Signature=Mp7r7xiWyyFNyrK5UMrevN9aKC0%3D  
&functionName=image  
&code=1
```

响应

```
{  
  "code": 0,  
  "message": "",  
  "codeDesc": "Success",  
  "data": {  
    "modTime": "2017-09-18 17:00:31",  
    "codeError": "",  
    "code": "# coding=utf-8\nimport uuid\nimport json\nimport os\nimport logging\nfrom PIL import
```

```

Image\nimport PIL.Image\nimport commands\nimport datetime\nfrom qcloud_cos import
CosClient\nfrom qcloud_cos import DownloadFileRequest\nfrom qcloud_cos import
UploadFileRequest\n\nprint('Loading function')\nappid = 1251762227 #使用帐号appid\nsecret_id =
u'AKIDutrojKI3CKQZNAr763UXks05898Lmciu' #使用云api secret id\nsecret_key =
u'96VJxxxxxxxxxxxxxxxxxxxx2' #使用云api secret key\nregion = u'gz' \n\ncos_client = CosClient(appid,
secret_id, secret_key, region)\nlogger = logging.getLogger()\n\nupload_bucket=u"imagedst"
\n\n\ndef resize_image(image_path, resized_path):\n with Image.open(image_path) as image:\n
image.thumbnail(tuple(x / 2 for x in image.size))\n image.save(resized_path)\n\n\ndef
delete_local_file(src):\n logger.info("\delete files and folders")\n if os.path.isfile(src):\n try: \n
os.remove(src) \n except: \n pass \n elif os.path.isdir(src): \n for item in os.listdir(src): \n
itemsrc=os.path.join(src,item) \n delete_file_folder(itemsrc) \n try: \n os.rmdir(src) \n except: \n pass
\n\n\ndef main_handler(event, context):\n logger.info("\start main handler")\n for record in
event['Records']:\n try:\n bucket = record['cos']['cosBucket']['name']\n key =
record['cos']['cosObject']['key'] \n download_path = '/tmp/{}'.format(uuid.uuid4(), key.split('/')[1])\n
upload_path = '/tmp/resized-{}'.format(key.split('/')[1])\n print("\Get from [%s] to download file
[%s]" %(bucket,key))\n\n # download image from cos\n request = DownloadFileRequest(bucket,
"\^"+key.split('/')[1], download_path)\n download_file_ret = cos_client.download_file(request)\n if
download_file_ret['code'] == 0:\n logger.info("\Download file [%s] Success" % key)\n
logger.info("\Image compress function start")\n starttime = datetime.datetime.now()\n\n #compress
image here\n resize_image(download_path, upload_path)\n endtime = datetime.datetime.now()\n
logger.info("\compress image take " + str((endtime-starttime).microseconds/1000) + "ms")\n\n
#upload the compressed image to resized bucket\n request = UploadFileRequest(upload_bucket,
u'/' + key.split('/')[1].decode('utf-8'), upload_path.decode('utf-8'))\n upload_file_ret =
cos_client.upload_file(request)\n logger.info("\upload image, return message: " +
str(upload_file_ret))\n\n #delete local file\n delete_local_file(str(download_path))\n
delete_local_file(str(upload_path))\n else:\n logger.error("\Download file [%s] Failed, err: %s" % (key,
download_file_ret['message']))\n return -1\n except Exception as e:\n print(e)\n print("Error getting
object {} from bucket {}. Make sure the object exists and your bucket is in the same region as this
function.'.format(key, bucket))\n raise e",
"description": "",
"triggers": [
{
"modTime": "2017-09-08 17:42:55",
"type": "cos",

```

