目录

API参考手册

文档版本--2016-09-13

什么是云数据库Memcached?

云数据库Memcached是金山云推出的兼容memcacehd协议的缓存服务。支持主从热备，提供自动容灾切换、实例监控、在线扩容等数据库服务。

服务提供了用于实例操作的OpenAPI，可以根据需要开发应用程序，实现实例的创建，更配、清空 等多种自动化任务。

文档概述

本指南提供了API的说明以及使用案例。
通用请求部分介绍了API服务的基本信息，公共参数，签名机制和报错信息等内容。
实例管理部分介绍了实例操作的API和编程实例。

数据类型部分介绍了 API自定义的数据类型。

主要内容

服务信息

服务信息

- **服务地址** memcached.api.ksyun.com
- **通讯协议** 支持HTTP或HTTPS协议
- **请求方法** 详见具体API接口
- **请求参数** 每个请求都需要包含公共的请求参数和操作所特有的请求参数，详见具体API接口
- **编码方式** 请求及结果返回都使用UTF-8字符集编码
- **返回结果** 仅支持json返回结果

Region及可用区

各个region都通过同一个endpoint来访问API服务：memcached.api.ksyun.com

Region名称	类型	Region代码	可用区
北京6	VPC机房	cn-beijing-6	cn-being-6b
上海2	VPC机房	cn-shanghai-2	cn-hanghai-2a

公共参数

支持GET和POST两种HTTP方法。

固定请求域名：memcached.api.ksyun.com（不指定区域则默认为cn-beijing-6，若要指定区域：memcached.{Region}.api.ksyun.com）

云数据库Memcached的服务名称(Service)指定为memcached

详情请参考：公共参数

签名机制

数据库的openAPI支持GET和POST两种HTTP方法，具体流程请参见签名机制。

Python Demo

```
import hashlib
import hmac
import time
import urllib.request

import requests

# Python 3.9.6
def http_execute(methed, host, heads, data=None):
    url = 'https://%s/?' % host
    for key in data:
        url += urllib.request.quote(key, '~') + '=' + urllib.request.quote(str(data[key]), '~') + '&'
    response = requests.request(methed, url, headers=heads, data=None)
    return response;

# sigon方法
def sign(params, secret_key):
    str_encode = ''
    param_keys = sorted(params.keys())
    for key in param_keys:
        str_encode += urllib.request.quote(key, '~') + '=' + urllib.request.quote(str(params[key]), '~') + '&'
    return hmac.new(bytes(secret_key, 'utf-8'), bytes(str_encode[:-1], 'utf-8'), hashlib.sha256).hexdigest()

class ksyun_iam_api_tools():
    """金山云API相关"""

    def __init__(self):
        self.AK = 'your_ak'
        self.SK = 'your_sk'
        self.SERVICE = "krds"
        self.REGION = "cn-beijing-6"
        self.HOST = "%s.%s.api.ksyun.com" % (self.SERVICE, self.REGION)
        self.additional_headers = {'Accept': 'application/json'}

    def describe_db_instances(self):
        data = {
            'Accesskey': self.AK,
            'Service': 'krds',
            'Action': 'DescribeDBInstances',
            'Version': '2016-07-01',
            'Timestamp': time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime()),  # 使用UTC格式的时间
            'SignatureVersion': '1.0',
            'SignatureMethod': 'HMAC-SHA256',
        }
        data["Signature"] = sign(data, self.SK)
        return http_execute("GET", self.HOST, self.additional_headers, data)

if __name__ == '__main__':
    api = ksyun_iam_api_tools()
    instances = api.describe_db_instances()
    print(instances.text)
```

