目录

目录 1
迁移预检查 3
CreatePrecheck 创建预检查 3
请求方法 3
请求参数 3
返回值 3
请求示例 3
返回示例 3
DescribePrecheck 查看预检查结果 3
请求方法 4
请求参数 4
返回值 4
请求示例 4
返回示例 4
迁移任务管理 5
CreateTask 创建任务 5
请求方法 5
请求参数 5
返回值 6
请求示例 6
返回示例 6
DescribeTask 任务查询 6
请求方法 6
请求参数 6
返回值 6
请求示例 7
返回示例 7
OperateTask 操作任务 7
请求方法 7
请求参数 7
返回值 8
请求示例 8
返回示例 8
跨云厂商迁移 8
跨云厂商迁移 8
前提条件 8
迁移限制 8
操作步骤 8
1. 新建DTS数据迁移服务 8
2. 源库及目标信息 8
任务信息 8
源库信息 8
所需信息： 8
数据库连通性检测 8
目标库信息 9
3. 迁移类型及对象 9
选择类型： 9
迁移类型包括结构迁移、全量数据迁移及增量数据迁移。 9
迁移对象： 9
4. 预检查 9

用户创建一致性校验 9
用户创建一致性校验 9
描述 9
请求参数 9
返回样例 9
签名机制 10
Python Demo 10
Java Demo 10
Go Demo 12
公共参数 13

迁移预检查

CreatePrecheck 创建预检查

在正式迁移数据库之前，对源实例和目标实例进行预检查，以便在迁移正式开始前发现可能存在的配置类错误。

注意：目前仅支持在VPC机房创建预检查任务。

请求方法

POST

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	固定值：CreatePrecheck
Version	API版本号	String	是	固定值：2018-01-08

POST 参数 参数格式：JSON

英文名称	中文名称	类型	是否必须	备注
SourceType	来源数据库类型	String	是	公网迁移："Public" 或 专线迁移："SpecialLine" 或 金山云Krds迁移："Krds"
SourceInstanceId	源实例ID	String	否	在 SourceType 为 "Krds" 时为必传参数
SourceRegion	源实例所在机房	String	是	如："cn-shanghai-2" 或 "cn-beijing-6"
SourceHost	源实例地址	String	否	在 SourceType 为 "Public"（通过公网迁移） 或 "SpecialLine"（通过专线迁移） 时为必传参数
SourcePort	源实例端口	String	否	在 SourceType 为 "Public" 或 "SpecialLine" 时为必传参数
SourceUsername	源实例用户名	String	是	-
SourcePassword	源实例密码	String	是	-
TargetInstanceId	目标实例ID	String	是	-
TargetRegion	目标实例所在机房	String	是	
region	调用接口名称	String	是	必须与源实例所在机房(SourceRegion)一致
DbSchema	待迁移库表结构	String	是	目前仅支持整库迁移 固定值 "{"is_full":true}"
SpecialLineId	专线通道ID	String	是	在 SourceType 为 "SpecialLine" 时为必传参数

返回值

英文名称	中文名称	类型	备注
PrecheckId	预检查ID	String	-

请求示例

```
http://dts.cn-shanghai-2.api.ksyun.com/?Action=CreatePrecheck&Version=2018-01-08

BODY:
{"SourceType":"Krds","SourceUsername":"admin","SourcePassword":"Aa123456","SourceRegion":"cn-shanghai-2","SourceInstanceId":"b1cb52f0-db94-4379-b3b3-5e6c6ae77dd6","TargetInstanceId":"ce2e67fe-89f5-47fb-96f2-f190f1380314","TargetRegion":"cn-shanghai-2","DbSchema":"{\"is_full\":true}","region":"cn-shanghai-2"}
```

返回示例

```
{"Data":{"PrecheckId":"5aeae8ca-beda-4644-8f6e-ee3b21eb4be3"},"RequestId":"f02cd65a-86d4-439c-9dc1-185a3972ed98"}
```

DescribePrecheck 查看预检查结果

查看预检查结果。

请求方法

GET

请求参数

英文名称	中文名称	类型	是否必填	备注
Action	调用接口名称	String	是	取固定值：DescribePrecheck
Version	API版本号	String	是	取固定值：2018-01-08
PrecheckId	预检查ID	String	是	-