```
"triggerDesc": "{\"event\":\"cos:ObjectCreated:*\"}",
"triggerName": "imagesrc-1251762227.cn-south.myqcloud.com",
"addTime": "2017-09-08 17:42:55"
}
],
"handler": "imageresize.main_handler",
"errNo": 0,
"codeSize": 605400,
"timeout": 3,
"version": "LATEST",
"codeResult": "success",
"memorySize": 128,
"runtime": "Python2.7",
"functionName": "image"
}
}
```

创建函数

接口描述

功能描述

CreateFunction 用于创建函数，调用该接口时必须指定函数名称、代码和处理方法，其他的参数如：函数描述、运行时内存大小、运行超时时间等都为可选参数。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

语法示例：

GET https://scf.api.qcloud.com/v2/index.php/?Action=CreateFunction

&<公共请求参数>

&functionName=<FunctionName>

&code=<代码 zip 文件 base64 编码后的字符串>

&handler=<function.handler>

&description=<Description String>

&memorySize=<128>

&timeout=<2>

&runtime=<Python2.7>

&codeObject.cosBucketName=<cosBucketName>

&codeObject.cosObjectName=<cosObjectName>

请求参数

以下请求参数为本接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 CreateFunction。

参数名称	描述	类型	必选
functionName	创建的函数名称，函数名称支持26个英文字母大小写、数字、连接符和下划线，第一个字符只能以字母开头，最后一个字符不能为连接符或者下划线，名称长度2-60	String	是
code	包含函数及其依赖项的 zip 格式文件，使用该接口时要求将 zip 文件的内容转成 base64 编码，并在 base64 编码后的字符前加上 @ 字符后使用URL编码处理。zip 文件最大支持 10M 注意：该字段不参与鉴权。	String	是
handler	函数处理方法名称，名称格式支持 "文件名称.方法名称" 形式，文件名称和函数名称之间以 "." 隔开，文件名称和函数名称要求以字母开始和结尾，中间允许插入字母、数字	String	是

参数名称	描述	类型	必选
	、下划线和连接符，文件名称和函数名字的长度要求是 2-60 个字符		
description	函数描述,最大支持 1000 个英文字母、数字、空格、逗号、换行符和英文句号，不支持中文	String	否
memorySize	函数运行时内存大小，默认为 128M，可选范围 128MB-1536MB，并且以 128MB 为阶梯	Int	否
timeout	函数最长执行时间，单位为秒，可选值范围 1-300 秒，默认为 3 秒	Int	否
runtime	函数运行环境，目前仅支持 Python2.7，Python3.6，Nodejs6.10 和 Java8	Int	否
codeObject	scf 支持从 cos 导入代码，该参数指定 cos 路径，包括 cosBucketName 和 cosObjectName， 注意： 该字段不能与code同时指定。	Object	否

注意：一个帐号在一个地域有函数配额限制，同时每个函数支持的触发器同样有配额限制。

响应

响应示例：

```
{
```

```
"code": 0,  
"message": "",  
"codeDesc": "Success"  
}
```

响应参数

参数名称	描述	类型
code	公共错误码, 0 表示成功, 其他值表示失败。详见错误码页面的 公共错误码	Int
message	模块错误信息描述, 与接口相关	String
codeDesc	英文错误码, 成功返回 Success, 失败有相应的英文说明	String

实际案例

请求

GET https://scf.api.qcloud.com/v2/index.php?

Action=CreateFunction

&Nonce=610072102

&Region=gz

&Timestamp=1505790265

&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu

&functionName=test7

&description=helloWorldfunction

&memorySize=128

&timeout=3

&handler=index.main_handler

&code=%40UEsDBBQAAAAIAEWpMkuPqFdcYgAAAHAAAAIAAAAAW5kZXgucHklicOQDAQQPcS
d5iw8Al7cQI3sBTRqTZppzKm4viKt3zvldC3PWxBWdpHiKKHV%2BSZQg1%2BtbSYIZRDrvFCki6dJHhLM
%2BYZJA62JHU1oXMB5sBOVc1fGCUyQWG%2BpDI4ODddPFBLAQI%2FABQAAAAIAEWpMkuPqFdcY

gAAAHAAAAIACQAAAAAAAAIAAAAAAABpbmRleC5weQoAIAAAAAAAQAYAHFx%2BHN%2F
MNMBIZcEb38w0wEhlwRvfzDTAVBLBQYAAAAAAQABAFoAAACIAAAAAAA%253d
&Signature=XuQknRcWlZvpBItk7G6hmoOuPfE%3D

响应

```
{  
  "code": 0,  
  "message": "",  
  "codeDesc": "Success"  
}
```

删除函数

接口描述

功能描述

DeleteFunction 删除函数并同步删除和该函数绑定的触发器。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 DeleteFunction。

参数名称	描述	类型	必选
functionName	要删除的函数名称	String	是

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

响应参数

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String

实际案例

请求

