

ALIBABA CLOUD

阿里云

专有云企业版

DataWorks
开发指南

产品版本：V3.12.0

文档版本：20211125

 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 <code>cd /d C:/window</code> 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	<code>bae log list --instanceid</code> <code>Instance_ID</code>
[] 或者 [a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ } 或者 {a b}	表示必选项，至多选择一个。	<code>switch {active stand}</code>

目录

1.产品简介	09
2.基本术语	10
3.API概览	11
4.调用方式	13
4.1. Java SDK	13
4.2. HTTP API	16
4.2.1. 服务地址	17
4.2.2. 通信协议	17
4.2.3. 请求方法	17
4.2.4. 字符编码	17
4.2.5. 接口鉴权	17
4.2.6. 返回值	18
4.2.7. 调用示例	18
5.HTTP API	20
5.1. 新建节点	20
5.2. 更新节点	20
5.3. 批量更新节点	21
5.4. 删除节点	23
5.5. 向上查询父节点	24
5.6. 向下查询子节点	25
5.7. 查询节点的代码	25
5.8. 查询节点	26
5.9. 根据ID查询节点	28
5.10. 统计节点总数	29
5.11. 查询任务实例	30
5.12. 查询任务实例运行日志	32

5.13. 根据任务ID查询实例	33
5.14. 按层数查询父节点实例	34
5.15. 按层数查询子节点实例	35
5.16. 查询统计应用下某个状态的实例数量	36
5.17. 查询统计指定业务日期的任务实例的状态数量	36
5.18. 查询多个应用不同类型任务实例的状态分布	37
5.19. 重跑实例	39
5.20. 恢复实例	40
5.21. 设置实例成功唤醒下游实例	40
5.22. 终止任务	41
5.23. 重跑下游实例	42
5.24. 更新实例调度配置	42
5.25. 暂停实例	43
5.26. 批量终止实例	44
5.27. 批量重跑实例	44
5.28. 批量恢复任务	45
5.29. 批量暂停实例	46
5.30. 批量设置成功	46
5.31. 统计当日实例总数	47
5.32. 查询业务流程实例	48
5.33. 执行补数据操作	48
5.34. 执行冒烟测试	49
5.35. 执行手动业务流程	50
5.36. 终止业务流程	51
5.37. 提交ZIP包	52
5.38. 查询ZIP包提交状态	52
5.39. 数据结构	53
5.39.1. NodeModifyDto	53

5.39.2. TaskEntity	54
5.39.3. DagEntity	56
5.39.4. TaskStatus	57
5.39.5. DagStatus	58
5.39.6. ProgramType	58
5.39.7. CycleType	58
5.39.8. DependencyType	58
6.Java SDK	59
6.1. GetInstanceSummary	59
6.2. AddResGroupGateWay	60
6.3. CreateConnection	61
6.4. CreateDag	62
6.5. CreatedQCEntity	64
6.6. CreatedQCFollowler	65
6.7. CreatedQCRule	66
6.8. CreateManualDag	67
6.9. CreateMetaSpiderJob	68
6.10. CreateResGroup	69
6.11. DeleteConnections	71
6.12. DeleteDQCEntity	72
6.13. DeleteDQCFollowler	72
6.14. DeleteDQCRule	73
6.15. DeleteProjectResGroup	74
6.16. DeleteResGroupGateWay	75
6.17. GetDagDetail	76
6.18. GetDataServiceApiDetail	77
6.19. GetDataServiceAppDetail	78
6.20. GetDefaultTenant	79

6.21. GetDQCEntity	79
6.22. GetDQCfollower	82
6.23. GetDQCRule	82
6.24. GetNodeDetail	86
6.25. GetNodeUpdateStatus	91
6.26. GetResGroupAk	93
6.27. GetResGroupGatewayList	94
6.28. GetResGroupList	95
6.29. GetTableColumn	98
6.30. GetTableList	100
6.31. GetTaskLog	101
6.32. GetUserInfo	102
6.33. ListConnection	103
6.34. ListDataServiceApps	106
6.35. ListDataServiceAuthedApi	107
6.36. ListPermission	108
6.37. ListProjectModule	108
6.38. ListProjectModules	109
6.39. ListProject	109
6.40. ListTenantModule	110
6.41. ListUserPermission	110
6.42. ModifyBusiness	111
6.43. ModifyNode	113
6.44. ModifySolution	115
6.45. RerunTask	116
6.46. ResumeTask	117
6.47. SearchBusiness	118
6.48. SearchManualDagNodeInstance	120

6.49. SearchSolution	120
6.50. SearchTasks	122
6.51. TestConnectivity	124
6.52. UpdateDQCfollower	126
6.53. UpdateDQCRule	127
6.54. UpdateResGroupGateWay	130
6.55. CheckMetaTable	131
6.56. GetMetaDB	132
6.57. GetMetaTable	133
6.58. GetMetaTableIntroWiki	134
6.59. ListMetaColumnLineage	135
6.60. ListMetaTableChangeLog	136
6.61. ListMetaTableColumn	137
6.62. ListMetaTableImpact	138
6.63. ListMetaTableIntroWiki	139
6.64. ListMetaTableLineage	140
6.65. ListMetaTableOutput	141
6.66. ListMetaTablePartition	142
6.67. SearchMetaTables	143
6.68. GetStorageAndCalcMetrics	144
6.69. GetInstanceStatistic	145
6.70. GetInstanceLog	146
6.71. GetNodeCode	147
6.72. QueryInstances	148
7.获取AccessKey	151

1. 产品简介

DataWorks起源于2009年，是一站式大数据智能研发与治理平台，一个从工作室、工坊、车间到工具集等都齐备的大数据工场，帮助企业快速高效地构建数据中台，进行数据分析，实现数据转型。

2.基本术语

项目空间（Project）

项目空间是阿里云大数据集成服务平台最基本的组织对象，是您管理表（Table）、资源（Resource）、自定义函数（UDF）、节点（Node）、权限等的基本单元。

节点

节点是指通过数据开发界面提交发布或者调度API新建接口创建的调度定义信息。

业务流程

业务流程包含若干节点及其节点相互之间的依赖关系。

实例/任务实例

节点需要通过DataWorks调度系统转换成任务实例才能运行。

3.API概览

调度系统API

- 新建节点
- 更新节点
- 批量更新节点
- 删除节点
- 向上查询父节点
- 向下查询子节点
- 查询节点的代码
- 查询节点
- 根据ID查询节点
- 统计节点总数
- 查询任务实例
- 查询实例运行日志
- 根据任务ID查询实例
- 按层数查询父节点实例
- 按层数查询子节点实例
- 查询统计应用下任务实例的状态数量
- 查询统计指定业务日期的任务实例的状态数量
- 查询多个应用不同类型任务实例的状态分布
- 重跑实例
- 恢复实例
- 设置实例成功唤醒下游实例
- 终止任务
- 重跑下游实例
- 更新实例调度配置
- 暂停实例
- 批量终止实例
- 批量重跑实例
- 批量恢复任务
- 批量暂停实例
- 批量设置成功
- 统计当日实例总数
- 查询业务流程实例
- 执行补数据操作
- 执行冒烟测试
- 执行手动业务流程
- 终止业务流程
- 提交ZIP包

- 查询ZIP包提交状态

4. 调用方式

DataWorks提供Java SDK和HTTP两种类型的调用接口方式。Java SDK是一种新的接口调用方式，后续版本新的接口均以Java SDK提供。HTTP调用方式保持不变，不会增加新的接口，后续版本将逐步下线，推荐您使用Java SDK。

4.1. Java SDK

如果Client需要调用Java SDK，需要指明如下初始参数：服务地址、AK、产品名、RegionId和Endpoint名称。

服务地址Endpoint

通常服务地址的规则为字符串dataworks+专有云的部署域名。假设专有云的dataworks IDE访问地址为http://ide.envxxx.yyyyy.com。则服务地址为dataworks.envxxx.yyyyy.com。如果实际情况和此情况有所差别，请联系阿里云技术支持。

AccessID和AccessKey

专有云用户的阿里云账号的AccessID和AccessKey。

产品名

产品名为固定的字符串：dataworks-private-cloud。

RegionId和Endpoint名称

- RegionId为固定值default。
- endpointName为固定值default。

调用授权

调用Java SDK前，您需要为调用SDK的阿里云账号授权，只有经过授权的阿里云账号才能正常调用到Java SDK的接口服务。

接口均开启了白名单的访问控制，即默认没有加到白名单中的云账号均无法访问，没有在白名单中（即没有被授权）的云账号访问以上接口会报如下错误：

```
com.aliyuncs.exceptions.ClientException: InvalidApi.NotFound : Specified api is not found, please check your url and method.
```

出现上述异常，说明当前云账号没有调用此接口的权限，需要按照如下方法进行授权。

您可以将需要授权的云账号和要给该账号授权调用的接口告诉DataWorks的技术支持，根据现场环境的信息生成加密的license：

1. 登录天基。

详情请参见专有云 [运维指南](#) 中ASO操作指南 > 运维工具 > 天基平台运维下的 [登录天基监控系统](#) 章节。

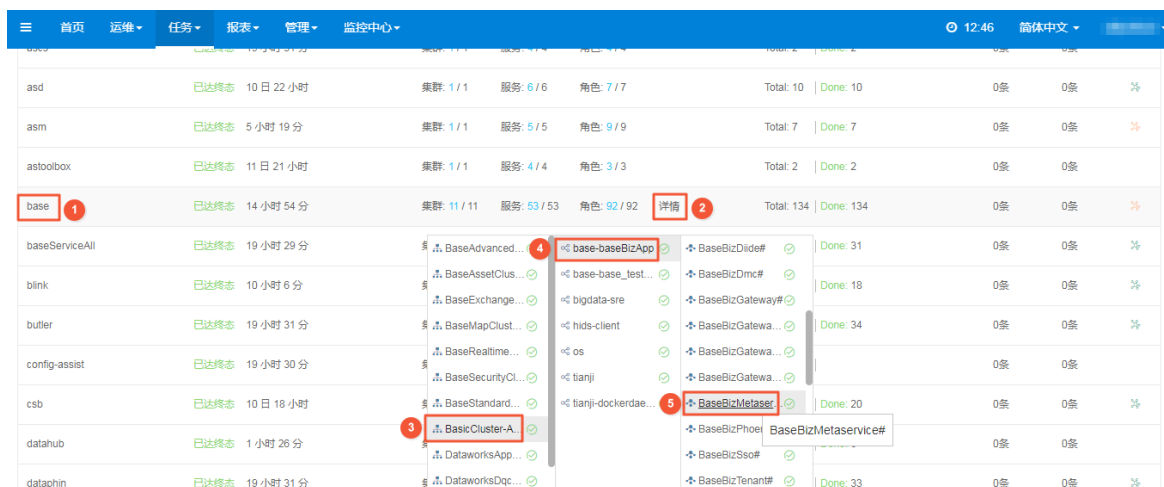
2. 在左侧导航栏，单击**监控**。
3. 鼠标悬停至**任务**，单击**部署概况**。



4. 单击部署详情，进入部署详情页面。



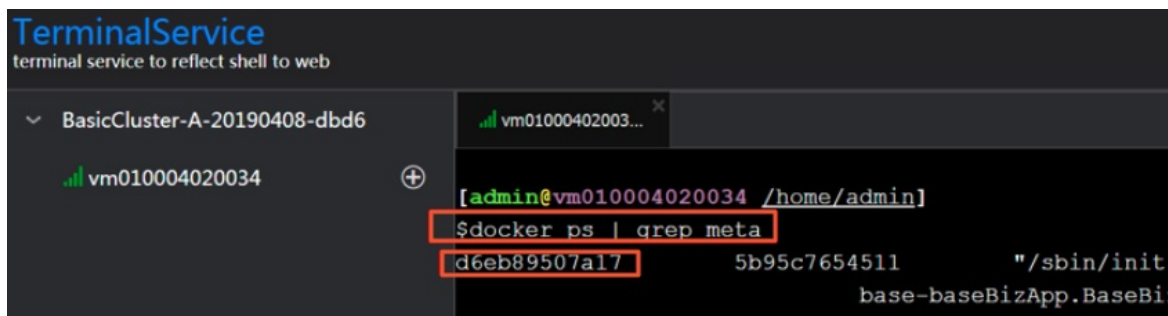
5. 鼠标悬浮至base产品，单击详情，选择BasicCluster-A-20190708-376d Machine List > base-baseBizApp > BaseBizMetaservice。



6. 单击相应机器后的Terminal，登录机器。



7. 执行 `docker ps | grep meta`，找到base-biz-metaservice容器的ID，如下图中示例的是d6eb89507a17。



8. 登录容器docker exec -it d6eb89507a17 bash后，即可应用该license进行授权。



应用的方式为执行如下命令：

```
curl -X POST -d "license=4AA481CE03EAF6529ECA131ED6E29AAD511B66A760C33786AD241BE1D90D7007C4056FA26FD13318B92BCFB4B254E42" "http://127.0.0.1/dataos/v1/conf_dataos_api";
```

其中license=后面的参数是DataWorks技术支持提供的license值。如果授权成功，该命令便会返回 {"data": "ok", "errCode": 0, "errMsg": "正常", "requestId": "0a04142615555747936747847e031b"}，此时调用对应API便不会报没有权限的错误。

示例代码

```
public void setup () throws IOException, ClientException {
    popEndpoint = "dataworks. envxxx.yyyyy.com";
    popEndpointName = "default";
    accessId = "accessId";
    accessKey = "accessKey";
    regionId = "default";
    popProduct = "dataworks-private-cloud";
    X509TrustAll.ignoreSSLCertificate();
    DefaultProfile.addEndpoint(popEndpointName, regionId, popProduct, popEndpoint);
    IClientProfile profile = DefaultProfile.getProfile(regionId, accessId, accessKey);
    IAcsClient client = new DefaultAcsClient(profile);
}
```

```

/** * 示例: DemoApiRequest *
 * @throws ServerException *
 * @throws ClientException */
@Test
public void test_DemoApi() throws ServerException, ClientException {
    DemoApiRequest request = new DemoApiRequest();
    request.setKeyword("abc");
    DemoApiResponse response = client.getAcsResponse(request);
    System.out.println(gson.toJson(response));
}

```

生成Java SDK

Java SDK调用依赖的SDK生成方式如下所示：

在现场的浏览器访问`sdngen.env4b.shuguang.com`（此处的域名仅为示例，实际的域名规则为`sdngen.现场域名`），页面打开内容如下图所示。

产品	最后生成时间	Java SDK	PHP SDK	Python SDK	Go SDK	NET SDK

配置	说明
选择产品	选择dataworks-private-cloud。
选择版本	选择2019-01-17。
选择语言	选择需要的语言。

单击**生成SDK**，然后直接下载SDK即可使用。


说明 每次DataWorks在专有云中重新部署，所有API的权限都会清空，需要驻场人员根据license重新进行授权。

调度系统的调用账号accountId/accountPwd/clientName

涉及到调度系统的API，需要填写accountId、accountPwd和clientName公共参数，获取方法如下：

1. 找现场运维人员登录base-biz-tsp应用的数据库（库名为dwtsps）。
2. 往cloudapi_account表中插入一条记录，accountId/accountPwd/clientName即为对应的调用账号。

4.2. HTTP API

 **警告** DataWorks HTTP API已停止维护，出于兼容性考虑会暂时保留，将在后续版本逐步下线。请您尽快升级至Java SDK。

4.2.1. 服务地址

API接口按照功能划分成了不同的功能模块，每个模块使用不同的域名访问，具体域名请参见各个接口的文档。

4.2.2. 通信协议

API的所有接口在专有云通过HTTP进行通信。

4.2.3. 请求方法

您可以根据相应API文档中接口指定的方式进行请求。

- 如果使用GET方式，均从Query String获取参数，即根据Query String中的查询参数（Query parameter）返回结果。
- 如果使用POST方式，则从Query String获取参数，Request Body中的参数将被忽略。

4.2.4. 字符编码

均使用UTF-8编码。

4.2.5. 接口鉴权

Header

base_id（必填）

tenant_id（必填）

鉴权

调用API的接口需要经过签名，调用方需要进行以下操作：

1. 在url中增加2个固定字段：baseKey和timestamp。
2. 生成签名，加到http header中，名为signature。

- 以GET方法为例。

假设用户B提供了http接口为/aetapi，需要的query parameter分别为name和age。用户A原有的调用url应该是 /getapi?name=fengyeng&age=20，会接入后端服务模块（token中心）进行Appcode一类的认证，您可以自行选择认证方式，控制其他人访问API。此处在此处增加了两个参数，所以新的url应该是 /getapi?name=fengyeng&age=20?baseKey=dsa134×tamp=123456789，随后利用新的url中的query parameter结合token生成signature，调用方再在header中增加一个变量signature:dsafadfsa141fdsaf1。

- 以POST方法为例。

假设用户B提供了http接口为/postapi，A原有的调用url应该是/postapi，接入token中心后，在query parameter中增加了两个参数，所以新的url应该是/postapi?baseKey=dsa134×tamp=123456789，随后利用新的url中的query parameter结合token生成signature，调用方再在header中增加一个变量signature:dsafadfsa141fdsaf1。