返回值

英文名称	中文名称	类型	备注
Id	预检查子ID	String	-
PrecheckId	预检查ID	String	-
Name	预检查子项名称	String	-
Status	预检查状态	String	PASSED: 检查通过 FAILED: 检查失败 ERROR: 检查错误 CHECKING: 检查中
Data	附加信息	String	一般为检查失败的原因

请求示例

http://dts.cn-shanghai-2.api.ksyun.com/?Action=DescribePrecheck&Version=2018-01-08&PrecheckId=bcd408c7-c6ea-4c18-8977-e6dfa0046f68

返回示例


```
{
    "Data": {
        "Progress": 100,
        "SubPrecheck": [
            {
                "Id": "6c2f43ff-5e2f-4a97-9296-291482bda45c",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "srcconn",
                "Status": "PASSED"
            },
            {
                "Id": "faf070a8-2847-403a-8d21-f7033ccdd492",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "dstconn",
                "Status": "PASSED"
            },
            {
                "Id": "0a0f73b0-6f53-4cdc-a860-458299b53eec",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "version",
                "Status": "PASSED"
            },
            {
                "Id": "a1b00e23-4019-4c51-900e-488181074f9b",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "serverid",
                "Status": "PASSED"
            },
            {
                "Id": "34fda4b5-41d6-4fe5-8eae-2a8344421429",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "binlogon",
                "Status": "PASSED"
            },
            {
                "Id": "02ab815a-b4fc-466e-9290-ef3441ce8b31",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "logmode",
                "Status": "PASSED"
            },
            {
                "Id": "47ea07f7-db87-4d0f-8580-6de44acd1b12",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "passwdmode",
                "Status": "PASSED"
```

```
            },
            {
                "Id": "4696ae4c-8490-4682-9cb3-ae3ceaf8bf6b",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "srcauth",
                "Status": "WARNING",
                "Data": {
                    "privlist": "Super_priv"
                }
            },
            {
                "Id": "bed45684-c246-4df9-b38c-18f22f694ad8",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "dstauth",
                "Status": "PASSED"
            },
            {
                "Id": "dcf076c6-4b04-4966-ae80-69d2291b1962",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "srcdup",
                "Status": "PASSED"
            },
            {
                "Id": "96c07f21-6444-4543-9238-c061ff236c4b",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "dstdup",
                "Status": "PASSED"
            },
            {
                "Id": "53ea03ec-34bd-48da-959d-4ae066bb53a9",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "restriction",
                "Status": "PASSED"
            },
            {
                "Id": "ca010f8d-ae93-4edd-a4b2-26655b314b4c",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "engine",
                "Status": "PASSED"
            },
            {
                "Id": "ee1d0a14-e98f-4c8d-aaca-dea8a62c6e1f",
                "PrecheckId": "bcd408c7-c6ea-4c18-8977-e6dfa0046f68",
                "Name": "maxconn",
                "Status": "PASSED"
            }
        ]
    },
    "RequestId": "d655492c-b01f-4d54-b4ff-97c9fd05ae29"
}
```


迁移任务管理

CreateTask 创建任务

创建迁移任务。

注意：当预检查全部通过时，才可以正式创建前一任务。

请求方法

GET

请求参数

英文名称	中文名称	类型	是否必填	备注
Action	调用接口名称	String	是	取固定值：CreateTask
Version	API版本号	String	是	取固定值：2018-01-08
SourceType	数据迁移类型	String	是	为 公网迁移："Public" 或 专线迁移："SpecialLine" 或 RDS到RDS 迁移 "Krds"
TaskName	任务名称	String	是	-

SubTask	迁移类型	String	是	结构迁移：SchemaMigration, 全量迁移：BackupRecovery, 增量迁移：RunReplication，多参数间使用‘，’分隔，后面的迁移类型依赖前面的迁移类型，具体用法可参见下方示例
PrecheckId	预检查ID	String	是	完全成功的预检查ID

返回值

英文名称	中文名称	类型	备注
task_id	任务ID	String	如创建失败则没有任务id

请求示例

```
http://dts.cn-shanghai-2.api.ksyun.com/?Action=CreateTask&Version=2018-01-08&PrecheckId=bcd408c7-c6ea-4c18-8977-e6dfa0046f68&SubTask=SchemaMigration,BackupRecovery,RunReplication&TaskName=test&SourceType=Krds
```