```
GET https://scf.api.qcloud.com/v2/index.php?Action=DeleteFunction
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu
&Nonce=51911
&Timestamp=1505726381
&Signature=xX2Z85V5WUesWDbkKQi9TOAFeQk%3D
&Region=gz
&functionName=test2
```

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

更新函数

接口描述

功能描述

UpdateFunction 更新函数的相关字段包括代码、处理方法、描述、运行时内存大小和运行超时时间等。

接口域名

无服务器云函数接口请求域名为：

scf.api.qcloud.com

请求

语法示例：

GET https://scf.api.qcloud.com/v2/index.php?

Action=UpdateFunction

&<公共请求参数>

&functionName=<FunctionName>

&code=<代码 zip 文件 base64 编码后的字符串>

&handler=<function.handler>

&description=<Description String>

&memorySize=<256>

&timeout=<30>

&runtime=<Python2.7>

&codeType=<Zipfile>

&codeObject.cosBucketName=<cosBucketName>

&codeObject.cosObjectName=<cosObjectName>

请求参数

以下请求参数为本接口的接口请求参数，正式调用时需要加上公共请求参数，见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 UpdateFunction。

参数名称	描述	类型	必选
functionName	要修改的函数名称	String	是
code	包含函数代码文件及其依赖项的 zip 格式文件，使用该接口时要求将 zip 文件的内容转成 base64 编码并在 base64 编码后的字符前加上 @ 字符后使用URL编码处理。zip 文件最大支持 10M 注意：该字段不参与鉴权。	String	否
handler	函数处理方法名称。名称格式支持“文件名称.函数名称”形式，文件名称和函数名称之间以"."隔开，文件名称和函数名称要求以字母开始和结尾，中间允许插入字母、数字、下划线和连接符，文件名称和函数名字的长度要求 2-60 个字符	String	否
description	函数描述。最大支持 1000 个英文字母、数字、空格、	String	否

参数名称	描述	类型	必选
	逗号和英文句号，不支持中文		
memorySize	函数运行时内存大小，默认为 128 M，可选范 128 M-1536 M	Int	否
timeout	函数最长执行时间，单位为秒，可选值范 1-300 秒，默认为 3 秒	Int	否
runtime	函数运行环境，目前仅支持 Python2.7，Python3.6，Nodejs6.10 和 Java8	Int	否
codeType	代码更新类型，当前支持三种：Inline表示在线更新，Zipfile表示通过代码包更新，Cos表示通过从 cos 导入代码更新	String	是
codeObject	scf 支持从 cos 导入代码，该参数指定 cos 路径，包括 cosBucketName 和 cosObjectName，注意：该字段不能与 code 同时指定。	Object	否

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

响应参数

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String

实际案例

请求