4.2.6. 返回值

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": <具体结果数据>
}
```

4.2.7. 调用示例

代码示例：

```
import org.apache.commons.lang3.StringUtils; import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient; import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpUriRequest; import org.apache.http.impl.client.HttpClientBuilder;
import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; import java.net.URLDecoder;
import java.security.MessageDigest; import java.util.Arrays;
public class SearchNodes {
    public static void main(String[] args) throws Exception {
        HttpClient httpClient = HttpClientBuilder.create().build();
        String uri = "http://baseapi.***.com/v1.0/node/prod?executeMethod=SEARCH&searchText=nodeTest";
        HttpUriRequest request = createHttpRequest(
            "base key",
            "base token", uri
        );
        request.setHeader("tenant_id", "租户 id"); request.setHeader("base_id", "用户 id");
        HttpResponse response = httpClient.execute(request); BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(response.getEntity().getContent())); StringBuilder stringBuilder = new StringBuilder();
        while(bufferedReader.ready()) {
            String line = bufferedReader.readLine(); if (line == null) {
                break;
            }
            stringBuilder.append(line);
        }
        System.out.println(stringBuilder.toString());
    }
    private static String byteArrayToHexString(byte[] byteArray) {
        final char[] HEX_DIGITS = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};
        String result = null;
        int l = byteArray.length; char[] str = new char[l << 1]; int k = 0;
        for (int i = 0; i < l; i++) {
            byte byte0 = byteArray[i];
            str[k++] = HEX_DIGITS[byte0 >>> 4 & 0xf];
            str[k++] = HEX_DIGITS[byte0 & 0xf];
        }
        result = new String(str); return result;
    }
}
```

```
result = new String(str); return result;
}
public static String tokenSign(String param, String salt) throws Exception {
    if (StringUtils.isEmpty(param) || StringUtils.isEmpty(salt)) {
        return null;
    }
    String query = URLDecoder.decode(param, "UTF-8");
    String[] pStr = query.trim().split("&");
    Arrays.sort(pStr);
    StringBuilder buf = new StringBuilder(); for (int i = 0; i < pStr.length; ++i) {
        buf.append(pStr[i]);
        buf.append("&");
    }
    String paramStr = buf.substring(0, buf.length() - 1);
    String sourceStr = salt + ":" + paramStr + ":" + salt;
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[] byteArray = md.digest(sourceStr.getBytes("UTF-8"));
    return byteArrayToHexString(byteArray);
}
protected static HttpRequest createHttpRequest(String baseKey, String baseToken, String uri) {
    StringBuilder buf;
    int index = uri.indexOf('?');
    if (index == -1) {
        index = uri.length();
        buf = new StringBuilder(uri.length() + baseKey.length() + 13);
        buf.append(uri).append('?');
    } else {
        buf = new StringBuilder(uri.length() + baseKey.length() + 13);
        buf.append(uri).append('&');
    }
    buf.append("baseKey=").append(baseKey);
    buf.append("&timestamp=").append(System.currentTimeMillis());
    HttpRequest request = new HttpGet(buf.toString()); String signature;
    try {
        signature = tokenSign(buf.substring(index + 1), baseToken);
    } catch (Exception ex) {
        throw new RuntimeException("error on sign for uri: " + uri, ex);
    }
    request.addHeader("signature", signature);
    return request;
}
}
```

5.HTTP API

5.1. 新建节点

URL: /v1.0/node/{projectEnv}/sync

Method: POST

Path Parameters:

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": CREATE_NODE_SYNC, # 操作类型
  # 更多详细字段参见本文档的数据结构部分, NodeModifyDto
  "nodeType": <INTEGER>, # 节点类型, 0: 正常任务 1: 一次性任务 2: 暂停 3: 空跑节点
  "dependentType": <INTEGER>, 0: 不跨版本 1: 一层子节点 2: 本节点 3: 自定义
  "nodeName": <STRING>, # 自己指定
  "prgType": <INTEGER>, # 节点脚本类型
  "priority": <INTEGER>, # 优先级
  "owner": <STRING>, # 用户
  "appId": <LONG>, # 项目ID
  "cycType": <INTEGER>, # 调度类型
  "codeSrc": <STRING>, # 节点代码
  "nodeFrom": <STRING>, # 节点来源
  "inputs": [{ # 节点输入
    "data": <STRING>, # 上游节点的输出
  }],
  "outputs": [{ # 节点输出
    "data": <STRING>, # 本节点的输出
  }],
  "uniqueId": <STRING>, # 自定义个唯一ID
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志, 排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": <LONG>, # Node ID
}
```

5.2. 更新节点

URL: /v1.0/node/{projectEnv}/sync

Method: POST

Path Parameters:

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": CREATE_NODE_SYNC, # 操作类型
  # 更多详细字段参见本文档的数据结构部分, NodeModifyDto
  "nodeType": <INTEGER>, # 节点类型, 0: 正常任务 1: 一次性任务 2: 暂停 3: 空跑节点
  "dependentType": <INTEGER>, 0: 不跨版本 1: 自定义 2: 一层子节点 3: 自己
  "nodeName": <STRING>, # 自己指定
  "prgType": <INTEGER>, # 节点脚本类型
  "priority": <INTEGER>, # 优先级
  "owner": <STRING>, # 用户
  "appId": <LONG>, # 项目ID
  "cycType": <INTEGER>, # 调度类型
  "codeSrc": <STRING>, # 节点代码
  "nodeFrom": <STRING>, # 节点来源
  "inputs": [{ # 节点输入
    "data": <STRING>, # 上游节点的输出
  }],
  "outputs": [{ # 节点输出
    "data": <STRING>, # 本节点的输出
  }],
  "uniqueId": <STRING>, # 自定义个唯一ID
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志, 排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true,
}
```

5.3. 批量更新节点

URL: /v1.0/node/{projectEnv}

Method: POST

Path Parameters:

参数名	类型	说明
-----	----	----

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
[
  {
    "executeMethod": "CREATE_NODE", # CREATE_NODE,UPDATE_NODE,DELETE_NODE
    "nodeId": <LONG>, # 更新时需要, 调度系统节点ID
    "uniqueId": <LONG>, # 必选, 客户端传过来的标识ID,在云端并不使用,客户端主要用于对照返回结果
    "nodeName": <STRING>, # 必选, 节点定义名称
    "nodeType": <INTEGER>, # 必选, 节点类型, 0: 正常任务 1: 一次性任务 2: 暂停 3: 空跑节点
    "prgType": <INTEGER>, # 必选, 执行代码类型, 如odps_shell、odps_sql等
    "description": <STRING>, # 可选, 描述信息
    "paraValue": <STRING>, # 可选, 参数信息
    "priority": <INTEGER>, # 可选, 优先级,默认为0
    "cronExpress": <STRING>, # 必选, cron表达式
    "cloudUUID": <LONG>, # 新建没有, 更新必选
    "fileId": <LONG>, # 可选, 代码文件ID
    "fileVersion": <LONG>, # 可选, 代码版本号
    "owner": <STRING>, # 必选, 负责人, 统一使用工号
    "resGroupId": <LONG>, # 必选, 所属资源组ID
    "appId": <LONG>, # 必选, 所属应用ID
    "baseLineId": <LONG>, # 所属基线ID
    "createUser": <STRING>, # 必选, 创建人
    "modifyUser": <STRING>, # 必选, 最新修改人
    "multiInstCheckType": <INTEGER>, # 可选, 多实例检测标识
    "cycType": <INTEGER>, # 必选, 0: 周期大于等于天 1: 周期小于天 2: 周期小于天并且实例按顺序执行 (1,2,3....)
    "dependentType": <INTEGER>, ## 必选, 0: 不跨版本 1: 一层子节点 2: 本节点 3: 自定义
    "dependentNodeIds": [<LONG>, ...], # 默认为依赖一层子节点
    "multiInstKillType": <INTEGER>, # 可选, 0: 不做 1: kill非日常 2: kill日常
    "refreshToTask": <BOOLEAN>, # 可选, 是否在当前实例生效
    "updateNodeOnly": <BOOLEAN>, # 是否刷新上下游节点元数据, 默认false
    "codeSrc": <STRING>, # updateNodeOnly为false则必须, 代码
    "envType": <INTEGER>, # 可选, 运行环境类型
    "nodeFrom": <STRING>, # 可选, 节点的来源,从哪个系统发布过来的
    "inputs": [
      {
        "data": <STRING>, # 必选, 节点输入的数据
      },
      ...
    ], # 必须, 节点输入
    "outputs": [
      {
        "data": <STRING>, # 必选, 节点输出的数据
      },
      ...
    ], # 必须, 节点输出
    "relatedFlowId": <LONG>, # 可选
    "startEffectDate": <STRING>, # 可选, 允许调度的起始日期
    "endEffectDate": <STRING>, # 可选, 允许调度的终止日期
    "rerunAble": <BOOLEAN>, # 必须, 设置是否可重跑
  }
]
```

```
"userExtension": <STRING>, # 可选, 扩展属性
"childNodeId": <LONG>, # 可选, 添加&删除节点依赖关系时使用的参数
"parentNodeId": <LONG>, # 可选
"createInstanceUrl": <STRING>, # 可选,
"dqcType": <INTEGER>, # dqc校验类型, 可选
"dqcDescription": <STRING>, # dqc描述 可选
"taskRerunTime": <INTEGER>, # 可选,
"taskRerunInterval": <LONG>, # 可选
"manualTrigger": <INTEGER>, # 可选
"syncRefreshTask": <BOOLEAN>, # 可选
"flowId": <LONG> # 必选
},
...
]
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": <STRING> # requestId
}
```

5.4. 删除节点

URL: /v1.0/node/{projectEnv}/sync

Method: POST

Path Parameters:

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": DELETE_NODE_SYNC, # 操作类型
  "nodeId": <LONG>, # 节点ID
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": {
    "节点ID": "success",
    ...
  }
}
```

5.5. 向上查询父节点

功能：按条件分层查询指定Node的父节点。

URL：/v1.0/node/{projectEnv}

Method：GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	SEARCH_NODE_PARENTS_BY_DEPTH, 必选
nodeId	Long	节点的ID, 必选
depth	Integer	遍历的层数, 必选
nodeIds	String	节点ID列表, 以逗号分隔, 可选
cloudUUID	Long	代码库对应的节点ID, 可选

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [<NodeEntity>, ...]
}
```

5.6. 向下查询子节点

功能：按条件分层查询指定Node的子节点。

URL：/v1.0/node/{projectEnv}

Method：GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	SEARCH_NODE_CHILDREN_BY_DEPTH, 必选
nodeId	Long	节点的ID, 必选
depth	Integer	遍历的层数, 必选
nodeIds	String	节点id列表, 以逗号分隔, 可选
cloudUUID	Long	代码库时相关的节点的ID, 可选

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [<NodeEntity>, ...]
}
```

5.7. 查询节点的代码

URL：/v1.0/node/{projectEnv}

Method：GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	shuoming
executeMethod	String	SEARCH_NODE_CODE
cloudUUID	Long	代码库时相关的节点的ID
version	Long	节点代码的版本号

Response

```
{
  "requestId": <STRING>, # 请求的ID, 用来定位日志, 排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": <STRING> # 代码内容
}
```

5.8. 查询节点

URL: /v1.0/node/{projectEnv}

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	shuoming
executeMethod	String	查询类型为SEARCH
nodeName	String	节点名, 可选
cloudUUID	Long	代码库时相关的节点的ID, 可选
nodeId	Long	节点的ID, 可选
version	Long	节点代码的版本号, 可选
inputs	String	节点的输入, 以逗号分隔, 可选
outputs	String	节点的输出, 以逗号分隔, 可选

参数名	类型	shuoming
fileId	Long	文件ID, 可选
nodeIds	String	节点ID列表, 以逗号分隔, 可选
searchText	String	数据开发界面搜索的字符, 包括搜索节点ID和节点名字, 支持后模糊, 可选
prgType	String	程序类型, 可选
prgTypes	Integer	查询是多个类型时, 以逗号隔开, 可选
appId	Long	应用ID 如果为空, 在我的所有应用范围内搜索, 可选
owner	String	节点负责人, 可选
createUser	String	节点创建人, 可选
modifyUser	String	节点修改人, 可选
createTime	String	创建时间yyyy-MM-dd HH:mm:ss, 可选
modifyTime	String	修改时间 yyyy-MM-dd HH:mm:ss, 可选
baseLineId	Long	基线ID, 可选
deployDate	String	发布时间, 可选
orderBy	String	可选
isDetail	Boolean	是否查询节点的详细信息, 可选
depth	Integer	遍历的层数, 可选
flowId	Long	工作流ID
nodeType	Integer	0, 正常调度任务被日常调度 1, 手动任务不会被日常调度 2, 暂停任务被日常调度, 但启动调度时直接被置为失败 3, 空跑任务被日常调度, 但启动调度时直接被置为成功 可选
tenantId	Long	租户ID, 可选
bizDate	String	业务业务时间格式[yyyy-MM-dd], 可选
prgName	String	程序类型名称, 可选
isOnline	Integer	是否在线 1: 在线 0: 下线, 可选

参数名	类型	说明
modifyStartTime	String	修改起始时间，可选
modifyEndTime	String	修改截止时间，可选
deployStartingDate	String	发布起始时间，在指定一段时间内查询节点，可选
deployDeadlineDate	String	发布截止时间，在指定一段时间内查询节点，可选
nodeForm	String	节点的来源
rerunAble	Boolean	设置是否可重跑，可选
bizId	Long	业务流程ID，可选
solId	Long	解决方案ID，可选
filePaths	String	文件路径，可选
isSmoke	Boolean	可选
resGroupId	Long	资源组ID，可选
isDeploy	Boolean	可选
deployType	Integer	发布节点的类型，0: all 1: new（新增节点） 2: update（修改节点） 3: delete（删除节点），可选
pageStart	Integer	
pageSize	Integer	

Response

```
{
  "requestId": <STRING>, # 请求的ID，用来定位日志，排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [
    <NodeEntity>,
    ...
  ]
}
```

5.9. 根据ID查询节点

URL: /v1.0/node/{projectEnv}

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	SEARCH_NODE_BY_ID
nodeId	Long	
isDetail	Boolean	是否查询节点的详细信息

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": <NodeEntity>
}
```

5.10. 统计节点总数

URL: /v1.0/node/{projectEnv}

Method: GET

Headers: application/json; charset=UTF-8

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	SEARCH_NODE_COUNT

Response

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": null,
  "returnValue": 283232, # 节点数
  "success": true
}
```

5.11. 查询任务实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	环境标识，DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为SEARCH。
appId	Long	任务实例所属应用ID。
appIds	String	多值使用逗号分隔。
taskId	Long	任务实例ID。
nodeId	Long	任务实例对应的节点定义ID。
inGroupId	Long	实例的周期号。
nodeIds	String	节点ID列表，使用逗号分隔。
dagId	Long	任务实例所属的流程实例ID。
taskType	Integer	任务类型 <ul style="list-style-type: none">0：正常1：一次性任务2：暂停的节点实例3：空跑
taskTypes	String	任务类型，多值使用逗号分隔。

参数名	类型	说明
taskStatuses	String	任务状态，多值使用逗号分隔。
dagType	Integer	DAG类型 <ul style="list-style-type: none">1: routing2: smoke3: completeData4: onceTime
prgType	Integer	执行代码类型。
status	Integer	任务状态。
owner	String	负责人。
bizdate	String	业务日期。
odpsInstanceId	String	MaxCompute任务实例ID。
bizBeginHour	String	任务开始时间。
bizEndHour	String	任务结束时间。
beginBizDate	String	任务开始业务日期。
endBizDate	String	任务结束业务日期。
searchText	String	前端搜索的字符，包括搜索节点ID和节点名称。
baseLineId	Long	所属基线ID。
createTime	String	接受提交请求的时间。
isDetail	Boolean	是否查询节点的详细信息，可选。
opSeq	Long	操作序列号。
depth	Integer	遍历的层数，可选。
historyId	Long	查询任务日记的参数。
costTime	Integer	耗时，单位为秒。
startRunTimeFrom	String	分时间段查询开始运行的时间。
startRunTimeTo	String	分时间段查询结束运行的时间。
finishRunTimeFrom	String	分时间段查询的开始时间。
finishRunTimeTo	String	分时间段查询的结束时间。

参数名	类型	说明
createTimeFrom	String	开始创建的时间。
createTimeTo	String	结束创建的时间。
prgTypes	String	查询多个程序类型时，以逗号分隔。
taskIds	String	任务ID列表。
rerunAble	Boolean	设置是否可重跑，可选。
dagName	String	业务流程名称。
orderBy	String	可选。
bizId	Long	业务流程ID。
solId	Long	解决方案ID。
gmtdate	String	处理日期。
createUser	String	创建人。
modifyUser	String	最新修改人。
modifyTime	String	最新修改时间。
deployDate	String	发布时间。
pageStart	Integer	开始分页。
pageSize	Integer	分页大小。

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [
    <TaskEntity>,
    ...
  ]
}
```