Java Demo

```
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import org.apache.commons.codec.digest.HmacAlgorithms;
```

```
import org.apache.commons.codec.digest.HmacUtils;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

// jdk1.8.0_241
public class ram_demo {
    private static final Pattern ENCODED_CHARACTERS_PATTERN;

    static {
        StringBuilder pattern = new StringBuilder();
        pattern.append(Pattern.quote("+"))
                .append("|")
                .append(Pattern.quote("*"))
                .append("|")
                .append(Pattern.quote("%7E"))
                .append("|")
                .append(Pattern.quote("%2F"));
        ENCODED_CHARACTERS_PATTERN = Pattern.compile(pattern.toString());
    }

    /**
     * 注意空格( )会编码成+号，所以+最后需要替换成%20（%20为空格）
     * 在URLEncode后需对三种字符替换：加号（+）替换成 %20、星号（*）替换成 %2A、 %7E 替换回波浪号（~）
     */
    private static String urlEncode(final String value, final boolean path) {
        if (value == null) {
            return "";
        }

        try {

            String encoded = URLEncoder.encode(value, StandardCharsets.UTF_8.name());
            Matcher matcher = ENCODED_CHARACTERS_PATTERN.matcher(encoded);
            StringBuffer buffer = new StringBuffer(encoded.length());

            while (matcher.find()) {
                String replacement = matcher.group(0);
                if ("+".equals(replacement)) {
                    replacement = "%20";
                } else if ("*".equals(replacement)) {
                    replacement = "%2A";
                } else if ("%7E".equals(replacement)) {
                    replacement = "~";
                } else if (path && "%2F".equals(replacement)) {
                    replacement = "/";
                }

                matcher.appendReplacement(buffer, replacement);
            }
            matcher.appendTail(buffer);
            return buffer.toString();
        } catch (UnsupportedEncodingException ex) {
            throw new RuntimeException(ex);
        }
    }

    private static String getCanonicalizedQueryString(Map<String, Object> params) {
        final Map<String, String> tmap = new TreeMap<>();
        for (Map.Entry<String, Object> entry : params.entrySet()) {
            String key = entry.getKey();
            Object value = entry.getValue();
            String encodedParamName = urlEncode(key, false);
            String encodedValues = urlEncode(value.toString(), false);
            tmap.put(encodedParamName, encodedValues);
        }

        final StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, String> entry : tmap.entrySet()) {
            if (sb.length() > 0) {
                sb.append("&");
            }
            sb.append(entry.getKey()).append('=').append(entry.getValue());
        }
        return sb.toString();
    }
```

```
    public static String signature(Map<String, Object> params, String secretKey) {
        String canonicalizedQueryString = getCanonicalizedQueryString(params);
        return new HmacUtils(HmacAlgorithms.HMAC_SHA_256, secretKey).hmacHex(canonicalizedQueryString);
    }

    public static String utc2Local(String utcTime, String utcTimePatten, String localTimePatten) throws ParseException {
        SimpleDateFormat utcFormater = new SimpleDateFormat(utcTimePatten);
        utcFormater.setTimeZone(TimeZone.getTimeZone("UTC"));//时区定义并进行时间获取
        Date gpsUTCDate = null;
        gpsUTCDate = utcFormater.parse(utcTime);
        SimpleDateFormat localFormater = new SimpleDateFormat(localTimePatten);
        localFormater.setTimeZone(TimeZone.getDefault());
        String localTime = localFormater.format(gpsUTCDate.getTime());
        return localTime;
    }

    public static void test1() throws IOException {
        String accesskey = "your_ak";
        String secretKey = "your_sk";

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        TimeZone zone = TimeZone.getTimeZone("UTC");
        Calendar cal = Calendar.getInstance(zone);
        sdf.setTimeZone(zone);

        Map<String, Object> params = new HashMap<>();
        params.put("Accesskey", accesskey); //公共参数
        params.put("Service", "krds"); //公共参数
        params.put("Action", "DescribeDBInstances"); //公共参数
        params.put("Version", "2016-07-01"); //公共参数
        params.put("Timestamp", sdf.format(cal.getTime())); //公共参数
        params.put("SignatureVersion", "1.0"); //公共参数
        params.put("SignatureMethod", "HMAC-SHA256"); //公共参数
        params.put("Signature", signature(params, secretKey));

        String product_name = "krds";
        String region = "cn-beijing-6";

        String url = "https://" + product_name + "." + region + ".api.ksyun.com/?" + getCanonicalizedQueryString(params);
        OkHttpClient client = new OkHttpClient().newBuilder()
                .build();
        Request request = new Request.Builder()
                .url(url)
                .method("GET", null)
                .addHeader("Accept", "application/json")
                .build();
        Response response = client.newCall(request).execute();

        System.out.println(new String(response.body().bytes()));
    }

    public static void main(String[] args) {
        try {
            test1();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Go Demo