```
GET https://scf.api.qcloud.com/v2/index.php?
Action=UpdateFunction
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu
&Nonce=44144
&Timestamp=1505803748
&Region=gz
&Signature=iAPNvkpfW95lc96bhMtxTIN8pXY%3D
&functionName=test5
&memorySize=1024
&timeout=300
```

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```


获取函数运行日志

接口描述

功能描述

GetFunctionLogs 根据设置的日志查询条件返回函数日志。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

语法示例：

GET https://scf.api.qcloud.com/v2/index.php?Action=GetFunctionLogs

&<公共请求参数>

&offset= <OffsetValue>

&limit= <Limit>

&orderby= <sort type>

&functionName= <FunctionName>

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 GetFunctionLogs。

参数名称	描述	类型	必选
functionName	函数名称	String	否

参数名称	描述	类型	必选
offset	数据偏移量，默认值为 0	Int	否
limit	返回数据长度，默认值为 20	Int	否
order	以升序还是降序的方式对日志进行排序，可选值 desc 和 asc	String	否
orderby	根据某个字段排序日志，支持以下字段：startTime、functionName、requestId、duration 和 memUsage	String	否
filter	日志过滤条件。可用来区分正确和错误日志，filter.retCode=not0 表示只返回错误日志，filter.retCode=is0 表示只返回正确日志，不传，则返回所有日志。	Object	否

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "total": 2,
  "data": [
    {
      "functionName": "testnode",
      "retMsg": "\"ok\"",
      "requestId": "b5a9c642-9c51-11e7-a46e-5254001df6c6",
      "startTime": "2017-09-18 17:14:01",
      "retCode": 0,
    }
  ]
}
```

```
"invokeFinished": 1,
"duration": 2.735,
"billDuration": 100,
"memUsage": 131072,
"log": "2017-09-18T09:14:01.236Z\tundefined\tHello
World\n2017-09-18T09:14:01.236Z\tundefined\t{ Records: [ { CMQ: [Object] } ]
}\n2017-09-18T09:14:01.237Z\tundefined\t{ msgBody: '4',\n msgId: '3096224743817224',\n msgTag:
',\n\n publishTime: '2017-09-18T17:14:01Z',\n requestId: '5761047512720426890',\n subscriptionName:
'alfred-test',\n topicName: 'alfred-test',\n topicOwner: 1251762227,\n type: 'topic'
}\n2017-09-18T09:14:01.237Z\tundefined\t{ callbackWaitsForEmptyEventLoop: [Getter/Setter],\n
done: [Function: done],\n succeed: [Function: succeed],\n fail: [Function: fail],\n memory_limit_in_mb:
128,\n time_limit_in_ms: 30000 }\n"
},
{
"functionName": "testnode",
"retMsg": "\"ok\"",
"requestId": "b33b9d0b-9c51-11e7-b38f-525400c7c826",
"startTime": "2017-09-18 17:13:57",
"retCode": 0,
"invokeFinished": 1,
"duration": 7.437,
"billDuration": 100,
"memUsage": 131072,
"log": "2017-09-18T09:13:57.155Z\tundefined\tHello
World\n2017-09-18T09:13:57.156Z\tundefined\t{ Records: [ { CMQ: [Object] } ]
}\n2017-09-18T09:13:57.158Z\tundefined\t{ msgBody: '3',\n msgId: '3096224743817223',\n msgTag:
',\n\n publishTime: '2017-09-18T17:13:57Z',\n requestId: '5761047512720426853',\n subscriptionName:
'alfred-test',\n topicName: 'alfred-test',\n topicOwner: 1251762227,\n type: 'topic'
}\n2017-09-18T09:13:57.159Z\tundefined\t{ callbackWaitsForEmptyEventLoop: [Getter/Setter],\n
done: [Function: done],\n succeed: [Function: succeed],\n fail: [Function: fail],\n memory_limit_in_mb:
128,\n time_limit_in_ms: 30000 }\n"
}
]
}
```

响应参数

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String
total	日志总量	Int
data	返回日志信息列表	Object

返回的每条日志数据包含如下数据结构：

参数名称	描述	类型
function_name	该日志对应的函数名称	String
retMsg	函数执行完成后的返回值	String
requestId	执行该函数对应的requestId	String
startTime	函数开始执行时的时间点	String
ret_code	函数执行结果，如果是 0 表示执行成功，其他值表示失败	Int
invokeFinished	函数调用是否结束，如果是 1 表示执行结束，其他值表示调用异常	Int
duration	函数执行耗时，单位为 ms	Float
billDuration	函数计费时间，根据 duration 向上取最近的 100ms，单位为ms	Int
memUsage	函数执行时消耗实际内存大小，单位 为 Byte	Int
log	函数执行过程中的日志输出	String

实际案例

请求

```
GET https://scf.api.qcloud.com/v2/index.php?
Action=GetFunctionLogs
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu
&Nonce=47400
&Timestamp=1505726068
&Region=gz
&Signature=gcFINOIX37qkwL%2FIkiRY9hQDSkM%3D
&functionName=testnode
&limit=2
```

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "total": 2,
  "data": [
    {
      "functionName": "testnode",
      "retMsg": "\"ok\"",
      "requestId": "b5a9c642-9c51-11e7-a46e-5254001df6c6",
      "startTime": "2017-09-18 17:14:01",
      "retCode": 0,
      "invokeFinished": 1,
      "duration": 2.735,
      "billDuration": 100,
      "memUsage": 131072,
      "log": "2017-09-18T09:14:01.236Z\tundefined\tHello
World\n2017-09-18T09:14:01.236Z\tundefined\t{ Records: [ { CMQ: [Object] } ]
}\n2017-09-18T09:14:01.237Z\tundefined\t{ msgBody: '4',\n msgId: '3096224743817224',\n msgTag:
',\n publishTime: '2017-09-18T17:14:01Z',\n requestId: '5761047512720426890',\n subscriptionName:
'alfred-test',\n topicName: 'alfred-test',\n topicOwner: 1251762227,\n type: 'topic'
"
```