5.12. 查询任务实例运行日志

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为SEARCH_TASK_RUNLOG。
taskId	Long	和historyId指定其一即可。
historyId	Long	和taskId指定其一即可。

Response

```
{
  "requestId": <STRING>, # 请求的ID, 用来定位日志, 排查问题。
  "returnCode": "0", # 0表示调用成功。
  "returnMessage": <STRING>, # 返回执行的详细信息。
  "successCode": "0",
  "returnValue": <STRING>
}
```

5.13. 根据任务ID查询实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	环境标识, DEV/PROD

URL parameters

参数名	类型	说明
executeMethod	String	查询类型为SEARCH_TASK_BY_ID。
taskId	Long	任务ID。

参数名	类型	说明
isDetail	Boolean	是否获取APP详情。

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题。
  "returnCode": "0", # 0表示调用成功。
  "returnMessage": <STRING>, # 返回执行的详细信息。
  "successCode": "0",
  "returnValue": <TaskEntity>
}
```

5.14. 按层数查询父节点实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为SEARCH_TASK_PARENTS_BY_DEPTH
taskId	Long	任务实例所属的流程实例ID
depth	Integer	节点层次

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [
    <TaskEntity>,
    ...
  ]
}
```

5.15. 按层数查询子节点实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	环境标识, DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为SEARCH_TASK_CHILDREN_BY_DEPTH。
taskId	Long	任务实例所属的流程实例ID。
depth	Integer	节点层次。

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题。
  "returnCode": "0", # 0表示调用成功。
  "returnMessage": <STRING>, # 返回执行的详细信息。
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [
    <TaskEntity>,
    ...
  ]
}
```

5.16. 查询统计应用下某个状态的实例数量

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为 SEARCH_TASK_STATISTICS
appld	Long	
bizdate	String	

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": {
    "dagType": <INTEGER>,
    "appld": <LONG>,
    "bizdate": <STRING>, # 业务日期
    "noRun": <INTEGER>,
    "running": <INTEGER>,
    "waittingResource": <INTEGER>,
    "failure": <INTEGER>,
    "success": <INTEGER>,
    "waittingTime": <INTEGER>
  }
}
```

5.17. 查询统计指定业务日期的任务实例的状态数量

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为 SEARCH_TASK_STATISTICS_BY_OWNER
bizdate	String	

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": {
    "dagType": <INTEGER>,
    "appId": <LONG>,
    "bizdate": <STRING>, # 业务日期
    "noRun": <INTEGER>,
    "running": <INTEGER>,
    "waittingResource": <INTEGER>,
    "failure": <INTEGER>,
    "success": <INTEGER>,
    "waittingTime": <INTEGER>
  }
}
```

5.18. 查询多个应用不同类型任务实例的状态分布

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	环境标识，DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	查询类型为 SEARCH_TASK_SUCCESSINFO。
bizdate	String	业务日期。
costTime	Integer	耗时，单位为秒。
startRunTimeFrom	String	任务执行开始的时间。
startRunTimeTo	String	任务执行完成的时间。
finishRunTimeFrom	String	分时间段查询开始的时间。
finishRunTimeTo	String	分时间段查询完成的时间。
createTimeFrom	String	开始创建的时间。
createTimeTo	String	结束创建的时间。
appld	Long	应用ID。
searchText	String	前端搜索的字符，包括搜索节点ID和节点名字，支持后模糊。
nodeId	Long	节点ID。
nodeIds	String	节点ID列表。
taskId	String	任务实例所属的流程实例ID。
taskStatuses	String	任务状态。
taskTypes	String	任务类型。
prgTypes	String	接受提交请求的时间。
createTime	String	
modifyTime	String	最新修改的时间。
deployDate	String	最新发布日期。
gmtdate	String	处理日期。
bizBeginHour	String	开始时间。

参数名	类型	说明
bizEndHour	String	结束时间。
bizBeginDate	String	开始的业务日期。
bizEndDate	String	结束的业务日期。

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": {
    "totalCount": <INTEGER>,
    "notRunCount": <INTEGER>,
    "waitTimeCount": <INTEGER>,
    "waitResCount": <INTEGER>,
    "runningCount": <INTEGER>,
    "failureCount": <INTEGER>,
    "successCount": <INTEGER>
  }
}
```

5.19. 重跑实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "RERUN_TASK",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.20. 恢复实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "RESUM_TASK",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.21. 设置实例成功唤醒下游实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "SETSUCCESS_TASK",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.22. 终止任务

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "KILL_TASK",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.23. 重跑下游实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "FIXDATA",
  "taskId": <LONG>,
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.24. 更新实例调度配置

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "REFRESH_NODE",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.25. 暂停实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "STOP_TASK",
  "taskId": <LONG>
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.26. 批量终止实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "BATCH_KILL_TASK",
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.27. 批量重跑实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "BATCH_RERUN_TASK",
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.28. 批量恢复任务

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "BATCH_RESUM_TASK",
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.29. 批量暂停实例

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "BATCH_STOP_TASK",
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.30. 批量设置成功

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "BATCH_SETSUCCESS_TASK",
  "includeTaskIds": [<LONG>, ...]
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.31. 统计当日实例总数

URL: /v1.0/task/{projectEnv}

Headers: application/json; charset=UTF-8

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Url Parameters

参数名	类型	说明
executeMethod	String	SEARCH_TASK_STATISTICS_BY_DATE
dagType	Integer	可选，默认1
bizdate	String	可选，默认当天日期的前一天，格式yyyy-MM-dd

Response

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": null,
  "returnValue": 283232, # 实例数
  "success": true
}
```

5.32. 查询业务流程实例

URL: /v1.0/dag/{project Env}

Method: GET

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

URL Parameters

参数名	类型	说明
executeMethod	String	值为SEARCH_DAG_BY_ID
dagId	Long	任务实例所属的流程实例ID

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": [<DagEntity>, ...]
}
```

5.33. 执行补数据操作

URL: /v1.0/dag/{project Env}

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "CREATE_DAG",
  "name": <STRING>, # 自己定一个，补数据工作流名称
  "type": 3, # 流程实例类型, 3为补数据
  "rootNodeId": <LONG>, # 补数据的起始，根节点ID，如果只补一个节点，则该节点设置为rootNodeId
  "includeNodeIds": [<LONG>, ...], # 包含的节点ID列表，若只补一个节点，该节点也得在includeNodeIds中
  "excludeNodeIds": [<LONG>, ...], # 排除不补数据的节点ID列表
  "startBizDate": <STRING>, # 补数据起始业务日期 yyyy-MM-dd hh:mm:ss
  "endBizDate": <STRING>, # 补数据结束业务日期 yyyy-MM-dd hh:mm:ss
  "bizBeginTime": <STRING>, # 小时任务的开始时间 13:04:04
  "bizEndTime": <STRING>, # 小时任务的结束时间 14:04:04
  "isParallel": <BOOLEAN>, # 补数据的DAG是否可以同时并行运行
  "nodeParas": {<LONG>: <STRING>, ...}, # 可选，自定义刷新参数,Key节点ID,Value参数实际值
  "dagPara": <STRING>, # 可选，设置Dag参数，此参数会同步到本次Dag中所有的实例中
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": <LONG>, # dag id
}
```

5.34. 执行冒烟测试

URL: /v1.0/dag/{projectEnv}

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "CREATE_DAG",
  "name": <STRING>, # 自定义一个工作流名称
  "type": 2, # 流程实例类型, 2为冒烟
  "bizdate": <STRING>, # 业务日期 yyyy-MM-dd hh:mm:ss
  "rootNodeId": <LONG>, # 根节点ID
  "rootNodeAppId": <LONG>, # 根节点所属应用ID
  "includeNodeIds": [], # 留空
  "excludeNodeIds": [], # 留空
  "bizBeginTime": <STRING>, # 小时任务的开始时间 13:04:04
  "bizEndTime": <STRING>, # 小时任务的结束时间 14:04:04
  "isParallel": false, # 执行冒烟测试的实例是否可以并行运行
  "nodeParas": {<LONG>: <STRING>, ...}, # 可选, 自定义刷新参数, Key节点ID, Value参数实际值
  "dagPara": <STRING>, # 可选, 设置Dag参数, 此参数会同步到本次Dag中所有的实例中
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": <LONG>, # dag id
}
```

5.35. 执行手动业务流程

URL: /v1.0/dag/{project Env}

Method: POST

Path Parameters

参数名	类型	说明
project Env	String	DEV/PROD

Body

```
{
  "executeMethod": "CREATE_BUS_PROC_DAG",
  "name": <STRING>, # 手动业务流程实例名称, 自己定一个
  "type": 5, # 手动业务流程实例类型为5
  "flowId": <LONG>, # 手动业务流程ID
  "bizdate": <STRING>, # 业务日期, 格式yyyy-MM-dd hh:mm:ss
  "rootNodeAppId": <LONG>, # 所属应用ID
  "includeNodeIds": [],
  "excludeNodeIds": [],
  "isParallel": false,
  "nodeParas": {<LONG>: <STRING>, ...}, # 可选, 自定义刷新参数, Key节点ID, Value参数实际值
  "dagPara": <STRING>, # 可选, 设置Dag参数, 此参数会同步到本次Dag中所有的实例中
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID, 用来定位日志, 排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "count": <INTEGER>,
  "returnValue": <LONG>, # dag id
}
```

5.36. 终止业务流程

URL: /v1.0/dag/{projectEnv}

Method: POST

Path Parameters

参数名	类型	说明
projectEnv	String	DEV/PROD

Body

```
{
  "executeMethod": "KILL_DAGS",
  "dagIds": [<LONG>, ...], # 批量终止的DagIDs
}
```

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": <BOOLEAN>
}
```

5.37. 提交ZIP包

功能：提交发布，提交ZIP包，包中有流程和节点定义、资源，返回RequestID。

URL：/submission/project/{projectId}

Method：POST

Headers：Content-type: multipart/form-data

Form Data：file:zip文件

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": true
}
```

5.38. 查询ZIP包提交状态

URL：/submission/project/{projectId}/{requestId}

Path Parameters

参数名	类型	说明
requestId	String	

Response

```
{
  "requestId": <STRING>, # 请求的ID,用来定位日志,排查问题
  "returnCode": "0", # 0表示调用成功
  "returnMessage": <STRING>, # 返回执行的详细信息
  "successCode": "0",
  "returnValue": {
    "createTime": <STRING>, # 接受提交请求时间
    "name": <STRING>, # 生成的名称标识,格式为[调用方的系统名-backToCheckId-createTime]
    "status": <STRING>, # success: 执行成功, fail:执行失败,running: 正在执行
    "message": <STRING>
  }
}
```

5.39. 数据结构

5.39.1. NodeModifyDto

```
{
  "nodeId": <LONG>, # 更新时需要, 调度系统节点ID
  "uniqueId": <LONG>, # 必选, 客户端传过来的标识ID,在云端并不使用,客户端主要用于对照返回结果
  "nodeName": <STRING>, # 必选, 节点定义名称
  "nodeType": <INTEGER>, # 必选, 节点类型, 0: 正常任务 1:一次性任务 2: 暂停 3:空跑节点
  "prgType": <INTEGER>, # 必选, 执行代码类型, 如odps_shell、odps_sql等
  "description": <STRING>, # 可选, 描述信息
  "paraValue": <STRING>, # 可选, 参数信息
  "priority": <INTEGER>, # 可选, 优先级,默认为0
  "cronExpress": <STRING>, # 必选, cron表达式
  "cloudUUID": <LONG>, # 新建没有, 更新必选
  "fileId": <LONG>, # 可选, 代码文件ID
  "fileVersion": <LONG>, # 可选, 代码版本号
  "owner": <STRING>, # 必选, 负责人
  "resGroupId": <LONG>, # 必选, 所属资源组ID
  "appId": <LONG>, # 必选, 所属应用ID
  "baseLineId": <LONG>, # ?, 所属基线ID
  "createUser": <STRING>, # 必选, 创建人
  "modifyUser": <STRING>, # 必选, 最新修改人
  "cycType": <INTEGER>, # 必选, 0: 周期大于等于天 1: 周期小于天 2: 周期小于天并且实例按顺序执行 (1,2,3....)
  "dependentType": <INTEGER>, # 必选, 0: 不跨版本 1: 自定义 2: 一层子节点 3: 自己
  "dependentNodeIds": [<LONG>, ...], # dependentType为3的必选, 自定义跨版本依赖的节点, 逗号分隔
  "multiInstKillType": <INTEGER>, # 可选, 0:不做 1: kill非日常 2:kill日常
  "refreshToTask": <BOOLEAN>, # 可选, 是否在当前实例生效
  "updateNodeOnly": <BOOLEAN>, # 是否刷新上下游节点元数据, 默认false
  "codeSrc": <STRING>, # updateNodeOnly为false则必须, 代码
  "envType": <INTEGER>, # 可选, 运行环境类型
  "nodeFrom": <STRING>, # 可选, 节点的来源,从哪个系统发布过来的
  "inputs": [
    {
      "data": <STRING>, # 必选, 节点输入输出
    },
    ...
  ], # 必须, 输入
  "outputs": [
```

```

    {
      "data": <STRING>, # 必选, 节点输入输出
    }
    , ...
  ], # 必须, 输出
  "relatedFlowId": <LONG>, # 可选
  "startEffectDate": <STRING>, # 可选, 允许调度的起始日期
  "endEffectDate": <STRING>, # 可选, 允许调度的终止日期
  "rerunAble": <BOOLEAN>, # 必须, 设置是否可重跑
  "userExtension": <STRING>, # 可选, 扩展属性
  "childNodeId": <LONG>, # 可选, 添加&删除节点依赖关系时使用的参数
  "parentNodeId": <LONG>, # 可选
  "createInstanceUrl": <STRING>, # 可选,
  "dqctype": <INTEGER>, # dqc校验类型, 可选
  "dqcdescription": <STRING>, # dqc描述 可选
  "taskRerunTime": <INTEGER>, # 可选,
  "taskRerunInterval": <LONG>, # 可选
  "manualTrigger": <INTEGER>, # 可选
  "syncRefreshTask": <BOOLEAN>, # 可选
  "flowId": <LONG> # 必选
}

```

5.39.2. TaskEntity

```

private Long id; //历史任务自增ID
private Long taskId; // 任务实例ID
private Long nodeId; // 任务实例对应的节点定义ID
private Long dagId; // 任务实例所属的流程实例ID
private Integer inGroupId; // 该周期是当天的第几个周期, 小时任务
private Integer taskType; // 节点类型: 0: 正常 1: 一次性任务 2:表示暂停的节点实例 3: 空跑
private Integer dagType; // DAG类型: 1: routing 2: smoke 3: completeData 4:onceTime
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date dueTime; // 任务定时时间
private Integer status; // 任务状态
private Long opSeq; // 操作序列号
private Integer opCode; // 操作命令
private String owner; // 负责人
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date bizdate; // 业务日期
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date gmtdate; // 处理日期
private String gateway; // 任务运行的gateway机器
private String gwProcessId; // 任务运行的gateway进程号
private String gwLogFile; // 任务运行的gateway日志
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date alisaReturnTime; // alisa的回调序列号
private String prgName; // 执行代码路径
private Integer prgType; // 执行代码类型, 如odps_sql等
private Integer priority; // 优先级

```

```
private Integer weight; // 权重
private String execName; // 执行脚本路径
private Long fileId; // 代码文件ID
private Integer fileVersion; // 代码文件版本号
private String odpsProjectName; // 项目名称
private String paraValue; // 参数信息
private String gwLogLocalFile; // 任务运行的gateway本地日志
private String resGroupIdentifier; // 资源组识别符
private Long resGroupId; // 任务实例所属资源组ID
private Long baseLineId; // 任务实例所属基线ID
private Long appId; // 任务实例所属应用ID
private String appName; // 应用名
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date beginWaitTimeTime; // 变成等待时间状态的时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date beginWaitResTime; // 变成等待资源状态的时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date beginRunningTime; // 变成运行中状态的时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date finishTime; // 任务结束时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date createTime; // 创建时间
private String createUser; // 创建人
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date modifyTime; // 最新修改时间
private String modifyUser; // 最新修改人
private Integer multiInstCheckType; // 多实例检测标识
private Integer multiKillType; // 多实例kill
private Integer cycType; // 周期类型
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date cycTime; // 周期时间
private Integer roleType; // 节点角色类型，如叶节点、非叶节点
private Integer dependentType; // 依赖关系类型
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date deployDate; // 发布时间
private String nodeName; // 节点名称
private Integer rerunTimes; // 重试次数
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date nodeModifyTime; // 节点的修改时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date refreshTime;
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
@JsonDeserialize(using = DateDeserializer.class)
private Date delayExecTime;
```

```
private Boolean rerunAble;  
private List<TaskRelationDto> parentTaskRelations; // 父关系  
private List<TaskRelationDto> childTaskRelations; // 子关系  
private Integer isRunOver; // 表示实例是否运行过  
private Long bizId;  
private List<MetaTable> parentOutputTabMetas; // 依赖表的Meta产出信息  
private List<MetaTable> outputTabMetas; // 本实例输出表的Meta产出信息
```