```
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
    "io/ioutil"
    "net/http"
    "net/url"
    "strings"
    "time"
)

// go version go1.16.5 darwin/amd64

//获取signature签名
func sign(params url.Values, sk string) string {
    strEncode := params.Encode()
    strEncode = strings.Replace(strEncode, "+", "%20", -1)
    h := hmac.New(sha256.New, []byte(sk))
```

```
    h.Write([]byte(strEncode))
    return hex.EncodeToString(h.Sum(nil))
}

func main() {

    params := url.Values{}
    params.Add("Accesskey", "your_ak")
    params.Add("Service", "krds")
    params.Add("Action", "DescribeDBInstances")
    params.Add("Version", "2016-07-01")
    params.Add("Timestamp", time.Now().UTC().Format("2006-01-02T15:04:05Z")) //通过time.Now().UTC().Format("2006-01-02T15:04:05Z
")实现
    params.Add("SignatureVersion", "1.0")
    params.Add("SignatureMethod", "HMAC-SHA256")
    sk := "your_sk"
    signature := sign(params, sk)
    params.Add("Signature", signature)

    strEncode := params.Encode()
    strEncode = strings.Replace(strEncode, "+", "%20", -1)

    url := "https://krds.cn-beijing-6.api.ksyun.com/?" + strEncode
    method := "GET"
    client := &http.Client{}
    req, _ := http.NewRequest(method, url, nil)
    req.Header.Add("Accept", "application/json")
    res, _ := client.Do(req)
    defer res.Body.Close()
    body, _ := ioutil.ReadAll(res.Body)
    fmt.Println(string(body))
}
```

通用错误信息

Code值	解释	HTTP Status Code
MissingParameter	表示请求参数不	400
MissingAction	表示没有提示出正确的Action 或 Version	400
InvalidParameter	非法的请求参数	400
RequestExpired	请求过期	400
ServiceUnavailable	服务不可用	503
ValidationError	请求验证失败	400
InvalidAction	非法的操作	400
BusinessError	程序中业务错误信息提示	400
InternalFailure	内部错误	500

缓存服务状态通用信息

status值	简义	解释
1	building	创建中
2	running	运行中
3	resizing	扩容中
4	deleting	删除中
5	locking	锁定中
6	unlocking	解锁中
7	configing	配置中
8	locked	已锁定
10	deleted	已删除
99	error	异常

CreateCacheCluster

创建主从实例。可以指定内存大小、网络类型以及使用时长

请求方式

POST

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	string	是	取固定值：CreateCacheCluster
Version	API版本号	String	是	取固定值： 2018-06-27
Name	缓存服务名称	String	是	支持6-64个中文或者英文字符，包括汉字，大小写字母，数字，下划线和中划线
Capacity	缓存容量大小，以GB为单位	integer	是	缓存资源为单主从模式时，可选值为：{1, 2, 4, 8, 16, 32, 64}；
SlaveNum	从节点个数据	Snteger	否	范围是 0~7个， 默认值为0
NetType	网络类型	Integer	是	固定值2， 目前只支持创建VPC实例。
VpcId	虚拟专用网络	String	是	VPC网络ID，可在网络控制台获取。
VnetId	终端子网id	String	是	终端子网ID，可在网络控制台获取（注意类型必须为终端子网）。
BillType	计费方式：默认为1	Integer	否	1:包年包月 5:按天先结
Duration	时长，默认值：1(单位:月)	Integer	否	billType=1(包年包月)则必填，最大支持范围是(1 ~36月)，开发自定
DurationUnit	时长单位	String	否	默认值：月
PassWord	密码	String	否	规则：(?=.*[A-Z]+.*)(?=.*[a-z]+.*)(?=.*[\d]+.*)([A-Za-z\d!@#$%^&*()_+=-]{8,30})
IamProjectId	项目ID	String	否	默认为0：默认项目
Engine	缓存服务引擎	String	是	取固定值：memcached

返回结果

英文名称	中文名称	类型	是否必须	备注
RequestId	请求ID	string	是	--
Data	返回数据信息	object	否	返回列表的数据信息，具体可以查看参数列表中的返回例子
Error	错误	string	否	Code：错误状态码 Message：文字描述

错误信息

关于所有操作返回的错误信息，请参照通用请求一通用错误信息

样例

- **请求样例**