```
}\\n2017-09-18T09:14:01.237Z\\tundefined\\t{ callbackWaitsForEmptyEventLoop: [Getter/Setter],\\n
done: [Function: done],\\n succeed: [Function: succeed],\\n fail: [Function: fail],\\n memory_limit_in_mb:
128,\\n time_limit_in_ms: 30000 }\\n"
},
{
  "functionName": "testnode",
  "retMsg": "\\n\\n",
  "requestId": "b33b9d0b-9c51-11e7-b38f-525400c7c826",
  "startTime": "2017-09-18 17:13:57",
  "retCode": 0,
  "invokeFinished": 1,
  "duration": 7.437,
  "billDuration": 100,
  "memUsage": 131072,
  "log": "2017-09-18T09:13:57.155Z\\tundefined\\tHello
World\\n2017-09-18T09:13:57.156Z\\tundefined\\t{ Records: [ { CMQ: [Object] } ]
}\\n2017-09-18T09:13:57.158Z\\tundefined\\t{ msgBody: '3',\\n msgId: '3096224743817223',\\n msgTag:
'',\\n publishTime: '2017-09-18T17:13:57Z',\\n requestId: '5761047512720426853',\\n subscriptionName:
'alfred-test',\\n topicName: 'alfred-test',\\n topicOwner: 1251762227,\\n type: 'topic'
}\\n2017-09-18T09:13:57.159Z\\tundefined\\t{ callbackWaitsForEmptyEventLoop: [Getter/Setter],\\n
done: [Function: done],\\n succeed: [Function: succeed],\\n fail: [Function: fail],\\n memory_limit_in_mb:
128,\\n time_limit_in_ms: 30000 }\\n"
}
]
}
```

获取函数监控数据

接口描述

功能描述

GetMonitorData

接口提供了获取无服务器云函数的监控数据。可以根据用户传入的函数名获得相应的监控数据。

接口域名

接口请求域名为：

monitor.api.qcloud.com

请求

语法示例：

GET https://monitor.api.qcloud.com/v2/index.php?Action=GetMonitorData

&<公共请求参数>

&namespace=qce/scf

&metricName=duration

&dimensions.0.name=functionName

&dimensions.0.value=test

&startTime=2017-07-24 14:00:00

&endTime=2017-07-24 14:05:00

&period=300

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#)

页面。其中，此接口的 Action 字段为 CreateFunction。

参数名称	描述	类型	必选
namespace	命名空间，每个云产品会有一个命名空间。无服务器云函数的命名空间名为： qce/scf	String	是
metricName	指标名称，想要获取的具体监控指标，如运行时间是 duration，调用次数是 invocation 等	String	是
dimensions.n.name	纬度的名称。当前无服务器云函数只有一个维度：functionName，与 dimensions.n.value 配合使用	String	是
dimensions.n.value	纬度名称对应的值	String	是
startTime	起始时间，如 " 2017-01-01 00:00:00" 。默认时间为当天的 " 00:00:00" 。	Datetime	否
endTime	结束时间，如 " 2017-01-01 10:00:00" 。默认为当前时间。 注意：endTime 不能小于startTime，且 endTime 与 startTime 最好为同一天	Datetime	否
period	监控统计周期。当前支持 60s 和 300s 两种粒度，如 不传，默认使用 300s 粒度。	Int	否

当前无服务器云函数可展示的指标 (metricName) 列表如下：

指标名称	含义	单位
duration	运行时间	毫秒(ms)
invocation	调用次数	次
error	调用错误次数	次

当前无服务器云函数只支持云函数名称维度，说明如下：

此维度体现的是一个云函数的监控指标，需要传入的维度 (dimensions.n.name) 如下：

维度	维度解释	格式
functionName	无服务器云函数名称	string

如果您想查询名称为 test 的云函数的运行时间 (duration) 的监控数据，则 API 请求示例如下：

GET https://monitor.api.qcloud.com/v2/index.php?Action=GetMonitorData

&<公共请求参数>

&namespace=qce/scf

&metricName=duration

&dimensions.0.name=functionName

&dimensions.0.value=test

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "metricName": "duration",
  "startTime": "2017-07-24 14:00:00",
  "endTime": "2017-07-24 14:05:00",
  "period": 300,
```

```
"dataPoints": [  
  0.65213,  
  0.5586  
]  
}
```