5.39.3. DagEntity


```

private String name; // 流程实例名称
private Integer type; // 流程实例类型
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date bizdate; // 业务日期 一次性任务传该值
private Long rootNodeId; // 根节点定义ID
private List<Long> includeNodeIds = new ArrayList<Long>(); // 包含的节点定义列表
private List<Long> excludeNodeIds = new ArrayList<Long>(); // 未包含的节点定义列表
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date startBizDate; // 补数据起始业务日期
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date endBizDate; // 补数据结束业务日期
private String bizBeginTime; // 小时任务的开始时间 13:04:04
private String bizEndTime; // 小时任务的结束时间 14:04:04
private Boolean isParallel; // 补数据的DAG是否可以同时并行运行
// //////////////////////////////////分割线：以上创建临时工作流所需属性////////////////////////////////////
private Long dagId; // 流程实例ID
private Integer status; // 流程实例状态
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date gmtdate; // 处理日期
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date startDate; // DAG实例的启动日期，用于查询
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date startTime; // 启动时间
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date finishTime; // 结束时间
private String rootTaskIds; // 根节点实例ID
private Long appId; // 所属应用ID
private Long parentDagId; // 父流程实例ID
private Long childDagId; // 子流程实例ID
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date createTime; // 创建时间
private String createUser; // 创建人
@JsonSerialize(using = DateSerializer.class, include = Inclusion.NON_NULL)
    @JsonDeserialize(using = DateDeserializer.class)
private Date modifyTime; // 最新修改时间
private String modifyUser; // 最新修改人
private Integer createStatus; // 1:initiated 2: creating 3:succcess 4:failure
private Long opSeq;
private Integer dagNum;
private Boolean isDependDaily = false; // 是否依赖开发环境的实例

```

5.39.4. TaskStatus

- NOT_RUN(1, "未运行"),

- WAIT_TIME(2, "等待时间"),
- WAIT_RESOURCE(3, "等待资源"),
- RUNNING(4, "运行中"),
- FAILURE(5, "运行失败"),
- SUCCESS(6, "运行成功");

5.39.5. DagStatus

- CREATED(1, "已创建"),
- RUNNING(4, "运行中"),
- FAILURE(5, "运行失败"),
- SUCCESS(6, "运行成功");

5.39.6. ProgramType

- IDE_SHELL(6),
- ODPS_SQL(10),
- ODPS_MR(11),
- POL_SYNC(23) 同步任务

5.39.7. CycleType

- DAY(0), // 天级别周期，包括天、周、月等任务，每天生成一个实例
- NOT_DAY(1), // 非天级别周期，包括小时、分钟等任务，每天生成多个实例
- NOT_DAY_SEQ(2); // 非天级别周期序列号，按序排列

5.39.8. DependencyType

- NONE(0), // 不跨版本
- USER_DEFINE(1), // 跨版本，自定义依赖
- CHILD(2), // 跨版本，一级子节点依赖
- SELF(3); // 自依赖

6.Java SDK

6.1. GetInstanceSummary

根据Aliuid获取reader、writer的实例运行统计信息。

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetInstanceSummary。
Aliuid	String	是	主账号Aliuid。
EndTime	String	是	结束时间满足yyyy-mm-dd格式。
StartTime	String	是	开始时间满足yyyy-mm-dd格式。
Type	String	是	type=1表示reader，type=2表示writer。

返回参数

参数名称	类型	描述
Data	Long	返回的统计信息列表
ErrCode	Long	错误码
ErrMsg	String	错误信息
RequestId	String	请求ID

请求示例

```
http(s)://[Endpoint]/?Action=GetInstanceSummary
&Aliuid=178*****2537
&EndTime=2019-02-21
&StartTime=2019-02-21
&Type=1
&<公共请求参数>
```

返回示例

```
{
  "Data": [
    {
      "InstanceCnt": 17,
      "AliyunKp": "1782****62537",
      "PluginName": "mysql", "TotalRecords": 4255624,
      "TotalBytes": 266409932
    }
  ],
  "ErrMsg": "success",
  "RequestId": "e19f8502-76d2-4cb1-ab54-b93013cfad17", "ErrCode": 0
}
```

6.2. AddResGroupGateWay

在资源组中增加gateway。

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值AddResGroupGateWay。
Baseld	String	是	云账号ID。
Identifier	String	是	资源组identifier。
Memory	Integer	是	内存大小。
NodeAddress	String	是	gateway IP地址。
NodeName	String	是	gateway机器名。
ProjectId	Long	是	工作空间ID。
TenantId	Long	是	租户ID。
VCpu	Integer	是	gateway CPU数量。

返回参数

参数名称	类型	描述
Data	Boolean	是否成功
ErrCode	Long	错误码
ErrMsg	String	错误信息

参数名称	类型	描述
RequestId	String	请求ID

请求示例

```
http(s)://[Endpoint]/?Action=AddResGroupGateWay
&BaseId=1486***73474
&Identifier=3907f9a58*****6b2506
&Memory=3
&NodeAddress=12.12.12.12
&NodeName=hostname
&ProjectId=75059
&TenantId=2678***8690
&VCpu=8
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc1748b*****493e5f15",
  "data": true,
  "errMsg": "success",
  "errCode": 0
}
```

6.3. CreateConnection

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateConnection。
Connection	String	是	数据源连接串信息。
Name	String	是	数据源名称。
ProjectId	Long	是	工作空间ID。
Type	String	是	数据源类型。
EnvType	Integer	否	环境标识，0：开发，1：生产。
TenantId	Long	否	租户ID。
Shared	Boolean	否	是否共享。

参数名称	类型	是否必选	说明
Description	String	否	描述信息。
SubType	String	否	目前仅支持MySQL类型。

返回参数

参数名称	类型	描述
Data	Long	如果创建成功返回数据源ID，如果创建失败，返回null。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=CreateConnection
&<公共请求参数>
```

6.4. CreateDag

请求参数

请求参数	类型	是否必选	说明
Action	String	是	系统规定参数，取值：CreateDag。
AccountId	String	是	调度系统分配的调用账户。
AccountPwd	String	是	调度系统分配的调用密码。
ClientName	String	是	调度系统分配的调用客户端名称。
BaseId	String	是	发起调用的云账号ID。

请求参数	类型	是否必选	说明
DagModifyDto	String	是	<pre>{ "executeMethod": "CREATE_DAG", "name": "test_create_dag", "type": 3, "includeNodeIds": [3231], "excludeNodeIds": [], "rootNodeId": 3231, "startBizDate": "2018-07-08 00:00:00", "endBizDate": "2018-07-09 00:00:00", "isParallel": false }</pre>
ProjectEnv	String	是	环境标识，PROD / DEV。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	异常信息。
ReturnValue	Long	返回值：DagID。

返回示例

```
{
  "returnCode":"0",
  "returnErrorSolution":"",
  "returnMessage":"",
  "requestId":"740ded9e-f6f5-4fde-a419-9b54***e8",
  "returnValue":300000119,
  "success":true
}
```

6.5. CreateDQCEntity

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateDQCEntity。
EnvType	String	是	计算引擎类型。
MatchExpression	String	是	分区表达式名称。
ProjectName	String	是	MaxCompute的项目名称。
TableName	String	是	表名。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码，0是正常，反之异常。
ReturnValue	Integer	返回成功添加的分区表达ID。

请求示例


```
/?Action=CreateDQCEntity
&EntityLevel=0
&EnvType=odps
&MatchExpression=ds=${yyyymmdd-1}
&ProjectName=autotest
&TableName=test_dqc_decimal_1119_2
&<公共请求参数>
```

返回示例


```
{
  "ReturnCode": "0",
  "ReturnValue": 4003923
}
```

6.6. CreateDQCfollower

CreateDQCfollower用于创建分区表达式订阅，包括邮件、短信、钉钉机器人和hook。

 **说明** 钉钉机器人有webhook地址，添加订阅管理后，发送消息给钉钉群通知用户，触发DQC告警。

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateDQCfollower。
ProjectName	String	否	MaxCompute项目名称。
Follower	String	否	订阅实体，邮件/短信为工号钉钉/hook为http服务地址。
AlarmMode	Integer	否	告警模式 1：邮件 2：邮件和短信 3：钉钉群机器人/hook 4：钉钉群机器人@ALL。
EntityId	Long	否	分区表达式ID。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码
ReturnValue	Integer	返回订阅ID。

请求示例

```
/?Action=CreateDQCfollower
&AlarmMode=2
&EntityId=4003922
&Follower=50624
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": 1726,
  "ReturnCode": "0"
}
```

6.7. CreateDQCRule

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值 CreateDQCRule。
EntityId	Integer	是	分区表达式ID。
ProjectName	String	是	MaxCompute项目名称。
Rules	String	是	规则集合。

返回参数

参数名称	类型	说明
ReturnCode	String	返回错误码。
ReturnValue	Boolean	返回是否添加成功。

请求示例

```
/?EntityId=4003922
&ProjectName=autotest
&Rules={
  "addTemplateRules":[{"property":"table_count",
    "propertyType":"table",
    "blockType":"0",
    "templateId":"7",
    "trend":"abs",
    "warningThreshold":10,
    "criticalThreshold":50,
    "ruleType":"0"
  }],
  "addSelfserviceRules":[{"property":"table_count",
    "propertyType":"table",
    "blockType":"0",
    "methodName":"count/table_count",
    "whereCondition":"id>100",
    "checker":"9",
    "trend":"abs",
    "operator":">",
    "expectValue":0,
    "comment":"哈哈",
    "ruleType":"1",
    "warningThreshold":null,
    "criticalThreshold":null}],
  "envType":"odps",
  "projectName":"autotest",
  "entityId":4003922}
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": true,
  "ReturnCode": "0"
}
```

6.8. CreateManualDag

您可在OpenAPI Explorer中进行可视化调试，并自动生成SDK调用示例。

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateManualDag。

参数名称	类型	是否必选	说明
Bizdate	String	是	业务日期，例如，2019-04-03 00:00:00。
FlowName	String	是	手动业务流程名称
ProjectName	String	是	项目空间名称
DagPara	String	否	业务流程参数 {"key1": "val1", "key2": "val2"}
NodePara	String	否	节点参数，Map<Long, String>。Long 为节点id。String为节点参数。 {"1232234255": "key1=val1 key2=val2", "321231412": "key1=val1 key2=val2"}。

返回参数

参数名称	类型	说明
RequestId	String	请求ID。
ReturnCode	String	返回错误码。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	异常信息。
ReturnValue	Long	返回值：dagId。

示例

```
{
  "RequestId": "*****",
  "ReturnCode": 0,
  "ReturnMessage": "",
  "ReturnErrorSolution": "",
  "ReturnValue": 123142124124
}
```

6.9. CreateMetaSpiderJob

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateMetaSpiderJob。
Baseld	String	是	云账号的ID。
DatasourceId	Long	是	数据源ID。
ProjectId	Long	是	工作空间的ID。
SpiderConfig	String	是	配置信息。
TenantId	Long	是	租户ID。
SpiderCron	String	否	定时表达式。

返回参数

参数名称	类型
Data	Boolean
ErrCode	Long
ErrMsg	String
RequestId	String

请求示例

```
http(s)://[Endpoint]/?Action=CreateMetaSpiderJob
&<公共请求参数>
```

6.10. CreateResGroup

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值CreateResGroup。
Baseld	String	是	云账号的ID。
Name	String	是	资源组名称。
ProjectId	Long	是	DataWorks工作空间的ID。

参数名称	类型	是否必选	说明
ProjectIds	JSON	是	DataWorks工作空间的ID列表。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
Data	Object	返回数据。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=CreateResGroup
&BaseId=1486821399873474
&Name=test20190509
&ProjectId=75059
&ProjectIds=["75059"]
&TenantId=267883377178690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0a98***e0b1b",
  "data": {
    "id": 334603176233473,
    "identifier": "e86f***64dde44",
    "name": "test1234",
    "nick": null,
    "type": "DI",
    "mode": "ISOLATE",
    "tenantId": 267883377178690,
    "enableKp": false,
    "cluster": "e86f1***64dde44",
    "quota": null,
    "isDefault": false,
    "sequence": null,
    "action": "create",
    "specs": {
      "resourceGroupTag": "1"
    }
  },
  "errMsg": "success",
  "errCode": 0
}
```

6.11. DeleteConnections

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteConnections。
ConnectionIds	String	是	连接ID列表。

返回参数

参数名称	类型	说明
Data	Boolean	是否删除成功。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=DeleteConnections
&<公共请求参数>
```

6.12. DeleteDQCEntity

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteDQCEntity。
EntityId	Long	是	分区表达式的ID。
EnvType	String	是	计算引擎的类型，包括MaxCompute、hive和streaming。
ProjectName	String	是	MaxCompute项目名称。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码
ReturnValue	Boolean	返回是否成功

请求示例

```
/?Action=DeleteDQCEntity
&EntityId=4003922
&EnvType=odps
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": true,
  "ReturnCode": "0"
}
```

6.13. DeleteDQCFollower

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteDQCfollower。
ProjectName	Long	否	MaxCompute项目名称。
Id	Long	否	订阅ID。

返回参数

参数名称	类型	描述
ReturnCode	String	返回码
ReturnValue	Boolean	是否成功

请求示例

```
/?Action=DeleteDQCfollower
&Id=1724
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": true,
  "ReturnCode": "0"
}
```

6.14. DeleteDQCRule

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteDQCRule。
ProjectName	String	否	MaxCompute项目名称。
Id	Long	否	规则ID。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码

参数名称	类型	说明
ReturnValue	Boolean	是否成功删除指定的规则

请求示例

```
/?Action=DeleteDQCRule
&Id=279580661
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnCode": "0",
  "ReturnValue": true
}
```

6.15. DeleteProjectResGroup

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteProjectResGroup。
BaseId	String	是	云账号的ID。
Identifier	String	是	资源组identitier。
ProjectId	Long	是	工作空间ID。
ResGroupId	Long	是	资源组ID。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
Data	Boolean	是否成功删除指定的资源组。如果已删除，返回true。如果未删除，返回false。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。

参数名称	类型	说明
RequestId	String	请求RequestId。

请求示例

```
http(s)://[Endpoint]/?Action=DeleteProjectResGroup
&BaseId=1486821399873474
&Identifier=e86f1a3ba2ca451eab8851ed564dde44
&ProjectId=75059 &ResGroupId=334603176233473
&TenantId=267883377178690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc1746d15574010397142703e6afb",
  "data": true,
  "errMsg": "success",
  "errCode": 0
}
```

6.16. DeleteResGroupGateWay

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值DeleteResGroupGateWay。
BaseId	String	是	云账号的ID。
Identifier	String	是	资源组identifier。
NodeName	String	是	GateWay名称。
ProjectId	Long	是	工作空间ID。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
Data	Boolean	是否成功删除GateWay
ErrCode	Long	错误码

参数名称	类型	说明
ErrMsg	String	错误信息
RequestId	String	请求requestid

请求示例

```
http(s)://[Endpoint]/?Action=DeleteResGroupGateWay
&BaseId=1486821399873474
&Identifier=3907f9a5832942f9b63b0908476b2506
&NodeName=hostname
&ProjectId=75059
&TenantId=267883377178690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc1748b15574028120678613e5f16",
  "data": true,
  "errMsg": "success",
  "errCode": 0
}
```

6.17. GetDagDetail

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetDagDetail。
AccountId	String	是	调度系统分配的调用账户。
AccountPwd	String	是	调度系统分配的调用密码。
ClientName	String	是	调度系统分配的调用客户端标识。
BaseId	String	是	发起调用的阿里云用户ID。
ProjectEnv	String	是	环境标识，PROD/DEV。
TenantId	Long	是	租户ID。

参数名称	类型	是否必选	说明
DagId	Long	否	工作流实例ID。

返回参数

参数名称	类型	说明
Count	Integer	结果数。
RequestId	String	请求ID。
ReturnCode	String	返回错误码。
ReturnMessage	String	错误信息。
ReturnErrorSolution	String	错误的解决方案。
ReturnValue	Long	返回值。

示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "14***4-e059-4bf4-a2c1-***323f",
  "returnValue": [
    {
      "name": "海量补数据分组测试15",
      "type": 3,
      "rootNodeId": 1,
      "status": 6,
      "flowId": 1
    }
  ],
  "success": true
}
```

6.18. GetDataServiceApiDetail

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值 GetDataServiceApiDetail。
ApiId	Long	是	API的ID。

参数名称	类型	是否必选	说明
Verbose	Long	否	是否显示详细文本。

返回参数

参数名称	类型	说明
Data	Boolean	返回获取到的详细信息。
ErrCode	Integer	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=GetDataServiceApiDetail
&<公共请求参数>
```

6.19. GetDataServiceAppDetail

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值 GetDataServiceAppDetail。
BaseId	String	是	云账号的ID。
ProjectId	Long	是	工作空间ID。

返回参数

参数名称	类型	说明
Data	Boolean	返回获取到的详细信息。
ErrCode	Integer	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=GetDataServiceAppDetail
&<公共请求参数>
```

6.20. GetDefaultTenant

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetDefaultTenant。

返回参数

参数名称	类型	说明
ErrCode	Integer	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。
Tenant	String	租户信息。
TenantCode	String	租户代码。
TenantOwnerBaseId	String	租户Owner的ID。