```
http://memcached.api.ksyun.com/
?Action=CreateCacheCluster
&Version=2018-06-27
&Name=MyRedisCluster
&Capacity=1
&NetType=2
&VpcId=b33a2276-64a8-4c04-b28e-da253c8add32
&VnetId=c2e0abd7-13df-461a-bd8d-3b92faebf111
&BillType=5
&IamProjectId=0
&Engine=memcached
&PassWord=Xxxxx
```

- **返回样例**

```
{
    "RequestId": "332ab1f4-5f00-4fc1-90b7-1e87d0b6834b",
    "Data": {
        "CacheId": "643e17d1-a2f5-4aff-95d3-c630206cfdd7",
        "Name": "bing-openapi-define004",
        "Size": "1",
        "Port": "6379",
        "SubOrderId": "MREDIS2S171220155806121485666"
```

```
    }
}
```

DescribeCacheClusters

查询缓存服务列表，允许您根据特定的条件查询缓存资源，并支持对缓存资源的排序，默认每次查出10个缓存服务。

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：DescribeCacheClusters
Engine	服务引擎	String	是	取固定值memcached
Version	API版本号	String	是	取固定值： 2018-06-27
CacheId	缓存服务ID	String	否	
Name	缓存服务名称	String	否	
Vip	缓存服务IP地址	String	否	
VpcId	虚拟专用网络ID	String	否	只适用于VPC网络下的缓存服务
VnetId	虚拟专用网路EndpointID，或vpclbID	String	否	只适用于VPC网络下的缓存服务
Offset	查询数据的起始位置	Integer	否	默认为0
Limit	需要从起始位置开始查询的缓存服务的个数	Integer	否	取值范围为[1~100]，默认为10
OrderBy	排序字段	String	否	可传值为{name,asc，name,desc，created,asc，created,desc}，默认按照创建时间降序，只有排序字段时，默认按照升序排列
IamProjectId	项目ID	String	否	默认是0(默认项目),如果查询全部项目，需要传入所有的项目ID，‘,’隔开

请求方式

GET

返回结果

请求的返回信息请参照样例,包含以下字段。

英文名称	中文名称	类型	备注
cacheId	缓存服务ID	string	
name	缓存服务名称	string	--
engine	缓存服务引擎	string	
mode	缓存服务方式	byte	1表示集群方式，2表示单主从方式
size	缓存服务内存大小	integer	单位：GB
port	缓存服务端口号	integer	
vip	缓存服务IP地址	string	
status	缓存服务当前状态	byte	对照关系请见缓存服务状态通用信息
createTime	缓存服务创建时间	date	
netType	缓存服务网络类型	byte	1表示基础网络，2表示VPC网络
vpcId	虚拟专用网络ID	string	只适用于VPC网络下的缓存服务
vnetId	虚拟专用网路EndpointID，或vpclbID	string	只适用于VPC网络下的缓存服务
billType	订单类型	integer	包年包月
serviceBeginTime	服务开始时间	date	
serviceEndTime	服务结束时间	date	
offset	查询数据的起始位置	integer	
limit	需要从起始位置开始查询的缓存服务的个数	integer	
total	总记录数	integer	
iamProjectId	项目ID	string	
iamProjectId	项目名称	string	
protocol	实例版本	string	

错误信息

关于所有操作返回的错误信息，请参照通用请求-通用错误信息

请求样例