响应参数

参数名称	描述	类型
code	公共错误码，0 表示成功，其他值表示失败。详见错误码页面的 公共错误码	Int
codeDesc	英文错误码	String
message	错误信息详细描述	String
startTime	起始时间	Datetime
endTime	结束时间	Datetime
metricName	指标名称	String
period	监控统计周期	Int
dataPoints	监控数据列表。数组每个元素是监控点数据	Object

实际案例

请求

```
GET https://monitor.api.qcloud.com/v2/index.php?  
  Action=GetMonitorData  
  &SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu  
  &Nonce=62089  
  &Timestamp=1505804102  
  &Region=gz  
  &Signature=LWI%2BdudHPe56OYcsvKWiXImdO5s%3D  
  &namespace=qce%2Fscf
```

```
&metricName=duration
&dimensions.0.name=functionName
&dimensions.0.value=test5
&startTime=2017-09-19+12%3A00%3A00
&endTime=2017-09-19+14%3A05%3A00
&period=300
```

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "metricName": "duration",
  "startTime": "2017-09-19 12:00:00",
  "endTime": "2017-09-19 14:05:00",
  "period": "300",
  "dataPoints": [
    2.983,
    1.131,
    0.732,
    0.773,
    0.921,
    0.901,
    0.775,
    0.931,
    0.812,
    0.757,
    1.06,
    0.805,
    0.834,
    0.812,
    0.881,
```

```
0.807,  
0.831,  
0.992,  
0.793,  
1.968,  
0.868,  
0.93,  
6.365,  
0.732,  
0.836,  
1.433  
]  
}
```

运行函数

接口描述

功能描述

InvokeFunction

接口可以触发指定的函数运行，目前支持同步和异步两种触发方式。同步调用的情况下返回值会多一个

payLoad

字段用于表示函数的返回值。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的Action字段为 InvokeFunction。

参数名称	描述	类型	必选
functionName	需要运行的函数名称	String	是
invokeType	触发类型。可选值为 RequestResponse(同步) 和	String	否

参数名称	描述	类型	必选
	Event(异步) ，默认为同步		
param	运行函数时的参数，以json 格式传入，最大支持的参数 长度是 1M	String	否
logType	同步调用时指定该字段，返 回值会包含 4K 的日志，可选值为 None 和 Tail ，默认值为 None 。当该值为 Tail 时，返回参数中的 logMsg	String	否

参数名称	描述	类型	必选
	字段会包含对应的函数执行日志		

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": {
    "duration": 0.422,
    "memUsage": 126976,
    "log": "log in func\n",
    "requestId": "c471ac84-2f15-11e7-8501-5254007d2563",
    "retMsg": "Hello from SCF"
  },
  "billDuration": 100
}
```

响应参数

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String
data	json格式，包含本次函数执行的相关 信息	Object

data字段中包含的数据结构如下：

参数名称	描述	类型
requestId	本次执行函数的requestId	String
logMsg	同步调用时才有此返回，表示执行过程中的日志输出	String
retMsg	同步调用时才有此返回，表示执行函数的返回	String
errMsg	同步调用时才有此返回，表示执行函数的错误返回信息	String
memUsage	同步调用时才有此返回，执行函数时的内存大小，单位为Byte	Int
duration	同步调用时才有值，表示执行函数的耗时，单位是毫秒	Int
billDuration	同步调用时才有值，表示函数的计费耗时，单位是毫秒	Int

实际案例

异步请求

请求

```
GET https://scf.api.qcloud.com/v2/index.php?  
Action=InvokeFunction  
&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu  
&Nonce=1306  
&Timestamp=1505728098  
&Region=gz  
&Signature=T%2FFgPfv0zgig8sy72EcUnVUXPkw%3D  
&functionName=testnode  
&param=%7B%22test%22%3A%22value%22%7D  
&invokeType=Event
```

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success",
  "data": {
    "requestId": "7fe1685d-9c56-11e7-b38f-525400c7c826"
  }
}
```

同步请求

请求

```
GET https://scf.api.qcloud.com/v2/index.php?
  Action=InvokeFunction
  &SecretId=AKIDutrojKl3CKQZNAr763UXks05898Lmciu
  &Nonce=39288
  &Timestamp=1505728154
  &Region=gz
  &Signature=4RWbHaCfd5ZletZf4p8jUxN%2BqgU%3D
  &functionName=testnode
  &param=%7B%22test%22%3A%22value%22%7D
  &invokeType=RequestResponse
  &logType=Tail
```

响应