返回示例

```
{
    "Data": {
        "TaskId": "7bd9202b8c75497d"
    },
    "RequestId": "ca20f0b7-8485-4cbf-99bc-67e277fbb8a9"
}
```

DescribeTask 任务查询

查看已经创建好的任务。

请求方法

GET

请求参数

英文名称	中文名称	类型	是否必填	备注
Action	调用接口名称	String	是	取固定值：DescribeTask
Version	API版本号	String	是	取固定值：2018-01-08
TaskId	数据迁移类型	String	否	若不传，则返回任务列表
Marker	记录开始偏移量	Integer	否	若不传，则默认值为0
MaxRecords	每页结果中包含的最大条数	Integer	否	取值范围1-100

返回值

英文名称	中文名称	类型	备注
TaskName	任务名称	String	-
TaskId	任务ID	String	-
TaskStatus	任务状态	String	运行中：RUNNING 运行失败：ERROR 完成：FINISHED 失败：FAILED 暂停：PAUSED 删除：DELETED
Created	创建时间	String	-
ConnConfigId	配置ID	String	忽略
SourceType	迁移类型	String	-
SourceHost	源实例地址	String	-
SourcePort	源实例端口	String	-
TargetInstanceId	目标实例ID	String	
SubTask	子任务任务类型	String	-
SubTasks	子任务列表	String	-
Marker	下次获取记录开始偏移量	Integer	-
MaxRecords	每页结果中包含的最大条数	Integer	-
TotalCount	记录总数	Integer	-

请求示例

```
http://dts.cn-shanghai-2.api.ksyun.com/?Action=DescribeTask&Version=2018-01-08&&TaskId=7bd9202b8c75497d
```

返回示例


```
{
    "Data": {
        "Tasks": [
            {
                "TaskName": "test",
                "TaskId": "7bd9202b8c75497d",
                "TaskStatus": "UNSTARTED",
                "Created": "Oct 15, 2018 7:28:35 PM",
                "ConnConfigId": "7cc8fe9f-ca8d-4129-9683-9e641f22f21d",
                "SourceType": "Krds",
                "SourceHost": "10.102.2.51",
                "SourcePort": 3306,
                "TargetInstanceId": "ce2e67fe-89f5-47fb-96f2-f190f1380314",
                "SubTask": "SchemaMigration,BackupRecovery,RunReplication",
                "SubTasks": [
                    {
                        "Id": "68adea68-f877-4c46-9c3d-65e3573a47b8",
                        "TaskId": "7bd9202b8c75497d",
                        "ConnConfigId": "7cc8fe9f-ca8d-4129-9683-9e641f22f21d",
                        "Name": "RunReplication",
                        "Status": "UNSTARTED",
                        "Judging": 0,
                        "Region": "cn-shanghai-2",
                        "Created": "Oct 15, 2018 7:28:35 PM",
                        "Updated": "Oct 15, 2018 7:28:35 PM",
                        "Deleted": 0,
                        "Progress": 0,
                        "AccountId": "73403574"
                    },
                    {
                        "Id": "6b0a63ed-df35-4232-aeb8-a0c025e177f6",
                        "TaskId": "7bd9202b8c75497d",
                        "ConnConfigId": "7cc8fe9f-ca8d-4129-9683-9e641f22f21d",
                        "Name": "BackupRecovery",
                        "Status": "UNSTARTED",
                        "Judging": 0,
                        "Region": "cn-shanghai-2",
                        "Created": "Oct 15, 2018 7:28:35 PM",
                        "Updated": "Oct 15, 2018 7:28:35 PM",
                        "Deleted": 0,
                        "Progress": 0,
                        "AccountId": "73403574"
                    },
                    {
                        "Id": "73a2b2be-dc8d-4d38-8e3d-ad531601f1c8",
                        "TaskId": "7bd9202b8c75497d",
                        "ConnConfigId": "7cc8fe9f-ca8d-4129-9683-9e641f22f21d",
                        "Name": "SchemaMigration",
                        "Status": "UNSTARTED",
                        "Judging": 0,
                        "Region": "cn-shanghai-2",
                        "Created": "Oct 15, 2018 7:28:35 PM",
                        "Updated": "Oct 15, 2018 7:28:35 PM",
                        "Deleted": 0,
                        "Progress": 0,
                        "AccountId": "73403574"
                    }
                ]
            }
        ]
    },
    "RequestId": "21cdd228-3a3d-40b8-8308-073a612a2a98"
}
```