```
https://memcached.cn-shanghai-2.api.ksyun.com/
?Action=DescribeCacheClusters
&Version=2018-06-27
&Engine=memcached
&IamProjectId=0,20,21
&OrderBy=name,desc
&Offset=0
&Limit=2
```

返回样例

```
HttpStatusCode=200
```


```
{
  "reqId": "xxxxxxxxxx",
  "code": 0,
  "message": "查询缓存服务列表成功",
  "data": {
    "list": [
      {
        "cacheId": "b4a45b78-7baa-41a0-937e-63b94c3d7390",
        "name": "hdh-test-2",
        "securityGroupId": "687",
        "engine": "redis",
        "mode": 2,
        "size": 1,
        "port": 6379,
        "vip": "172.31.253.133",
        "status": 2,
        "createTime": "2016-08-08 20:14:49",
        "netType": 2,
        "vpcId": "95f66aa8-75a7-4e16-ba1e-db4e2d532ac4",
        "vnetId": "21d0c9f5-228f-4330-a169-5f896453c117",
        "iamProjectId":"0",
        "iamProjectName":"默认项目"
      },
      {
        "cacheId": "4bd89c73-b51d-47ed-a4e3-88d88ff14434",
        "name": "gbz-test-2",
        "securityGroupId": "683",
        "engine": "redis",
        "mode": 2,
        "size": 1,
        "port": 6379,
        "vip": "172.31.253.131",
        "status": 2,
        "createTime": "2016-08-03 15:14:20",
        "netType": 2,
        "vpcId": "95f66aa8-75a7-4e16-ba1e-db4e2d532ac4",
        "vnetId": "21d0c9f5-228f-4330-a169-5f896453c117",
        "iamProjectId":"0",
        "iamProjectName":"默认项目"
      }
    ],
    "offset": 0,
    "limit": 2,
    "total": 10
  }
}
```


DescribeCacheCluster

查询缓存服务详情，根据缓存服务ID获得缓存资源的详细信息以及该缓存资源的订单信息

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注

Action	调用接口名称	string	是	取固定值： DescribeCacheCluster
Version	API版本号	string	是	取固定值： 2018-06-27
CacheId	缓存服务ID	string	是	
Engine	缓存服务引擎	string	是	取固定值： memcached

请求方式

GET

返回结果

英文名称	中文名称	类型	备注
cacheId	缓存服务ID	String	
name	缓存服务名称	String	--
engine	缓存服务引擎	String	
mode	缓存服务方式	Byte	1表示集群方式，2表示单主从方式
size	缓存服务内存大小	Integer	单位：GB
port	缓存服务端口号	Integer	
vip	缓存服务IP地址	String	
status	缓存服务当前状态	Byte	对照关系请见缓存服务状态通用信息
createTime	缓存服务创建时间	Date	
netType	缓存服务网络类型	Byte	1表示基础网络，2表示VPC网络
vpcId	虚拟专用网络ID	String	只适用于VPC网络下的缓存服务
vnetId	虚拟专用网路EndpointID，或vpclbID	String	只适用于VPC网络下的缓存服务
usedMemory	缓存服务已使用内存大小	bigDecimal	单位：GB
iamProjectId	项目ID	String	
iamProjectName	项目名称	String	
protocol	实例版本	String	

错误信息

关于所有操作返回的错误信息，请参照通用请求一通用错误信息

请求样例