请求示例

```
http(s)://[Endpoint]/?Action=GetDefaultTenant
&<公共请求参数>
```

6.21. GetDQCEntity

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetDQCEntity。
ProjectName	String	否	DataWorks工作空间名称。
TableName	String	否	表名称。

参数名称	类型	是否必选	说明
EnvType	String	否	计算引擎类型，包括MaxCompute/hive/streaming。
MatchExpression	String	否	分区表达式，可不填写，不填写返回表下的所有分区表达式。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码
ReturnValue	String	返回的分区表达式列表

请求示例

```
http(s)://[Endpoint]/?Action=GetDQCEntity
&EnvType=odps
&MatchExpression=dt=${yyyymmdd-1}
&ProjectName=autotest
&TableName=test_dqc_decimal_1119_2
&<公共请求参数>
```

返回示例


```
{
  "ReturnValue": {
    "Entity": [
      {
        "EntityLevel": 0,
        "ProjectName": "autotest",
        "MatchExpression": "dt=${yyyymmdd-3}",
        "Followers": "050624",
        "OnDuty": "050624",
        "Sql": 0,
        "GmtCreate": "2018-11-26 15:06:32",
        "EnvType": "odps", "Task": 0,
        "HasRelativeNode": false, "Id": 4003918,
        "TableName": "test_dqc_decimal_1119_2",
        "GmtModify": "2018-11-26 15:06:32"
      },
      {
        "EntityLevel": 0,
        "ProjectName": "autotest",
        "MatchExpression": "dt=${yyyymmdd-1}",
        "Followers": "050624",
        "OnDuty": "050624",
        "Sql": 0,
        "GmtCreate": "2018-11-26 22:31:13",
        "EnvType": "odps",
        "Task": 0,
        "HasRelativeNode": false,
        "Id": 4003922,
        "TableName": "test_dqc_decimal_1119_2",
        "GmtModify": "2018-11-26 22:31:13"
      },
      {
        "EntityLevel": 0,
        "ProjectName": "autotest",
        "MatchExpression": "dt=${yyyymmdd-2}",
        "Followers": "050624",
        "OnDuty": "050624",
        "Sql": 0,
        "GmtCreate": "2018-11-26 23:18:34",
        "EnvType": "odps",
        "Task": 0,
        "HasRelativeNode": false,
        "Id": 4003923,
        "TableName": "test_dqc_decimal_1119_2",
        "GmtModify": "2018-11-26 23:18:34"
      }
    ]
  },
  "ReturnCode": "0"
}
```

6.22. GetDQCfollower

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetDQCfollower。
ProjectName	String	否	DataWorks工作空间名称。。
EntityId	Long	否	分区表达式ID。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码。
ReturnValue	String	返回的订阅列表。

请求示例

```
http(s)://[Endpoint]/?Action=GetDQCfollower
&EntityId=4003922
&ProjectName=odps
&<公共请求参数>
```

返回示例

```
{
  "ReturnCode": "0",
  "ReturnValue": {
    "Follower": [
      {
        "ProjectName": "autotest",
        "AlarmMode": 1,
        "Follower": "050624",
        "Id": 1726,
        "TableName": "test_dqc_decimal_1119_2",
        "EntityId": "4003922"
      }
    ]
  }
}
```

6.23. GetDQCRule

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetDQCRule。
ProjectName	String	否	DataWorks工作空间名称。
EntityId	Long	否	分区表达式ID。

返回参数

参数名称	类型	说明
ReturnCode	String	返回码。
ReturnValue	List	返回的规则列表。

请求示例

```
http(s)://[Endpoint]/?Action=GetDQCRule
&EntityId=4003922
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": {
    "TemplateRules": {
      "TemplateRule": [
        {
          "TemplateName": "SQL task table rows, 1, 7, 30 days fluctuation test",
          "ProjectName": "autotest",
          "MatchExpression": "dt=${yyyymmdd-1}",
          "WarningThreshold": "40",
          "RuleCheckerRelationId": 1008005,
          "RuleType": 0,
          "MethodId": 8,
          "CriticalThreshold": "80",
          "FixCheck": false,
          "OnDuty": "050624",
          "Property": "table_count",
          "TemplateId": 7,
          "PropertyKey": "table_count",
          "MethodName": "table_count",
          "HistoryWarningThreshold": "history max:40%,history min:10%",
          "BlockType": 1,
          "CheckerId": 7,
          "TableName": "test_dec_decimal_1110_2"
```

```

    "tablename": "test_dqc_decimal_1119_2",
    "Id": 279580663,
    "EntityId": 4003922,
    "HistoryCriticalThreshold": "history max:80%,history min:50%",
    "Trend": "abs"
  },
  {
    "TemplateName": "SQL task table rows, 1,7, 30 days fluctuation test",
    "ProjectName": "autotest",
    "MatchExpression": "dt=${yyyyymmdd-1}",
    "WarningThreshold": "10",
    "RuleCheckerRelationId": 1008007,
    "RuleType": 0,
    "MethodId": 8,
    "CriticalThreshold": "50",
    "FixCheck": false,
    "OnDuty": "050624",
    "Property": "table_count",
    "TemplateId": 7,
    "PropertyKey": "table_count",
    "MethodName": "table_count",
    "HistoryWarningThreshold": "history max:10%,history min:10%",
    "BlockType": 0,
    "CheckerId": 7,
    "TableName": "test_dqc_decimal_1119_2",
    "Id": 279580665,
    "EntityId": 4003922,
    "HistoryCriticalThreshold": "history max:50%,history min:50%",
    "Trend": "abs"
  },
  {
    "TemplateName": "SQL task table rows, 1,7, 30 days fluctuation test",
    "ProjectName": "autotest",
    "MatchExpression": "dt=${yyyyymmdd-1}",
    "WarningThreshold": "10",
    "RuleCheckerRelationId": 1008009,
    "RuleType": 0,
    "MethodId": 8,
    "CriticalThreshold": "50",
    "FixCheck": false,
    "OnDuty": "050624",
    "Property": "table_count",
    "TemplateId": 7,
    "PropertyKey": "table_count",
    "MethodName": "table_count",
    "HistoryWarningThreshold": "history max:10%,history min:10%",
    "BlockType": 0,
    "CheckerId": 7,
    "TableName": "test_dqc_decimal_1119_2",
    "Id": 279580667,
    "EntityId": 4003922,
    "HistoryCriticalThreshold": "history max:50%,history min:50%",
    "Trend": "abs"
  }
}
1

```

```
,
},
"SelfserviceRules": {
  "SelfserviceRule": [
    {
      "ProjectName": "autotest",
      "MatchExpression": "dt=${yyyymmdd-1}",
      "Checker": 9,
      "RuleCheckerRelationId": 1008004,
      "WhereCondition": "id>100",
      "RuleType": 1,
      "MethodId": 21,
      "FixCheck": true,
      "CheckerName": "compared with a fixed value",
      "OnDuty": "050624",
      "Property": "table_count",
      "PropertyKey": "table_count",
      "MethodName": "count/table_count",
      "BlockType": 0,
      "TableName": "test_dqc_decimal_1119_2",
      "Id": 279580662,
      "CheckerId": 6,
      "EntityId": 4003922,
      "Trend": "abs"
    },
    {
      "ProjectName": "autotest",
      "MatchExpression": "dt=${yyyymmdd-1}",
      "Checker": 9,
      "RuleCheckerRelationId": 1008006,
      "WhereCondition": "id>100",
      "RuleType": 1,
      "MethodId": 21,
      "FixCheck": true,
      "CheckerName": "compared with a fixed value",
      "OnDuty": "050624",
      "Property": "table_count",
      "PropertyKey": "table_count",
      "MethodName": "count/table_count",
      "BlockType": 0,
      "TableName": "test_dqc_decimal_1119_2",
      "Id": 279580664,
      "CheckerId": 6,
      "EntityId": 4003922,
      "Trend": "abs"
    },
    {
      "ProjectName": "autotest",
      "MatchExpression": "dt=${yyyymmdd-1}",
      "Checker": 9,
      "RuleCheckerRelationId": 1008008,
      "WhereCondition": "id>100",
      "RuleType": 1,
      "MethodId": 21,
      "FixCheck": true,
```

```
"CheckerName": "compared with a fixed value",
"OnDuty": "050624",
"Property": "table_count",
"PropertyKey": "table_count",
"MethodName": "count/table_count",
"BlockType": 0,
"TableName": "test_dqc_decimal_1119_2",
"Id": 279580666,
"CheckerId": 6,
"EntityId": 4003922,
"Trend": "abs"
},
{
  "ProjectName": "autotest",
  "MatchExpression": "dt=${yyyyymmdd-1}",
  "Checker": 9,
  "RuleCheckerRelationId": 1008010,
  "WhereCondition": "id>100",
  "RuleType": 1,
  "MethodId": 21,
  "FixCheck": true,
  "CheckerName": "compared with a fixed value",
  "OnDuty": "050624",
  "Property": "table_count",
  "PropertyKey": "table_count",
  "MethodName": "count/table_count",
  "BlockType": 0,
  "TableName": "test_dqc_decimal_1119_2",
  "Id": 279580668,
  "CheckerId": 6,
  "EntityId": 4003922, "Trend": "abs"
}
]
}
},
"ReturnCode": "0"
}
```

6.24. GetNodeDetail

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetNodeDetail。
AccountId	String	是	调度系统分配的调用账户。

参数名称	类型	是否必选	说明
AccountPwd	String	是	调度系统分配的调用密码。
BaseId	String	是	发起调用的阿里云用户ID。
ClientName	String	是	调度系统分配的调用客户端标识。
NodeId	Long	是	节点ID。
ProjectEnv	String	是	环境标识PROD / DEV。
TenantId	Long	是	租户ID。
IsDetail	Boolean	否	是否有详细信息, true/false。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID, 如果有问题可以把此跟踪ID发给调度系统开发者, 以便排查问题。
ReturnCode	String	返回错误码, 0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	异常信息。
ReturnValue	String	无

请求示例

```
http(s)://[Endpoint]/?Action=GetNodeDetail
&EntityId=4003922
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": {
    "TemplateRules": {
      "TemplateRule": [
        {
          "TemplateName": "SQL task table rows, 1,7, 30 days fluctuation test",
          "ProjectName": "autotest",
          "MatchExpression": "dt=${yyyy}mmdd.11"
```

```

    "MatchExpression": "dt=${yyyymmdd-1}",
    "WarningThreshold": "40",
    "RuleCheckerRelationId": 1008005,
    "RuleType": 0,
    "MethodId": 8,
    "CriticalThreshold": "80",
    "FixCheck": false,
    "OnDuty": "050624",
    "Property": "table_count",
    "TemplateId": 7,
    "PropertyKey": "table_count",
    "MethodName": "table_count",
    "HistoryWarningThreshold": "history max:40%,history min:10%",
    "BlockType": 1,
    "CheckerId": 7,
    "TableName": "test_dqc_decimal_1119_2",
    "Id": 279580663,
    "EntityId": 4003922,
    "HistoryCriticalThreshold": "history max:80%,history min:50%",
    "Trend": "abs"
  },
  {
    "TemplateName": "SQL task table rows, 1,7, 30 days fluctuation test",
    "ProjectName": "autotest",
    "MatchExpression": "dt=${yyyymmdd-1}",
    "WarningThreshold": "10",
    "RuleCheckerRelationId": 1008007,
    "RuleType": 0,
    "MethodId": 8,
    "CriticalThreshold": "50",
    "FixCheck": false,
    "OnDuty": "050624",
    "Property": "table_count",
    "TemplateId": 7,
    "PropertyKey": "table_count",
    "MethodName": "table_count",
    "HistoryWarningThreshold": "history max:10%,history min:10%",
    "BlockType": 0,
    "CheckerId": 7,
    "TableName": "test_dqc_decimal_1119_2",
    "Id": 279580665,
    "EntityId": 4003922,
    "HistoryCriticalThreshold": "history max:50%,history min:50%",
    "Trend": "abs"
  },
  {
    "TemplateName": "SQL task table rows, 1,7, 30 days fluctuation test",
    "ProjectName": "autotest",
    "MatchExpression": "dt=${yyyymmdd-1}",
    "WarningThreshold": "10",
    "RuleCheckerRelationId": 1008009,
    "RuleType": 0,
    "MethodId": 8,
    "CriticalThreshold": "50",
    "FixCheck": false,

```



```

"OnDuty": "050624",
"Property": "table_count",
"TemplateId": 7,
"PropertyKey": "table_count",
"MethodName": "table_count",
"HistoryWarningThreshold": "history max:10%,history min:10%",
"BlockType": 0,
"CheckerId": 7,
"TableName": "test_dqc_decimal_1119_2",
"Id": 279580667,
"EntityId": 4003922,
"HistoryCriticalThreshold": "history max:50%,history min:50%",
"Trend": "abs"
}
],
},
"SelfserviceRules": {
"SelfserviceRule": [
{
"ProjectName": "autotest",
"MatchExpression": "dt=${yyyymmdd-1}",
"Checker": 9,
"RuleCheckerRelationId": 1008004,
"WhereCondition": "id>100",
"RuleType": 1,
"MethodId": 21,
"FixCheck": true,
"CheckerName": "compared with a fixed value",
"OnDuty": "050624",
"Property": "table_count",
"PropertyKey": "table_count",
"MethodName": "count/table_count",
"BlockType": 0,
"TableName": "test_dqc_decimal_1119_2",
"Id": 279580662,
"CheckerId": 6,
"EntityId": 4003922,
"Trend": "abs"
},
{
"ProjectName": "autotest",
"MatchExpression": "dt=${yyyymmdd-1}",
"Checker": 9,
"RuleCheckerRelationId": 1008006,
"WhereCondition": "id>100",
"RuleType": 1,
"MethodId": 21,
"FixCheck": true,
"CheckerName": "compared with a fixed value",
"OnDuty": "050624",
"Property": "table_count",
"PropertyKey": "table_count",
"MethodName": "count/table_count",
"BlockType": 0,

```

```
"TableName": "test_dqc_decimal_1119_2",
"Id": 279580664,
"CheckerId": 6,
"EntityId": 4003922,
"Trend": "abs"
},
{
  "ProjectName": "autotest",
  "MatchExpression": "dt=${yyyymmdd-1}",
  "Checker": 9,
  "RuleCheckerRelationId": 1008008,
  "WhereCondition": "id>100",
  "RuleType": 1,
  "MethodId": 21,
  "FixCheck": true,
  "CheckerName": "compared with a fixed value",
  "OnDuty": "050624",
  "Property": "table_count",
  "PropertyKey": "table_count",
  "MethodName": "count/table_count",
  "BlockType": 0,
  "TableName": "test_dqc_decimal_1119_2",
  "Id": 279580666,
  "CheckerId": 6,
  "EntityId": 4003922,
  "Trend": "abs"
},
{
  "ProjectName": "autotest",
  "MatchExpression": "dt=${yyyymmdd-1}",
  "Checker": 9,
  "RuleCheckerRelationId": 1008010,
  "WhereCondition": "id>100",
  "RuleType": 1,
  "MethodId": 21,
  "FixCheck": true,
  "CheckerName": "compared with a fixed value",
  "OnDuty": "050624",
  "Property": "table_count",
  "PropertyKey": "table_count",
  "MethodName": "count/table_count",
  "BlockType": 0,
  "TableName": "test_dqc_decimal_1119_2",
  "Id": 279580668,
  "CheckerId": 6,
  "EntityId": 4003922, "Trend": "abs"
}
]
},
"ReturnCode": "0"
}
```

6.25. GetNodeUpdateStatus

请求参数

参数名称	类型	是否必选	说明
Action	STRING	是	系统规定参数，取值GetNodeUpdateStatus。
AccountId	STRING	否	调度系统分配的调用API账号。
AccountPwd	STRING	否	调度系统分配的调用密码。
ClientName	STRING	否	调度系统分配的调用客户端名称。
BaseId	STRING	否	发起调用的云账号ID。
RequestId	STRING	否	ModifyNode是一个提交发布的API接口，该接口会返回一个ReturnValue，ReturnValue的值是一个RequestId。
TenantId	LONG	否	租户ID。
IsDetail	BOOLEAN	否	是否有详细信息，包括true和false。

返回参数

参数名称	类型	说明
RequestId	STRING	请求的跟踪ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	STRING	返回错误码，0为调用成功。
ReturnErrorSolution	STRING	问题的解决方案。
ReturnMessage	STRING	异常信息。
ReturnValue	STRING	ReturnValue的值是一个RequestId。