```
{
  "code": 0,
  "message": "",
```

```
"codeDesc": "Success",
"data": {
  "memUsage": 131072,
  "log": "2017-09-18T09:49:15.049Z\tundefined\t{ test: 'value' }\n",
  "retMsg": "\"ok\"",
  "billDuration": 100,
  "requestId": "a197c584-9c56-11e7-a46e-5254001df6c6",
  "duration": 2.278,
  "errMsg": "",
  "invokeResult": 0
}
```

触发器相关接口

设置函数触发器

接口描述

功能描述

SetTrigger 创建和修改函数触发器。目前支持 cos、cmq、timer 三种类型的触发器，每个函数最大支持 4 个 触发器。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

请求示例：

GET https://scf.api.qcloud.com/v2/index.php?Action=SetTrigger

&<公共请求参数>

&functionName=hell

&triggerName=test1

&type=timer

&triggerDesc=*/2 * * * *

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 SetTrigger。

参数名称	描述	类型	必选
functionName	新建触发器绑定的函数名称	String	是
triggerName	新建触发器名称。如果是定时触发器，名称支持英文字母、数字、连接符和下划线，最长100个字符；如果是其他触发器，见具体触发器绑定参数的说明	String	是
type	触发器类型，目前支持 cos、cmq、timer 类型	String	是
triggerDesc	触发器对应的参数，如果是 timer 类型的触发器其内容是 Linux cron 表达式，如果是其他触发器，见具体触发器说明	String	是
newTriggerName	更新定时触发器名称时使用该参数，新名称支持英文字母、数字、连接符和下划线，最长 100 个字符	String	否

注意事项：

- 一个帐号在一个区域最多支持 20 个函数，每个函数最多拥有 4 个触发器。
- 一个区域的 COS Bucket 只能绑定该区域的函数，不允许 Bucket 和函数之间建立跨区域绑定
- cos 触发器不支持更新操作。
- 一个区域的 cmq topic 队列只能绑定该区域的函数，不允许 cmq topic 队列和函数之间建立跨区域绑定。
- cmq 触发器不支持更新操作。

COS 绑定参数

在绑定 COS Bucket 事件时，触发器的 triggerName 格式是固定，格式如下：

<bucketName>-<UID>.<Region>.myqcloud.com

。

triggerDesc 中需要传入 json 格式的数据内容示例如下：

```
{"event": "cos:ObjectCreated:*"}
```

目前支持的 event 值如下表所示，绑定 COS 事件到具体的触发器时，存在如下绑定限制：

1. 如果先绑定

cos:ObjectCreated:*

事件，后续再绑定

cos:ObjectCreated

开头的事件都会失败。

2. 如果先绑定

cos:ObjectCreated

开头的除

cos:ObjectCreated:*

之外的事件，后续再绑定

cos:ObjectCreated:*

事件会失败。

3. 如果先绑定

cos:ObjectRemove:*

事件，后续再绑定

cos:ObjectRemove

开头的事件都会失败。

4. 如果先绑定

cos:ObjectRemove

头的除

cos:ObjectRemove:*

之外的事件，后续再绑定

cos:ObjectRemove:*

事件会失败。

5. 如果 cos 已经被绑定，执行创建 cos 触发器将失败。

6. cos 触发器不支持名称更新操作。

事件	描述
cos:ObjectCreated:Put	使用 Put Object 接口创建文件
cos:ObjectCreated:Post	使用 Post Object 接口创建文件
cos:ObjectCreated:Copy	使用 Put Object - Copy 接口创建文件
cos:ObjectCreated:Append	使用 Append Object 接口创建文件
cos:ObjectCreated:CompleteMultipartUpload	使用 CompleteMultipartUploadt 接口创建文件
cos:ObjectCreated:*	使用以上提到的 "ObjectCreated" 类型接口创建文件
cos:ObjectRemove>Delete	在未开启版本管理的 Bucket 下使用 Delete Object 接口删除的 Object，或者使用 versionid 删除指定版本的 Object
cos:ObjectRemove>DeleteMarkerCreated	在开启或者暂停版本管理的 Bucket 下使用 Delete

事件	描述
	Object 接口删除的 Object
cos:ObjectRemove:*	使用以上提到的 "ObjectRemove" 类型接口删除文件

注意

使用 cos 触发器时如果函数的执行结果需要保存到 bucket 上，需要将触发源的bucket和输出结果的bucket设置为两个不同的bucket，避免出现循环依赖导致函数持续运行。

CMQ 绑定参数

在绑定 CMQ Topic 队列时，触发器的 triggerName 写为 CMQ Topic 队列的主题名即可，例如

triggerName=test-topic

。

triggerDesc 无需填写。

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

响应参数

参数名称	描述	类型

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String

实际案例

设置定时触发器

请求