```
https://memcached.api.ksyun.com/
?Action=DescribeCacheCluster
&Version=2018-06-27
&CacheId=07493f09-1161-4442-80e5-e8e8bb1a2ecd
&Engine=memcached
```

返回样例

```
HttpStatusCode=200
```


```
{
  "RequestId": "f351caa8-e6e7-4ddd-90bd-4d9022016236",
  "Data": {
    "cacheId": "4ea00aeb-2c27-4dad-9fc1-c442ade0ecc6",
    "az": "cn-shanghai-3a",
    "name": "xb-test-backup-restore",
    "securityGroupId": "1165",
    "engine": "redis",
    "mode": 2,
    "size": 1,
    "port": 6379,
    "vip": "172.31.255.206",
    "slaveVip": "172.31.254.107",
    "slaveNum": 2,
    "status": 2,
    "createTime": "2016-11-04 10:46:13",
    "netType": 2,
    "vpcId": "95f66aa8-75a7-4e16-ba1e-db4e2d532ac4",
    "vnetId": "21d0c9f5-228f-4330-a169-5f896453c117",
    "timingSwitch": "On",
    "timezone": "00:00-01:00",
    "usedMemory": 0,
    "subOrderId": "MREDIS2S161104104613150077",
    "productId": "3c801860-92ed-4332-a4b2-226620e3c62a",
```

```
    "billType": 1,
    "orderType": 2,
    "orderUse": 1,
    "serviceBeginTime": "2016-11-04 10:46:07",
    "serviceEndTime": "2016-12-04 23:59:59",
    "iamProjectId": "0",
    "iamProjectName": "默认项目",
    "protocol": "redis 2.8"
  }
}
```


DeleteCacheCluster

功能描述

删除缓存服务，根据缓存服务ID删除缓存服务资源

请求参数

关于所有操作需要的通用参数，请参照通用请求—公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：DeleteCacheCluster
Version	API版本号	String	是	取固定值：2018-06-27
CacheId	缓存服务ID	String	是	
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

DELETE

返回结果

英文名称	中文名称	类型	是否必须
reqId	请求流水号	String	是
code	状态: 0: SUCCESS >0 : FAILED	Integer	是
message	文字描述	String	是
data	返回数据信息	Object	否

错误信息

关于所有操作返回的错误信息，请参照通用请求—通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=DeleteCacheCluster
&Version=2018-06-27
&CacheId=a45949b8-f870-4f77-877e-56ba546e0ce9
&Engine=memcached
```

- **返回样例**


```
{
    "RequestId": "7e6d174d-6977-42fb-8a6b-7104584bb3df",
```

```
    "Data": null
}
```

ResizeCacheCluster

更改缓存服务内存大小，单主从模式的缓存服务支持升配

请求参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	string	是	取固定值：ResizeCacheCluster
Version	API版本号	string	是	取固定值： 2018-06-27
CacheId	缓存服务ID	string	是	
Capacity	更配目标大小，以GB为单位	Integer	是	缓存资源为单主从模式时，可选值为：{1, 2, 4, 8, 16, 32, 64}；
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

POST

请求样例

```
https://memcached.api.ksyun.com/
?Action=ResizeCacheCluster
&Version=2018-06-27
&CacheId=xxxxxxxxxxxx
&Capacity=2
&Engine=memcached
```

返回样例

```
{
    "RequestId": "7e6d174d-6977-42fb-8a6b-7104584bb3df",
    "Data": null
}
```

FlushCacheCluster

功能描述

清空缓存服务，清除该缓存服务下的所有数据

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：FlushCacheCluster
Version	API版本号	String	是	取固定值： 2018-06-27
CacheId	缓存服务ID	String	是	
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

PUT

返回结果

请求的返回信息请参照样例

错误信息

关于所有操作返回的错误信息，请参照通用请求－通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=FlushCacheCluster
&Version=2018-06-27
&CacheId=xxxxxxxxxxxxxxxxx
&Engine=memcached
```

- **返回样例**