返回示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "0bbaa24515574698398786934e12b8",
  "returnValue": {
    "requestId": "0a64886b-1d63-4e5c-ad3e-d4eda4c2381c",
    "createTime": 1555065345000,
    "name": "ide-base",
    "status": "SUCCESS",
    "isDetail": true,
    "flowId": 28620441,
    "executeDetails": [
      {
        "uniqueId": "9745681_2",
        "cloudUUID": 111164284,
        "executeStatus": "SUCCESS",
        "executeMsg": "success",
        "dagId": null
      },
      {
        "uniqueId": "9745682_5",
        "cloudUUID": 111001035,
        "executeStatus": "SUCCESS",
        "executeMsg": "success",
        "dagId": null
      },
      {
        "uniqueId": "9745683_9",
        "cloudUUID": 111087244,
        "executeStatus": "SUCCESS",
        "executeMsg": "success",
        "dagId": null
      },
      {
        "uniqueId": "9745680_12",
        "cloudUUID": 111087242,
        "executeStatus": "SUCCESS",
        "executeMsg": "success",
        "dagId": null
      },
      {
        "uniqueId": "9745680_12",
        "cloudUUID": 111087242,
        "executeStatus": "INIT",
        "executeMsg": null,
        "dagId": null
      }
    ]
  },
  "success": true
}
```

6.26. GetResGroupAk

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetResGroupAk。
Baseld	String	是	阿里云账号ID。
Identifier	String	是	资源组identitier。
ProjectId	Long	是	DataWorks项目ID。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
ErrCode	Long	错误码
ErrMsg	String	错误信息。
RequestId	String	请求requestID。

请求示例

```
http(s)://[Endpoint]/?Action=GetResGroupAk
&Baseld=148****873474
&Identifier=3907f9****6b2506
&ProjectId=75059
&TenantId=2678***78690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0a98a368***97e56fd",
  "data": {
    "username": "zz_3907f9***08476b2506",
    "password": "vcn9*****",
    "rank": 4,
    "ossBackUp": false,
    "callback": false,
    "callbackUrl": ""
  },
  "errMsg": "success",
  "errCode": 0
}
```

6.27. GetResGroupGatewayList

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetResGroupGatewayList。
BaseId	String	是	阿里云账号ID。
Identifier	String	是	资源组identifier。
ProjectId	Long	是	DataWorks项目ID。
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
Data	List	返回数据。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=GetResGroupGatewayList
&Baseld=1486****73474
&Identifier=3907f9a5*****76b2506
&ProjectId=75059
&TenantId=2678****78690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0a98a35415574015996432719e2c97",
  "data": [{
    "clusterName": "3907f9a5832942f9b63b0908476b2506",
    "nodeName": "1.2.3.50",
    "nodeAddress": "1.2.3.50",
    "maxSlot": 32,
    "useSlot": 0,
    "state": 1,
    "live": false,
    "startMon": false,
    "regionId": 1,
    "nodeProperties": "{\"memory\":8,\"vCpu\":4}",
    "vCpu": 4,
    "memory": 8,
    "useDmu": 0
  }],
  "errMsg": "success",
  "errCode": 0
}
```

6.28. GetResGroupList

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetResGroupList。
AlisaDetail	Boolean	是	是否显示节点信息。
Baseld	String	是	云账号ID。
Level	String	是	资源组等级，项目级别，租户级别。取值可以是project和tenant。
ProjectId	Long	是	DataWorks工作空间ID。
ResourceGroupType	String	是	资源组类型。

参数名称	类型	是否必选	说明
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
Data	List	返回数据。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。

请求示例

```
http(s)://[Endpoint]/?Action=GetResGroupList
&AlisaDetail=true
&BaselId=14*****3474
&Level=project
&ProjectId=75059
&ResourceGroupType=di
&TenantId=2678*****690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0a98a3****52214e2c97",
  "data": [{
    "id": 29810****9522,
    "identifier": "group_267****178690",
    "name": "默认资源组",
    "nick": null,
    "isDefault": true,
    "networkType": null,
    "nodes": null,
    "maxSlot": null,
    "useSlot": null,
    "useDmu": null,
    "maxDmu": null,
    "projects": null,
    "resourceGroupType": null,
    "diResourceGroupType": 0,
    "buyType": "按量付费"
  }],
  {
    "id": 200000003001,
    "identifier": "3907f9a58***08476b2506",
    "name": "y1y1y1",
```



```
"nick": null,
"isDefault": false,
"networkType": 1,
"nodes": [{
  "clusterName": "3907f9a583****8476b2506",
  "nodeName": "1.2.3.50",
  "nodeAddress": "1.2.3.50",
  "maxSlot": 32,
  "useSlot": 0,
  "state": 1,
  "live": false,
  "startMon": false,
  "regionId": 1,
  "nodeProperties": "{\"memory\":8,\"vCpu\":4}",
  "vCpu": 4,
  "memory": 8,
  "useDmu": 0
}],
"maxSlot": 32,
"useSlot": 0,
"useDmu": 0,
"maxDmu": 16,
"projects": null,
"resourceGroupType": null,
"diResourceGroupType": 1,
"buyType": "按量付费"
},
{
  "id": 33224***49920,
  "identifier": "c43d****cb7fef18c137",
  "name": "cfr",
  "nick": null,
  "isDefault": false,
  "networkType": 1,
  "nodes": [{
    "clusterName": "c43da72697****c137",
    "nodeName": "deide",
    "nodeAddress": "10.0.0.22",
    "maxSlot": 8,
    "useSlot": 0,
    "state": 1,
    "live": false,
    "startMon": false,
    "regionId": 1,
    "nodeProperties": "{\"memory\":4,\"vCpu\":2}",
    "vCpu": 2,
    "memory": 4,
    "useDmu": 0
  ]},
  "maxSlot": 8,
  "useSlot": 0,
  "useDmu": 0,
  "maxDmu": 4,
  "projects": null,
```

```
"resourceGroupType": null,
"diResourceGroupType": 1,
"buyType": "按量付费"
},
{
  "id": 329080157014786,
  "identifier": "bd49c8014fb64f53835793e94932a9ac",
  "name": "hello",
  "nick": null,
  "isDefault": false,
  "networkType": 1,
  "nodes": [{
    "clusterName": "bd49c801*****e94932a9ac",
    "nodeName": "djed***qduew",
    "nodeAddress": "0.1.0.0",
    "maxSlot": 24,
    "useSlot": 0,
    "state": 1,
    "live": false, "startMon": false, "regionId": 1,
    "nodeProperties": "{\"memory\":8,\"vCpu\":8}", "vCpu": 8,
    "memory": 8,
    "useDmu": 0
  }],
  "maxSlot": 24,
  "useSlot": 0,
  "useDmu": 0,
  "maxDmu": 12,
  "projects": null,
  "resourceGroupType": null,
  "diResourceGroupType": 1,
  "buyType": "按量付费"
}],
"errMsg": "success",
"errCode": 0
}
```

6.29. GetTableColumn

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetTableColumn。
Baseld	String	是	云账号ID。
DatasourceName	String	是	数据源名称。
DatasourceType	String	是	数据源类型。
ProjectId	Long	是	DataWorks工作空间ID。

参数名称	类型	是否必选	说明
SubType	String	是	目前仅支持MySQL类型。
Table	String	是	数据源表名。
TenantId	Long	是	租户ID。
Id	String	否	数据源ID。

返回参数

参数名称	类型	说明
Data		数据源中指定表的字段列表。
ErrCode	Long	错误码。
ErrMsg	String	错误message。
RequestId	String	请求requestID。

请求示例

```
http(s)://[Endpoint]?Action=GetTableColumn
&BaseId=148682***73474
&DatasourceName=odps_first
&DatasourceType=odps
&ProjectId=75059
&SubType=public
&Table=t
&TenantId=2678***78690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc1748b15574030794246845e5f16",
  "data": {
    "splitPk": [],
    "columns": [{
      "name": "id",
      "type": "STRING"
    }],
    "partitionColumns": [],
    "extraInfo": {}
  },
  "errMsg": "success",
  "errCode": 0
}
```

6.30. GetTableList

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetTableList。
Baseld	String	是	阿里云账号ID。
DatasourceName	String	是	数据源名称。
DatasourceType	String	是	数据源类型。
ProjectId	Long	是	DataWorks项目ID。
SubType	String	是	数据源子类型。
Table	String	是	数据源表名。
TenantId	Long	是	租户ID。
Id	Long	否	数据源ID。
PageSize	Long	否	分页大小。

返回参数

参数名称	类型	说明
Data		返回数据。
ErrCode	Long	错误码。
ErrMsg	String	错误message。
RequestId	String	请求requestID。

请求示例

```
http(s)://[Endpoint]/?Action=GetTableList
&Baseld=1486821399873474
&DatasourceName=odps_first
&DatasourceType=odps
&ProjectId=75059
&SubType=public
&Table=t
&TenantId=267883377178690
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc17****2e5f16",
  "data": {
    "tables": ["c", "t", "t1", "t2", "wowo", "zzyx", "oss_test", "oss_tes00t", "test_input"]
  },
  "errMsg": "success",
  "errCode": 0
}
```

6.31. GetTaskLog

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值GetTaskLog。
AccountId	String	是	调度系统分配的调用API账号。
AccountPwd	String	是	调度系统分配的调用密码。
Baseld	String	是	发起调用的云账号ID。
ClientName	String	是	调度系统分配的调用客户端名称。
ProjectEnv	String	是	指定在哪个环境中搜索实例信息列表，生产-PROD、开发-DEV。
TenantId	Long	是	租户ID，固定为1。
TaskId	Long	否	实例ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	无

参数名称	类型	说明
ReturnMessage	String	错误信息。
ReturnValue	String	日志内容。

请求示例

```
http(s)://[Endpoint]/?Action=GetTaskLog
&AccountId=C00000
&AccountPwd=account_pwd
&BaseId=0I0001
&ClientName=aone_app_name
&ProjectEnv=DEV
&TenantId=1
&TaskId=20000000
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": "I'm alog",
  "ReturnCode": "0",
  "RequestId": "023dbca2-3****f23fa26",
  "ReturnErrorSolution": "",
  "ReturnMessage": ""
}
```

6.32. GetUserInfo

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值为 <i>GetUserInfo</i>

返回参数

参数名称	类型	描述
ErrCode	INTEGER	错误码
ErrMsg	STRING	错误消息
RequestId	STRING	请求ID
User	-	用户信息

6.33. ListConnection

该API用于搜索数据源。

请求参数

参数名称	字符类型	是否必选	示例值	描述
ProjectId	LONG	是	12345	数据源ID
Name	STRING	否	"myds"	用户ID，用于计算引擎数据源AK查询，忽略其它场景，默认为空
Keyword	BOOLEAN	否	"mydatasource"	是否隐藏敏感字段，默认为 <i>false</i>
Status	INTEGER	否	0	数据源状态 <ul style="list-style-type: none">0：删除1：正常2：禁用
BindEngineId	LONG	否	12345	绑定的计算引擎ID
EnvType	INTEGER	否	1	运行环境类型 <ul style="list-style-type: none">0：开发环境1：生产环境
BaseId	STRING	否	"034302"	用户ID，用于查询计算引擎数据源AK。忽略其它场景，默认为空
Verbose	BOOLEAN	否	true	是否显示连接串详情，默认为false
PageNum	INTEGER	否	1	页号，默认为1
PageSize	INTEGER	否	10	页大小，默认为10

返回参数

参数名称	字符类型	示例值	描述
requestId	STRING	c9afd89b-7aef-47c9-9902-e0500d843268	请求ID，全局唯一，用于排查问题

参数名称	字符类型	示例值	描述
errCode	LONG	0	错误码 <ul style="list-style-type: none"> 正确：errCode=0 错误：errCode>0，具体错误对应的错误码
errMsg	STRING	success	错误信息，当errCode=0时，errMsg为空字符串或"success"
data	PagingResult<Connection>	-	数据源分页结果
TotalNum	INTEGER	-	总记录数
PageNum	INTEGER	-	页号
PageSize	INTEGER	-	页大小
Connections	List<Connection>	-	数据源列表
Id	LONG	12345	数据源ID
TenantId	LONG	12345	租户ID
ProjectId	LONG	12345	项目ID
Name	STRING	"testDataSource"	数据源名称
Type	STRING	"mysql"	连接串类型
SubType	STRING	""	连接串子类型
EnvType	INTEGER	0	运行环境类型，0：开发环境 1：生产环境
Connection	STRING	{"username":"u1","password":"p1"}	数据源连接串
Sequence	INTEGER	300	排序字段
Status	INTEGER	0	数据源状态 <ul style="list-style-type: none"> 0：删除 1：正常 2：禁用
Description	STRING	"this is a testdatasource"	数据源描述
Shared	BOOLEAN	true	是否为租户内共享

参数名称	字符类型	示例值	描述
GmtCreate	DATE	2018-11-11 00:00:00	创建时间
GmtModified	DATE	2018-11-11 00:00:00	最近修改时间
Operator	STRING	"032214"	最近修改人
ConnectTime	DATE	2018-11-11 00:00:00	最近连通时间
ConnectStatus	INTEGER	0	连通状态 <ul style="list-style-type: none">0：未测试1：连通成功2：连通失败

请求示例

```
/?ProjectId=12345
&Keyword="mydatasource"
&<公共请求参数>
```

正常返回示例

JSON格式

```
{
  "data":{
    "TotalNum":35,
    "PageNum":1,
    "PageSize":10,
    "Connections":[
      {
        "Id":12345,
        "TenantId":12345,
        "ProjectId":12345,
        "Name":"testDataSource",
        "Type":"mysql",
        "SubType":"",
        "EnvType":0,
        "Connection":{"username\\":"u1\\","password\\":"p1\\"},
        "Sequence":300,
        "Status":0,
        "Description":"this is a test datasource",
        "Shared":0,
        "GmtCreate":"2018-11-11 00:00:00",
        "GmtModified":"2018-11-11 00:00:00",
        "Operator":"032214",
        "ConnectTime":"2018-11-11 00:00:00",
        "ConnectStatus":0
      },...
    ]
  },
  "errCode":0,
  "errMsg":"success",
  "requestId":"c9afd89b-7aef-47c9-9902-e0500d843268"
}
```

异常返回示例

JSON格式

```
{
  "data":null,
  "errCode":1101002001,
  "errMsg":"invalid parameter",
  "requestId":"c9afd89b-7aef-47c9-9902-e0500d843268"
}
```

6.34. ListDataServiceApps

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值为ListDataServiceApps

参数名称	类型	是否必选	描述
BaseId	STRING	否	用户ID
AppName	STRING	否	应用名称
PageSize	INTEGER	否	页大小
PageNum	INTEGER	否	页码

返回参数

参数名称	类型	描述
Data	-	返回数据
ErrCode	INTEGER	错误码
ErrMsg	STRING	错误信息
RequestId	STRING	请求ID

6.35. ListDataServiceAuthedApi

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值为ListDataServiceAuthedApi
BaseId	STRING	是	用户ID
ProjectId	LONG	是	项目空间ID
ApiStatus	STRING	否	API状态
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页大小
Verbose	BOOLEAN	否	是否显示详情

返回参数

参数名称	类型	描述
Data	-	返回数据
ErrCode	INTEGER	错误码

参数名称	类型	描述
ErrMsg	STRING	错误消息
RequestId	STRING	请求ID

6.36. ListPermission

查询权限点列表：ListPermissionRequest

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListPermission
baseId	STRING	否	账号ID
tenantId	LONG	否	租户ID
projectId	LONG	否	项目ID
modules	STRING	否	模块名

返回参数

参数名称	类型	描述
Data	-	业务数据
errCode	INTEGER	错误码 <ul style="list-style-type: none">正确：errCode=0错误：errCode>0，具体错误对应的错误码
requestId	STRING	请求ID，全局唯一，用于排查问题

请求示例

```
/?Action=ListPermission
&baseId=
&modules=
&projectId=
&tenantId=
&<公共请求参数>
```

6.37. ListProjectModule

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListProjectModule
ProjectId	LONG	否	项目空间ID

返回参数

参数名称	类型	描述
ErrCode	INTEGER	错误码
ErrMsg	STRING	错误信息
Modules	-	模块列表
RequestId	STRING	请求ID

6.38. ListProjectModules

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListProjectModule
ProjectId	LONG	否	项目空间ID

返回参数

参数名称	类型	描述
ErrCode	INTEGER	错误码
ErrMsg	STRING	错误信息
Modules	-	模块列表
RequestId	STRING	请求ID

6.39. ListProject

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListProject

返回参数

参数名称	类型	描述
ErrCode	LONG	错误码
ErrMsg	STRING	错误信息
ProjectList	-	项目空间列表
RequestId	STRING	请求ID

6.40. ListTenantModule

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListTenantModule
TenantId	LONG	否	租户ID

返回参数

参数名称	类型	描述
ErrCode	INTEGER	错误码
ErrMsg	STRING	错误信息
Modules	-	模块列表
RequestId	STRING	请求ID

6.41. ListUserPermission

查询权限点列表：ListUserPermission

请求参数

参数名称	类型	是否必选	描述
Action	STRING	是	系统规定参数，取值ListUserPermission

参数名称	类型	是否必选	描述
BaseId	STRING	否	账号ID
TenantId	LONG	否	租户ID
ProjectId	LONG	否	项目ID
Modules	STRING	否	模块名

返回参数

参数名称	类型	描述
Data	-	业务数据
ErrCode	INTEGER	错误码 <ul style="list-style-type: none">正确：errCode=0错误：errCode>0，具体错误对应的错误码
ErrMsg	STRING	错误信息，当ErrCode=0时，ErrMsg为空字符串或者“success”
RequestId	STRING	请求ID，全局唯一，用于排查问题