OperateTask 操作任务

实现对任务状态的管理，包含：开始，暂停，停止，删除操作。

请求方法

GET

请求参数

英文名称	中文名称	类型	是否必填	备注
Action	调用接口名称	String	是	取固定值：DescribeTask
Version	API版本号	String	是	取固定值：2018-01-08
TaskId	数据迁移类型	String	是	-
ActionName	任务操作	String	是	开始：Start 暂停：Pause 停止：Stop 删除：Delete

返回值

若无错误，则无返回值。

请求示例

```
http://dts.cn-shanghai-2.api.ksyun.com/?Action=OperateTask&Version=2018-01-08&&TaskId=7bd9202b8c75497d&&ActionName=Start
```

返回示例

```
{
    "RequestId": "6fc99195-7cd6-45d2-ad38-41967e4a5ff8"
}
```

跨云厂商迁移

跨云厂商迁移

前提条件

- 迁移的源数据库必须支持公网连接。
- 必须拥有金山云 KRDS for MySQL 实例。

迁移限制

- 结构迁移不支持 event 的迁移。
- 增量迁移源库的 MySQL 实例需要开启 binlog。
- 增量迁移源库的 binlog_format 要设置为为 row。

操作步骤

1. 新建DTS数据迁移服务

登录控制台https://console.ksyun.com/#/home/Operation 在左侧导航栏中点击**数据迁移**。



2. 源库及目标信息

输入任务名称、源库和目标实例的信息。信息详情：

任务信息

任务名称： 任务的指定名称（默认会为每个任务自动生成一个名称）

源库信息

实例类型选择有公网IP的自建MySQL数据库

所需信息：

- 实例外网地址
- 实例端口
- 数据库账号
- 数据库密码

数据库连通性检测

成功：连接成功。（表示校验完全通过）

失败：接口错误、连接失败。（表示校验不通过，无法进行迁移。如果校验失败，请根据出错的校验项，检查并修改迁移任务信息，然后重试校验。） 失败原因：如未将DTS服务的外网ip添加至源实例白名单或防火墙中等原因。

目标库信息

- 实例地域（目前支持华北1区、华东1区、华南1区、中国香港）
- 实例名称





在校验通过后，您可以点击**授权白名单**并进入下一步开始迁移数据。

3. 迁移类型及对象

选择类型：

迁移类型包括结构迁移、全量数据迁移及增量数据迁移。

- 如果要做全量迁移，那么选择结构迁移＋全量数据迁移。（全量数据迁移的时间耗时会较长，请您耐心等待）
- 如果要做不停机迁移，那么选择结构迁移＋全量数据迁移＋增量数据迁移。

注：当选择增量迁移时，源MySQL实例需要开启binlog。 当选择增量迁移时，源库的binlog_format 要为row。 当选择增量迁移且源库如果为5.6及以上版本时，它的binlog_row_image必须为full。

迁移对象：

迁移对象可以选择整个实例，也可以选择部分库表。



4. 预检查

在迁移任务正式启动之前，会先进行前置预检查，只有预检查通过后，才能成功启动迁移。 如果预检查失败，那么可以点击具体检查项后的按钮，查看具体的失败详情，并根据失败原因修复后，重新进行预检查。







用户创建一致性校验

用户创建一致性校验

描述

通过这个接口，用户可以创建一个一致性检查的任务。这需要用户账户对来源实例拥有super权限。 创建一致性任务后，可以从任务列表接口查看任务信息。

请求参数

名称	描述	类型	是否必须	备注
Action	接口名称	String	是	固定值：CreateConsistencyCheck
TaskId	任务Id	String	是	例如：c2fda5fc154c4b9e

返回样例

```
{
"RequestId": "3b0cb36f-54a4-4c4e-b930-90b69f62f82e"
```

```
}
```

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))
}
```

公共参数

支持GET和POST两种HTTP方法。

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

数据传输服务(DTS)的服务名称(Service)指定为dts

详情请参考：公共参数