```
{
    "RequestId": "cd261ef0-71f2-4d11-b74c-875b6455b",
    "Data": null
}
```

RenameCacheCluster

重命名缓存服务，更改缓存服务的名称

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：RenameCacheCluster
Version	API版本号	String	是	取固定值： 2018-06-27
CacheId	缓存服务ID	String	是	
Name	缓存服务名称	String	是	
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

PUT

错误信息

关于所有操作返回的错误信息，请参照通用请求－通用错误信息

请求样例

```
https://memcached.api.ksyun.com/
?Action=RenameCacheCluster
&Version=2018-06-27
&CacheId=xxxxxxxxxxxxxx
&Name=xxxx
&Engine=memcached
```

返回样例

```
{
    "RequestId": "cd261ef0-71f2-4d11-b74c-875b6455b",
    "Data": null
}
```

UpdatePassword

修改缓存服务实例密码

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值： UpdatePassword
Version	API版本号	String	是	取固定值： 2018-06-27
CacheId	缓存服务ID	String	是	
Password	缓存服务密码	String	是	密码规则：(?=.*[A-Z]+.*)(?=.*[a-z]+.*)(?=.*[\d]+.*)([A-Za-z\d!@#$%^&*()_+=-]{8,30})，不传则设置为没有密码
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

PUT

错误信息

关于所有操作返回的错误信息，请参照通用请求－通用错误信息

请求样例

```
https://memcached.api.ksyun.com/
?Action=UpdatePassword
&Version=2018-06-27
&Engine=memcached
&Password=Test1101
```

返回样例

```
{
   "reqId": "xxxxxxxxxx",
   "code": 0,
   "data": null
}
```

DescribeCacheSecurityRules

功能描述

查询缓存服务安全规则，根据缓存服务ID查询该缓存服务下所有的安全规则以及相关信息

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数	英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：DescribeCacheSecurityRules	
Version	API版本号	String	是	取固定值： 2018-06-27	
CacheId	缓存服务ID	String	是		
Engine	缓存服务引擎	String	是	取固定值：memcached	

请求方式

GET

返回结果

请求的返回信息请参照样例	英文名称	中文名称	类型	备注
securityID	缓存服务安全规则ID	Long	--	
cacheId	缓存服务ID	String	--	
protocal	安全规则协议	String	--	
fromPort	端口	Integer	--	
toPort	端口	Integer	--	
cidr	安全规则具体信息	String	--	

错误信息

关于所有操作返回的错误信息，请参照通用请求一通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=DescribeCacheSecurityRules
&Version=2018-06-27
&CacheId=xxxxxxxxx
&Engine=memcached
```

- **返回样例（HttpStatusCode=200）**

```
{
   "RequestId": "xxxxxxxxxxx",
"Data": [
 {
   "securityRuleId": 21,
   "cacheId": "a96076d8-4b9c-4de9-a211-0e116e4874c3",
   "protocal": "tcp",
   "fromPort": 6379,
   "toPort": 6379,
   "cidr": "192.168.18.17/21"
 },
 {
   "securityRuleId": 22,
   "cacheId": "a96076d8-4b9c-4de9-a211-0e116e4874c3",
   "protocal": "tcp",
   "fromPort": 6379,
   "toPort": 6379,
   "cidr": "192.168.18.17/32"
 },
 {
   "securityRuleId": 23,
   "cacheId": "a96076d8-4b9c-4de9-a211-0e116e4874c3",
   "protocal": "tcp",
   "fromPort": 6379,
   "toPort": 6379,
   "cidr": "192.168.18.15/32"
 }
]
 }
```

DeleteCacheSecurityRule

功能描述

删除缓存服务安全规则，根据缓存服务的ID，删除对应的安全规则

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称			取固定值：DeleteCacheSecurityRule
Version	API版本号			取固定值： 2018-06-27
CacheId	缓存服务ID	String	是	--
SecurityRuleId	安全规则ID	Integer	是	安全组规则id在白名单列表中获取
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