6.42. ModifyBusiness

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值ModifyBusiness。
AccountId	String	否	调度系统分配的调用账户。
AccountPwd	String	否	调度系统分配的调用密码。
BaseId	String	否	发起调用的用户ID。

参数名称	类型	是否必选	说明
BusinessModifyDto	String	否	<pre>{ "executeMethod": "UPDATE_BUSINESS ", "bizId": 100000003, "bizKey": "biz_key_01", "bizName": "biz_name_01_xxxx ", "solId": 100000002, "appld": 78918, "tenantId": 2797***8706 }</pre>
ClientName	String	否	调度系统分配的调用客户端标识。
TenantId	Long	否	租户ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	无
ReturnMessage	String	错误信息。
ReturnValue	String	日志内容。

6.43. ModifyNode

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值ModifyNode。
AccountId	String	否	调度系统分配的调用账户。
AccountPwd	String	否	调度系统分配的调用密码。
ClientName	String	否	调度系统分配的调用客户端标识。

参数名称	类型	是否必选	说明
ModifyDtoList	String	否	<pre>[{ "executeMethod": "CREATE_NODE", "nodeType": 0, "cronExpress": "0 0 0 * * ?", "owner": " {{user_id}}", "createUser": "{{user_id}}", "modifyUser": " {{user_id}}", "dependentType": 0, "nodeName": "locale_test_node1 18", "prgType": 99, "priority": 0, "appld": 14255, "cycType": 1, "codeSrc": "", "nodeFrom": "postman", "inputs": [{ "data": "test_dev_prod_de pend_single_root" }], "outputs": ["data": "locale_test_node1 18out"], "uniqueId": "locale_test_node6 " }]</pre>
ProjectEnv	String	否	环境标识，PROD / DEV。
BaseId	String	否	发起调用的阿里云用户ID。
TenantId	Long	否	租户ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。
ReturnValue	String	异步提交的RequestId，用于调用GetNodeUpdateStatus接口查询提交发布状态。

返回示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "82e4c****a39fbda",
  "returnValue": "82e4c04****f5a39fbda",
  "success": true
}
```

6.44. ModifySolution

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值ModifySolution。
AccountId	String	否	调度系统分配的调用账户。
AccountPwd	String	否	调度系统分配的调用密码。
BaseId	String	否	发起调用的阿里云用户ID。
ClientName	String	否	调度系统分配的调用客户端标识。
TenantId	Long	否	租户ID。
SolutionModifyDto	String	否	解决方案信息，JSON。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。
ReturnValue	String	无

SolutionModifyDto示例

```
{
  "executeMethod": "UPDATE_SOLUTION",
  "solId": 100000001,
  "solName": "sol_name_01_xxxx",
  "appld": 78918,
  "tenantId": 279717776488706
}
```

返回示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "53ca5f***11bab4d0975",
  "returnValue": {
    "solId": 100000001,
    "solName": "sol_name_01_xxxx",
    "appld": 78918,
    "tenantId": 279717776488706
  },
  "success": true
}
```

6.45. RerunTask

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值RerunTask。

参数名称	类型	是否必选	说明
AccountId	String	是	调度系统分配的调用API账号。
AccountPwd	String	是	调度系统分配的调用密码。
BaseId	String	是	发起调用的云账号ID。
ClientName	String	是	调度系统分配的调用客户端名称。
ProjectEnv	String	是	环境标识。
TaskId	Long	是	无
TenantId	Long	是	租户ID。

返回参数

参数名称	类型	说明
RequestId	String	返回的唯一字符串。
ReturnCode	String	错误码。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。
ReturnValue	Boolean	返回值。

返回示例

```
{
  "RequestId": "",
  "RetrunCode": 12344,
  "ReturnErrorSolution": "****",
  "ReturnMessage": "****",
  "ReturnValue": true
}
```

6.46. ResumeTask

请求参数

参数名称	类型	是否必选	说明
------	----	------	----

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值ResumeTask。
AccountId	String	否	调度系统分配的调用API账号。
AccountPwd	String	否	调度系统分配的调用密码。
BaseId	String	否	发起调用的云账号ID。
ClientName	String	否	调度系统分配的调用客户端名称。
ProjectEnv	String	否	环境类型，包括PROD和DEV。
TaskId	Long	否	实例ID。
TenantId	Long	否	租户ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	错误码。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。

返回示例

```
{
  "RequestId": "1212adf***s-adfsdsf",
  "ReturnCode": 0,
  "ReturnMessage": "",
  "ReturnErrorSolution": "",
  "ReturnValue": true
}
```

6.47. SearchBusiness

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值SearchBusiness。
AccountId	String	否	调度系统分配的调用账户。
AccountPwd	String	否	调度系统分配的调用密码。
BaseId	String	否	发起调用的阿里云用户ID。
BizId	Long	否	业务流程ID，可选。
ClientName	String	否	调度系统分配的调用客户端标识。
BizName	String	否	业务流程名称。
BizKey	String	否	业务流程 Key，可选。
TenantId	Long	否	租户ID。
ApplId	Long	否	项目空间ID。
ApplIds	String	否	项目空间 ID 列表，逗号分隔，可选。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪 ID，如有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。
ReturnValue		无

返回示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "d4605f***145312",
  "returnValue": [
    {
      "bizId": 10010759,
      "bizName": "0 控制节点",
      "bizKey": "10010759",
      "appId": 12344,
      "tenantId": 121**212
    }
  ],
  "success": true
}
```

6.48. SearchManualDagNodeInstance

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值SearchManualDagNodeInstance。
ProjectName	String	否	DataWorks工作空间名称。
DagId	Long	否	手动业务流程运行实例ID，CreateManualDag接口返回的值。

返回参数

参数名称	类型	说明
Data	String	返回的手动业务流程内部运行任务的列表。
ErrCode	String	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求ID。
Success	Boolean	是否成功。

6.49. SearchSolution

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值SearchSolution。
AccountId	String	否	调度系统分配的调用账户。
AccountPwd	String	否	调度系统分配的调用密码。
Baseld	String	否	发起调用的阿里云用户ID。
ClientName	String	否	调度系统分配的调用客户端标识。
Solid	Long	否	解决方案ID，可选。
SolName	String	否	解决方案名称。
TenantId	Long	否	租户ID。
Appld	Long	否	项目空间ID。

返回参数

参数名称	类型	说明
RequestId	String	请求的跟踪 ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	错误码。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。

返回示例

```
{
  "returnCode": "0",
  "returnErrorSolution": "",
  "returnMessage": "",
  "requestId": "1381****b98d87c",
  "returnValue": [
    {
      "solId": 1000000007,
      "solName": "解决方案测试 222",
      "appId": 33646,
      "tenantId": 27***9020672
    }
  ],
  "success": true
}
```

6.50. SearchTasks

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值SearchTasks。
AccountId	String	是	调度系统分配的调用API账号。
AccountPwd	String	是	调度系统分配的调用API的密码。
BaseId	String	是	发起调用API的用户ID。
ClientName	String	是	调度系统分配的调用客户端名称。
ProjectEnv	String	是	指定在哪个环境中搜索实例信息列表，生产-PROD、开发-DEV。
NodeId	Long	否	节点ID，可以在DataWorks节点配置页面进行查看。
TenantId	Long	是	租户ID，固定为1。
DagId	Long	否	实例所属的DagId，每次调用CreateDag接口时会返回ID，可以根据此ID作为DagId查询。

参数名称	类型	是否必选	说明
Bizdate	String	否	筛选条件：业务日期，格式为yyyy-mm-dd 00:00:00。

返回参数

参数名称	类型	说明
Count	Integer	满足搜索查询条件的实例总数。
RequestId	String	请求的跟踪ID，如果有问题可以把此跟踪ID发给调度系统开发者，以便排查问题。
ReturnCode	String	返回错误码，0为调用成功。
ReturnErrorSolution	String	问题的解决方案。
ReturnMessage	String	错误信息。
ReturnValue		无

请求示例

```
http(s)://[Endpoint]/?Action=SearchTasks
&AccountId=C00000
&AccountPwd=account-pwd
&BaseId=010001
&ClientName=aone_app_name
&ProjectEnv=PROD &TenantId=1
&Bizdate=2018-11-17 00:00:00
&DagId=10000000
&NodeId=1000001
&<公共请求参数>
```

返回示例

```
{
  "Count": 3236869,
  "ReturnValue": [
    {
      "DqcInstance": "",
      "BeginWaitResTime": "",
      "ResGroupId": 30004716,
      "AppId": 14255,
      "TaskId": 9993181585,
      "PrgName": "",
      "Gmtdate": "2018-11-17 00:00:00",
      "NodeId": 20636213,
      "DqcDescription": "",
      "BeginRunningTime": "",
      "AlisaReturnTime": "",
      "FinishTime": "",
      "GwLogLocalFile": "",
      "DagId": 300041822,
      "ParaValue": "",
      "DueTime": "2018-11-17 00:06:00",
      "BaseLineId": 332,
      "NodeName": "trewtes",
      "CycType": 0,
      "DagType": 2,
      "InGroupId": 1,
      "Priority": 1,
      "RerunAble": true,
      "Status": 1,
      "ResGroupIdentifier": "group_30004716",
      "GwProcessId": "",
      "PrgType": 10,
      "Bizdate": "2018-11-16 00:00:00",
      "ExecName": "/opt/taobao/tbdpapp/odpswrapper/odpswrapper.py",
      "BeginWaitTimeTime": "",
      "Gateway": "",
      "GwLogFile": "",
      "TaskType": 0
    }
  ]
}
```

6.51. TestConnectivity

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值TestConnectivity。
Baseld	String	是	云账号ID。

参数名称	类型	是否必选	说明
Connection	String	是	数据源连接串。
Name	String	是	数据源名称。
ProjectId	Long	是	DataWorks工作空间ID。
SubType	String	是	目前仅支持MySQL类型。
TenantId	Long	是	租户ID，固定为1。
Type	String	是	数据源类型。
Id	Long	否	数据源ID。
UseUserDefinedSecret	Boolean	否	是否使用系统中存储的登录数据源的密码。

返回参数

参数名称	类型	说明
Data	Boolean	是否连通。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求requestId。

请求示例

```
http(s)://[Endpoint]/?Action=TestConnectivity
&BaseId=1486*****474
&Connection={ "jdbcUrl": "jdbc:mysql://12.11.12.1:3306/data",
"username": "root ",
"password": "****",
"tag": "public" }
&Name="test"
&ProjectId=75059
&SubType=public
&TenantId=26788***178690
&Type=mysql
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0a98***7e2c95",
  "data": null,
  "errMsg": "测试数据源连通性失败:error message: Communications link failure\n\nThe last packet sent success
fully to the server was 0 milliseconds ago. The driver has not received any packets from the server. error wit
h code: PROJECT_DATASOURCE_CONN_ERROR",
  "errCode": 610008
}
```

6.52. UpdateDQCfollower

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值 UpdateDQCfollower。
ProjectName	String	否	项目名称
Id	Long	否	订阅关系ID
Follower	String	否	订阅实体，邮件/短信为工号钉钉/hook为http服务地址。
AlarmMode	Integer	否	告警模式 1：邮件 2：邮件和短信 3：钉钉群机器人/hook :4：钉钉群机器人@ALL

返回参数

参数名称	类型	说明
ReturnCode	String	返回码。
ReturnValue	Boolean	是否更新成功。

请求示例

```
http(s)://[Endpoint]/?Id=279580663
&ProjectName=autotest
&BlockType=1
&Checker=9
&Comment=测试
&CriticalThreshold=40
&EntityId=4003922
&ExpectValue=0
&MethodName=count/table_count
&Operator=50624
&Property=table_count
&PropertyType=table
&RuleType=0
&TemplateId=7
&Trend=abs
&WarningThreshold=20
&WhereCondition=60
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": true,
  "ReturnCode": "0"
}
```

6.53. UpdateDQCRule

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值UpdateDQCRule。
ProjectName	String	是	DataWorks工作空间名称。
Id	Long	是	规则ID。
BlockType	Integer	否	强弱性 0：弱规则 1：强规则。
EntityId	Long	否	分区表达式ID。
Comment	String	否	规则描述。
Checker	Integer	否	Checker的ID，详情请参见下文的Checker说明表。

参数名称	类型	是否必选	说明
ExpectValue	String	否	期望值。
Trend	String	否	趋势。
MethodName	String	否	采样方法，如 max/table_count-count_distinct/user_defined等。详细取值请参见下文的Method取值说明表。
Operator	String	否	修改人。
Property	String	否	字段。
PropertyType	String	否	字段类型。
RuleType	Integer	否	规则类型0：系统定义1：自定义。
WhereCondition	String	否	过滤条件或自定义SQL语句。
CriticalThreshold	String	否	红色阈值。
WarningThreshold	String	否	橙色阈值。
TemplateId	Integer	否	模板ID。

Checker说明表

Checker Name	Checker ID
7天平均值波动	2
30天平均值波动	3
1天周期比较	4
7天周期比较	5
30天周期比较	6
7天方差波动	7
30天方差波动	8
与固定值比较	9
1、7、30波动比较	10

Checker Name	Checker ID
上一周期比较	11

Method取值说明表

Method名称	Method取值
平均值	avg
列	count
求和	sum
最小值	min
最大值	max
唯一值个数	count_distinct
用户自定义	user_defined
表行数	table_count
表大小	table_size
空值数	null_value
空值率	null_value/table_count
重复率	(table_count-count_distinct) /table_count
重复数	table_count-count_distinct
离散值	discrete_value_check
count/table_count	count/table_count
唯一值率	count_distinct/table_count

返回参数

参数名称	类型	说明
ReturnCode	String	返回码。
ReturnValue	Boolean	是否更新成功。

请求示例

```
http(s)://[Endpoint]/?Action=UpdateDQCfollower
&AlarmMode=1
&Follower=50624
&Id=1724
&ProjectName=autotest
&<公共请求参数>
```

返回示例

```
{
  "ReturnValue": true,
  "ReturnCode": "0"
}
```

6.54. UpdateResGroupGateWay

请求参数

参数名称	类型	是否必选	说明
Action	String	是	系统规定参数，取值UpdateResGroupGateway。
Baseld	String	是	云账号ID。
Identifier	String	是	资源组identifier。
Memory	Integer	是	gateway内存。
NodeName	String	是	gateway节点名称。
ProjectId	Long	是	DataWorks工作空间ID。
TenantId	Long	是	租户ID。
VCpu	Integer	是	gateway CPU的数量。

返回参数

参数名称	类型	说明
Data	Boolean	是否修改成功。
ErrCode	Long	错误码。
ErrMsg	String	错误信息。
RequestId	String	请求RequestId。

请求示例

```
http(s)://[Endpoint]/?Action=UpdateResGroupGateWay
&BaseId=1486821399873474
&Identifier=3907f9a58***8476b2506
&Memory=4
&NodeName=hostname
&ProjectId=75059
&TenantId=26788a***90
&VCpu=9
&<公共请求参数>
```

返回示例