```
GET https://scf.api.qcloud.com/v2/index.php?  
Action=SetTrigger  
&SecretId=AKIDutrojKl3CKQZNAr763UXks05898Lmciu  
&Nonce=23569  
&Timestamp=1505787511  
&Region=gz  
&Signature=5%2Bz3gBf7HrTevF0VoxO8iQViPl8%3D  
&functionName=test5  
&triggerName=time1  
&type=timer  
&triggerDesc=%2A%2F2+%2A+%2A+%2A+%2A
```

响应

```
{  
  "code": 0,  
  "message": "",  
  "codeDesc": "Success"  
}
```

设置 COS 触发器

请求

GET https://scf.api.qcloud.com/v2/index.php?

Action=SetTrigger

&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu

&Nonce=35339

&Timestamp=1505788030

&Region=gz

&Signature=xBqiia%2FWIA1ar7wyidagaoI5yZs%3D

&functionName=test5

&triggerName=scfdest-1251762227.cn-south.myqcloud.com

&type=cos

&triggerDesc=%7B%22event%22%3A%22cos%3AObjectCreated%3A%2A%22%7D

响应

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

设置 CMQ Topic 触发器

请求

GET https://scf.api.qcloud.com/v2/index.php?

Action=SetTrigger

&SecretId=AKIDutrojKI3CKQZNAr763UXks05898Lmciu

&Nonce=65137

&Timestamp=1505792960

&Region=gz
&Signature=6RWWnuBZSIEHoRxNQy6qUMB%2F6ss%3D
&functionName=test5
&triggerName=test-topic
&type=cmq

响应

```
{  
  "code": 0,  
  "message": "",  
  "codeDesc": "Success"  
}
```

删除函数触发器

接口描述

功能描述

DeleteTrigger 根据指定的函数名称和触发器名称，删除对应的触发器。

接口域名

接口请求域名为：

scf.api.qcloud.com

请求

请求示例：

GET https://scf.api.qcloud.com/v2/index.php?Action=DeleteTrigger

&<公共请求参数>

&functionName=hell

&type=timer

&triggerName=test2

请求参数

以下请求参数为本接口接口请求参数，正式调用时需要加上公共请求参数，详见 [公共请求参数](#) 页面。其中，此接口的 Action 字段为 DeleteTrigger。

参数名称	描述	类型	必选
functionName	函数名称	String	是
triggerName	要删除的触发器名称。	String	是

参数名称	描述	类型	必选
type	要删除的触发器类型，目前支持 cos、cmq、timer 类型	String	是
triggerDesc	如果删除的触发器类型为 COS 触发器，该字段为必填值，存放 JSON 格式的数据 <pre>{"event": "cos:ObjectCreated:*"}</pre> ，数据内容和 SetTrigger 接口中该字段的格式相同；如果删除的触发器类型为定时触发器或 CMQ 触发器，可以不指定该字段	String	是

响应

响应示例：

```
{
  "code": 0,
  "message": "",
  "codeDesc": "Success"
}
```

响应参数

参数名称	描述	类型
code	公共错误码， 0 表示成功，其他值表示失败	Int

参数名称	描述	类型
message	模块错误信息描述，与接口相关	String
codeDesc	英文错误码，成功返回 Success，失败有相应的英文说明	String

实际案例

请求

```
GET https://scf.api.qcloud.com/v2/index.php?  
Action=DeleteTrigger  
&SecretId=AKIDutrojKl3CKQZNAr763UXks05898Lmciu  
&Nonce=51955  
&Timestamp=1505793130  
&Region=gz  
&Signature=LaOsxyEvPzjIZTLQWk3AcM60%2FPc%3D  
&functionName=test5  
&triggerName=test-topic  
&type=cmq
```

响应

```
{  
  "code": 0,  
  "message": "",  
  "codeDesc": "Success"  
}
```