DELETE

返回结果

请求的返回信息请参照样例

错误信息

关于所有操作返回的错误信息，请参照通用请求－通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=DeleteCacheSecurityRule
&Version=2018-06-27
&CacheId=xxxxxxxxx
&SecurityRuleId=SecurityRuleId
```

- **返回样例**

```
{
     "RequestId": "xxxxxxxxxxxxxxxx",
      "Data": null
}
```

SetCacheSecurityRules

功能描述

设置缓存服务安全规则，为缓存服务添加安全规则

请求参数

关于所有操作需要的通用参数，请参照通用请求-公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：SetCacheSecurityRules
Version	API版本号	String	是	取固定值： 2018-06-27
CacheId	缓存服务ID	String	是	

SecurityRules.Cidr.N	安全规则IP地址，IP地址格式请参照样例	String	是	
Engine	缓存服务引擎	String	是	取固定值：memcached

请求方式

PUT

返回结果

请求的返回信息请参照样例

错误信息

关于所有操作返回的错误信息，请参照通用请求－通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=SetCacheSecurityRules
&Version=2018-06-27
&CacheId=CachedClusterId
&Engine=memcached
&SecurityRules.Cidr.1=192.168.18.17/21
&SecurityRules.Cidr.2=192.168.18.18/24
```

- **返回样例**

```
HttpStatusCode=200

{
       "RequestId": "xxxxxxxxxxxxxxx",
       "Data": null
}
```

DescribeRegions

功能描述

查询地域，可根据实例类型查询支持的地域，目前支持主从。返回有权限的地域合集。

请求参数

关于所有操作需要的通用参数，请参照通用请求—公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：DescribeRegions
Version	API版本号	String	是	取固定值：2016-07-01
engine	缓存服务引擎	String	是	取固定值：memcached
mode	实例类型	Byte	是	2表示主从

请求方式

GET

返回结果

请求的返回信息请参照样例

错误信息

关于所有操作返回的错误信息，请参照通用请求—通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=DescribeRegions
&Version=2018-06-27
&engine=memcached
&mode=2
```

- **返回样例**

```
{
    "RegionSet": [
        {
            "RegionName": "华东1（上海）",
            "Region": "cn-shanghai-2"
        },
        {
            "RegionName": "华北1（北京）",
            "Region": "cn-beijing-6"
        }
    ],
    "RequestId": "111"
}
```

DescribeAvailabilityZones

功能描述

查询可用区，可根据主从或者集群查询支持的可用区。返回有权限的可用区合集。

请求参数

关于所有操作需要的通用参数，请参照通用请求—公共参数

英文名称	中文名称	类型	是否必须	备注
Action	调用接口名称	String	是	取固定值：DescribeAvailabilityZones
Version	API版本号	String	是	取固定值：2016-07-01
engine	缓存服务引擎	String	是	取固定值：memcached
mode	实例类型	Byte	是	2表示主从

请求方式

GET

返回结果

请求的返回信息请参照样例

错误信息

关于所有操作返回的错误信息，请参照通用请求—通用错误信息

样例

- **请求样例**

```
https://memcached.api.ksyun.com/
?Action=DescribeAvailabilityZones
&Version=2018-06-27
&engine=memcached
&mode=2
```

- **返回样例**

```
{
    "AvailabilityZoneSet": [
        {
            "Region": "cn-shanghai-2",
            "AvailabilityZone": "cn-shanghai-2a"
        },
        {
            "Region": "cn-beijing-6",
            "AvailabilityZone": "cn-beijing-6b"
        }
    ],
    "RequestId": "111"
}
```

rules

英文名称	中文名称	类型	备注
securityRuleId	安全组中安全规则ID	String	--
cidr	安全规则具体信息	String	--
fromPort	端口	String	--
toPort	端口	String	--
protocal	安全规则协议	String	--