```
{
  "requestId": "0bc174***5f15",
  "data": true,
  "errMsg": "success",
  "errCode": 0
}
```

6.55. CheckMetaTable

CheckMetaTable用于检查表是否存在。

请求参数

参数名称	类型	是否必选	描述
TableGuid	STRING	否	表的唯一标识。

返回参数

参数名称	类型	描述
Data	Boolean	表是否存在。

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("CheckMetaTable");
cr.putQueryParameter("TableGuid", tableGuid);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.56. GetMetaDB

GetMetaDB用于查询MaxCompute项目的信息。

请求参数

参数名称	类型	是否必选	描述
DbGuid	STRING	否	数据库（项目）的唯一标识

返回参数

参数名称	数据类型	描述
AppGuid	STRING	数据库（项目）标识
ProjectName	STRING	项目名称
ProjectNameCn	STRING	项目的中文名称（业务描述）
OwnerId	STRING	项目Owner的云账号
OwnerName	STRING	项目Owner的名称
CreateTime	STRING	创建日期
ModifyTime	STRING	修改日期

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("GetMetaDB");
cr.putQueryParameter("DbGuid", appGuid);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.57. GetMetaTable

GetMetaTable用于获取表的详细信息。

请求参数

参数名称	类型	是否必选	描述
TableGuid	STRING	否	数据库（项目）的唯一标识。

返回参数

参数名称	数据类型	描述
AppGuid	STRING	表所在项目的标识
SrcType	INTEGER	来源类型，0表示ODPS。
LastDdlTime	STRING	最后表更新的时间
TableGuid	STRING	表的标识
OwnerId	STRING	表所有者的云账号
HeatDegree	INTEGER	表的热度
Labels	STRING	表的标签信息，多个标签之间使用逗号（,）分隔。
TableDesc	STRING	表的描述信息
HasPart	INTEGER	是否为分区表，1表示分区表。
IsVisible	INTEGER	是否可见，1表示可见。

参数名称	数据类型	描述
LifeCycle	INTEGER	生命周期
DataSize	LONG	表占用的存储空间
TableName	STRING	表的名称
CreateTime	STRING	表的创建时间
LastModifyTime	STRING	最后修改时间

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("GetMetaTable");
cr.putQueryParameter("TableGuid", tableGuid);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.58. GetMetaTableIntroWiki

GetMetaTableIntroWiki用于获取数据表的使用说明。

请求参数

参数名称	类型	是否必选	描述
TableGuid	STRING	是	数据库（项目）的唯一标识
WikiVersion	LONG	否	Wiki的版本号

返回参数

参数名称	数据类型	描述
GmtCreate	STRING	创建时间
GmtModified	STRING	修改时间
CreatorName	STRING	创建者的名称

参数名称	数据类型	描述
CreatorBaseId	STRING	创建者的BaseID
Content	STRING	表的描述信息

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("GetMetaTableIntroWiki");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("WikiVersion", 2);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.59. ListMetaColumnLineage

ListMetaColumnLineage用于查询字段的血缘关系。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
ColumnName	STRING	是	字段的名称
Direction	STRING	是	字段的上下游方向。UP表示上游，DOWN表示下游。

返回参数

参数	类型	描述
TableName	STRING	上游（或下游）表的名称
ColumnName	STRING	上游（或下游）字段的名称
ColumnGuid	STRING	上游（或下游）字段的唯一标识

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaColumnLineage");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("Direction", "DOWN");
cr.putQueryParameter("ColumnName", columnName);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.60. ListMetaTableChangeLog

ListMetaTableChangeLog用于查询表的变更日志。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
StartDate	STRING	否	开始日期，格式为yyyy-MM-dd。
EndDate	STRING	否	结束日期，格式为yyyy-MM-dd。
ChangeType	STRING	否	变更类型，包括CREATE_TABLE、ALTER_TABLE、DROP_TABLE、ADD_PARTITION和DROP_PARTITION。
ObjectType	STRING	否	主体类别，包括TABLE和PARTITION。
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页宽

返回参数

参数	类型	描述
GmtCreate	STRING	创建时间
GmtModified	STRING	修改时间
Guid	STRING	创建者名称
ChangeType	STRING	变更类型
ObjectType	STRING	主体类别
Operator	STRING	操作者名称
ChangeContent	STRING	变更内容

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableChangeLog");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("StartDate", "2020-03-22");
cr.putQueryParameter("EndDate", "2020-03-26");
cr.putQueryParameter("ChangeType", "ADD_PARTITION");
cr.putQueryParameter("ObjectType", "PARTITION");
cr.putQueryParameter("PageNum", String.valueOf(1));
cr.putQueryParameter("PageSize", String.valueOf(3));
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.61. ListMetaTableColumn

ListMetaTableColumn用于获取表的列信息。

请求参数

参数	类型	是否必选	描述
TableGuid	String	是	表的唯一标识

返回参数

参数	数据类型	描述
TableGuid	STRING	表唯一标识
TableName	STRING	表名称
ColumnGuid	STRING	字段唯一标识
ColumnName	STRING	字段名称
ColumnType	STRING	字段类型
SeqNumber	INTEGER	字段序号
IsPartitionCol	INTEGER	是否为分区字段，1表示是分区字段。
IsPrimaryKey	INTEGER	是否为主键，1表示是主键。
IsNullable	INTEGER	是否允许为空
Comment	STRING	字段备注

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableColumn");
cr.putQueryParameter("TableGuid", tableGuid);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.62. ListMetaTableImpact

ListMetaTableImpact用于查询表的影响分析结果。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
ColumnName	STRING	否	字段名称

参数	类型	是否必选	描述
Direction	STRING	是	包括 <i>OUT</i> 和 <i>PUT</i>

返回参数

参数	类型	描述
TableGuid	STRING	表的唯一标识

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableColumn");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("Direction", "OUTPUT");
cr.putQueryParameter("ColumnName", columnName);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.63. ListMetaTableIntroWiki

ListMetaTableIntroWiki用于查询表的使用说明。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识

返回参数

参数	类型	描述
TableGuid	STRING	表的唯一标识
GmtCreate	STRING	创建时间
GmtModified	STRING	修改时间
Creator	STRING	创建者的BaseID

参数	类型	描述
CreatorName	STRING	创建者的名称
CreatorBaseId	STRING	创建者的BaseID
Version	LONG	Wiki的版本号

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableIntroWiki");
cr.putQueryParameter("TableGuid", tableGuid);
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.64. ListMetaTableLineage

ListMetaTableLineage用于查询表的血缘关系。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
Direction	STRING	是	查询表的上下游方向，包括UP、DOWN和ALL。
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页宽

返回参数

参数	类型	描述
DstTableGuid	STRING	目标表的唯一标识
SrcTableGuid	STRING	来源表的唯一标识

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableLineage");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("Direction", "ALL");
cr.putQueryParameter("PageNum", String.valueOf(1));
cr.putQueryParameter("PageSize", String.valueOf(3));
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.65. ListMetaTableOutput

ListMetaTableOutput用于查询表的产出信息。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
StartDate	STRING	否	开始日期，格式为yyyy-MM-dd。
EndDate	STRING	否	结束日期，格式为yyyy-MM-dd。
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页宽

返回参数

参数	类型	描述
TableGuid	STRING	表的唯一标识
ProjectId	LONG	任务所属的DataWorks工作空间ID
TaskId	STRING	任务ID
TaskInstId	LONG	任务实例ID

参数	类型	描述
StartTime	STRING	开始时间
EndTime	STRING	结束时间
ExecuteTime	STRING	执行时间
WaitTime	STRING	等待时间

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTableOutput");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("StartDate", "2020-03-22");
cr.putQueryParameter("EndDate", "2020-03-26");
cr.putQueryParameter("PageNum", String.valueOf(2));
cr.putQueryParameter("PageSize", String.valueOf(2));
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.66. ListMetaTablePartition

ListMetaTablePartition用于获取表的分区信息。

请求参数

参数	类型	是否必选	描述
TableGuid	STRING	是	表的唯一标识
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页宽

返回参数

参数	类型	描述
AppGuid	STRING	项目标识

参数	类型	描述
TableGuid	STRING	表的唯一标识
TableName	STRING	表名称
PartitionGuid	STRING	分区的唯一标识
PartitionName	STRING	分区名称
CreateTime	STRING	创建时间
ModifyTime	STRING	修改时间
DataSize	LONG	占用的存储空间
Records	LONG	记录数量

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("ListMetaTablePartition");
cr.putQueryParameter("TableGuid", tableGuid);
cr.putQueryParameter("PageNum", String.valueOf(2));
cr.putQueryParameter("PageSize", String.valueOf(2));
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.67. SearchMetaTables

SearchMetaTables用于查询MaxCompute表的列表。

请求参数

参数	类型	是否必选	描述
Keyword	STRING	是	关键字
PageNum	INTEGER	否	页码
PageSize	INTEGER	否	页宽

返回参数

参数名称	数据类型	描述
AppGuid	STRING	项目的唯一标识
TableGuid	STRING	表的唯一标识
TableName	STRING	表名称
CreateTime	STRING	创建时间
LastDdlTime	STRING	最后变更结构的时间
LastModifyTime	STRING	最后更新时间
OwnerId	STRING	表的所有者的BaseID

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("SearchMetaTables");
cr.putQueryParameter("Keyword", tableGuid);
cr.putQueryParameter("PageNum", String.valueOf(1));
cr.putQueryParameter("PageSize", String.valueOf(2));
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.68. GetStorageAndCalcMetrics

GetStorageAndCalcMetrics用于查询指定项目或当前用户所属租户的存储和计算数据。

请求参数

参数	类型	是否必选	描述
AppGuids	STRING	否	appGuid使用逗号（,）分隔的字符串。
DateString	STRING	否	查询日期，默认为业务日期。

参数	类型	是否必选	描述
PageSize	INTEGER	否	页宽。

返回参数

参数	类型	描述
TotalStorage	LONG	表名称
TotalCpuCost	LONG	表的唯一标识

示例

```
private String product = "dataworks-private-cloud";
private String version = "2019-01-17";
private String appGuid = "odps.dw_meta";
private String tableGuid = "odps.dw_meta.dwd_d_odps_table";
private String columnName = "table_comment";
CommonRequest cr = new CommonRequest();
cr.setProduct(product);
cr.setVersion(version);
cr.setMethod(MethodType.GET);
cr.setAction("GetStorageAndCalcMetrics");
cr.putQueryParameter("AppGuids", appGuid);
cr.putQueryParameter("DateString", '2020-01-01');
CommonResponse cresp = client.getCommonResponse(cr);
assertNotNull(cresp);
assertNotNull(cresp.getData());
```

6.69. GetInstanceStatistic

GetInstanceStatistic用于获取实例的统计信息。

接口参数

参数	是否可选	取值	描述
Bizdate	否	STRING，格式为yyyy-MM-dd。	业务日期
Appld	否	LONG	工作空间ID

返回结果

```
{
  "ReturnErrorSolution": "",
  "ReturnCode": "0",
  "RequestId": "83190FF7-8EAF-44DB-9170-AC0AXXXX",
  "ReturnMessage": "",
  "ReturnValue": {
    "TotalCount": 15,
    "RunningCount": 0,
    "FailureCount": 0,
    "NotRunCount": 15,
    "SuccessCount": 0,
    "WaitTimeCount": 0,
    "WaitResourceCount": 0
  },
  "Success": true
}
```

调用示例

```
@Test
public void testGetInstanceStatistic() throws ClientException {
    CommonRpcRequest commonRpcRequest = new CommonRpcRequest(popProduct, popVersion, "GetInstanceStatistic");
    commonRpcRequest.setMethod(MethodType.POST);
    String projectId = getTestProperties("projectId");
    String bizdate = getTestProperties("bizdate");
    commonRpcRequest.putQueryParameter("Appld", projectId);
    commonRpcRequest.putQueryParameter("Bizdate", bizdate);
    HttpResponse response = client.doAction(commonRpcRequest);
    System.out.println(GsonUtils.gson.toJson(response.getHttpContentString()));
}
```

6.70. GetInstanceLog

GetInstanceLog用于获取任务实例的运行日志。

接口参数

参数	是否可选	取值	描述
InstanceId	否	LONG	实例ID
ProjectEnv	是	STRING，可选值包括 <i>PROD</i> 和 <i>DEV</i> 。	环境标识

返回结果

```
{
  "ReturnErrorSolution": "",
  "ReturnCode": "0",
  "RequestId": "CEC994D9-45BD-4531-B2CE-6XXXX",
  "ReturnMessage": "",
  "ReturnValue": "没有产生日志信息",
  "Success": true
}
```

调用示例

```
@Test
public void testGetInstanceLog() throws ClientException {
    CommonRpcRequest commonRpcRequest = new CommonRpcRequest(popProduct, popVersion, "GetInstanceLog");
    commonRpcRequest.setMethod(MethodType.POST);
    Long instanceId = Long.valueOf(getTestProperties("instanceId"));
    String projectEnv = getTestProperties("projectEnv");
    commonRpcRequest.putQueryParameter("InstanceId", instanceId);
    commonRpcRequest.putQueryParameter("ProjectEnv", projectEnv);
    HttpResponse response = client.doAction(commonRpcRequest);
    System.out.println(GsonUtils.gson.toJson(response.getHttpContentString()));
}
```

6.71. GetNodeCode

GetNodeCode用于获取节点的代码。

接口参数

参数	是否可选	取值	描述
NodeId	是	LONG	节点ID
ProjectEnv	是	STRING, 可选值包括 <i>PROD</i> 和 <i>DEV</i> , 默认值为 <i>PROD</i> 。	环境标识

返回结果

```
{
  "ReturnErrorSolution": "",
  "ReturnCode": "0",
  "RequestId": "30DB3056-F6F1-4A05-88EE-5XXXX",
  "ReturnMessage": "",
  "ReturnValue": "echo 'hello'"
}
```

调用示例

```
@Test
public void testGetNodeCode() throws ClientException {
    CommonRpcRequest commonRpcRequest = new CommonRpcRequest(popProduct, popVersion, "GetNodeCode");
    commonRpcRequest.setMethod(MethodType.POST);
    Long nodeId = Long.valueOf(getTestProperties("nodeId"));
    String projectEnv = getTestProperties("projectEnv");
    commonRpcRequest.putQueryParameter("NodeId", nodeId);
    commonRpcRequest.putQueryParameter("ProjectEnv", projectEnv);
    HttpResponse response = client.doAction(commonRpcRequest);
    System.out.println(GsonUtils.gson.toJson(response.getHttpContentString()));
}
```

6.72. QueryInstances

QueryInstances用于查询实例列表。

接口参数

参数	是否可选	取值	描述
Bizdate	否	STRING，格式为yyyy-MM-dd。	业务日期
ProjectId	否	LONG	工作空间ID
NodeId	是	LONG	节点ID
NodeName	是	STRING	节点名称
TaskStatus	是	STRING，可选值包括NOT_RUN、WAIT_TIME、WAIT_RESOURCE、RUNNING、CHECKING、CHECKING_CONDITION、FAILURE和SUCCESS。	任务状态
ProjectEnv	是	STRING，可选值包括PROD和DEV。	环境标识

返回结果

```
{
  "ReturnErrorSolution": "",
  "ReturnCode": "0",
  "RequestId": "7AFA7207-2F44-4ECC-AE77-XXXX",
  "ReturnMessage": "",
  "ReturnValue": [
    {
      "Status": "SUCCESS",
      "ModifyTime": 15881766XXXX,
      "InstanceId": 70652XXXX,
      "DagType": "DAILY",
      "NodeName": "延迟节点",
      "CreateTime": 1588088309461,
      "CycTime": 1588176600000,
      "BeginWaitTimeTime": 1588176XXXX,
      "DagId": 700126518687,
      "Bizdate": 1588089600000,
      "BeginRunningTime": 1588176671359,
      "FinishTime": 1588176680949,
      "NodeId": 700001891566,
      "BeginWaitResTime": 1588176601707
    },
    {
      "Status": "SUCCESS",
      "ModifyTime": 1588090445868,
      "InstanceId": 706518829092,
      "DagType": "DAILY",
      "NodeName": "延迟节点",
      "CreateTime": 1588001922168,
      "CycTime": 1588090200000,
      "BeginWaitTimeTime": 1588089826198,
      "DagId": 700126318686,
      "Bizdate": 1588003200000,
      "BeginRunningTime": 1588090305062,
      "FinishTime": 1588090445868,
      "NodeId": 700001891566,
      "BeginWaitResTime": 1588090203188
    }
  ]
}
```

调用示例

```
@Test
public void testQueryInstances() throws ClientException {
    CommonRpcRequest commonRpcRequest = new CommonRpcRequest(popProduct, popVersion, "QueryI
nstances");
    commonRpcRequest.setMethod(MethodType.POST);
    String projectId = getTestProperties("projectId");
    String projectEnv = getTestProperties("projectEnv");
    String bizdate = getTestProperties("bizdate");
    String pageNumber = getTestProperties("pageNumber");
    String nodeId = getTestProperties("nodeId");
    String nodeName = getTestProperties("nodeName");
    String taskStatus = getTestProperties("taskStatus");
    commonRpcRequest.putQueryParameter("Bizdate", bizdate);
    commonRpcRequest.putQueryParameter("ProjectId", projectId);
    commonRpcRequest.putQueryParameter("NodeId", nodeId);
    commonRpcRequest.putQueryParameter("PageNumber", pageNumber);
    commonRpcRequest.putQueryParameter("NodeName", nodeName);
    commonRpcRequest.putQueryParameter("ProjectEnv", projectEnv);
    commonRpcRequest.putQueryParameter("TaskStatus", taskStatus);
    HttpResponse response = client.doAction(commonRpcRequest);
    System.out.println(GsonUtils.gson.toJson(response.getHttpContentString()));
}
```

7. 获取AccessKey


访问身份验证中通过使用AccessKey ID和AccessKey Secret对称加密的方法来验证某个请求的发送者身份。AccessKey ID用于标示用户，AccessKey Secret是用户用于加密签名字符串。本章节将为您描述如何获取AccessKey。


前提条件

只有运营管理员和一级组织管理员可以获取组织AccessKey。

获取组织AccessKey

获取组织AccessKey的方法如下：


1. 管理员登录ASCM控制台。
2. 在页面顶部菜单栏上，单击**企业**。
3. 在**企业**页面的左侧导航栏中，单击**组织管理**。
4. 在组织结构中，单击要添加的上级组织后面的.
5. 在弹出的下拉菜单中，选择**获取accesskey**。
6. 在弹出的对话框中，查看组织Accesskey信息。


 **说明** 一级组织的AccessKey为系统自动分配，下级组织使用一级组织的AccessKey。

获取个人账号AccessKey

获取个人账号AccessKey的方法如下：

1. 管理员登录ASCM控制台。
2. 在系统界面右上角，单击当前登录用户的头像，选择**个人信息**。
3. 在**阿里云AccessKey**模块，您可以查看个人账户的AccessKey信息。

阿里云AccessKey 访问阿里云资源时，请使用此AccessKey		
区域	AccessKey ID 	AccessKey Secret 
cn-qingdao-sg-d01	xxxxxx	显示

 **说明** Accesskey ID和AccessKey Secret是您访问云资源时的密钥，具有该账号完整的权限，请您妥善保管。