阿里云
实时计算（流计算）

OpenAPI

文档版本：20190415

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *[-all|-t]*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *{stand | slave}*

目录

1 简介

您可以使用实时计算的OpenAPI来操作您在实时计算产品上的所有项目作业，达到和使用webUI操作相同的效果。

使用OpenAPI时，您需要先构造一个`DefaultAcsClient`类的客户端，传入相关配置，通过构造的客户端来发送和接受请求相关信息

所有的OpenAPI功能均成对出现，分为`x-request`和`x-response`两类，`request`的OpenAPI负责接受参数，通过用户创建的`client`客户端向实时计算发送请求，请求返回的结果是对应的`response`类，您可以通过get方法获取该类中存储的信息。

2 OpenAPI概览

OpenAPI概览

作业

OpenAPI	描述
CheckRawPlanJson	检测作业planjson的获取状态
CommitJob	提交作业
CreateJob	创建作业
DeleteJob	删除作业
GetJob	获取job信息
ListJob	搜索job
OfflineJob	下线作业
StartJob	启动作业
UpdateJob	更新作业
ValidateJob	作业语法检查

文件夹

OpenAPI	描述
MVFolder	移动文件夹
CreateFolder	创建文件夹
DeleteFolder	删除文件夹
GetFolder	获取文件夹
ListChildFolder	获取子目录

实例

OpenAPI	描述
BatchGetInstanceRunSummary	批量获取实例运行信息
GetInstanceResource	获取运行实例使用资源
GetInstance	获取运行实例详情

OpenAPI	描述
GetInstanceCheckpoint	获取运行实例的ckeckpoint
GetInstanceConfig	获取作业运行参数
GetInstanceDetail	获取实例运行的DAG图
GetInstanceExceptions	获取运行实例的failover信息
GetInstanceFinalState	获取运行实例最终状态
GetInstanceMetric	获取运行实例的metric信息
GetInstanceRunSummary	获取作业运行实例的运行概要
ListInstance	获取某个project下所有的运行实例
ModifyInstanceState	修改实例状态
GetRawPlanJson	获取原始执行计划

Package

OpenAPI	描述
CreatePackage	获取Package信息
DeletePackage	删除package
GetPackage	获得package信息
ListPackage	搜索package
GetRefPackageJob	获取引用特定package的作业
UpdatePackage	更新package

Queue

OpenAPI	描述
ListProjectBindQueue	查询project绑定的队列信息
ListProjectBindQueueResource	查询project绑定的队列的资源信息

3 RAM资源授权

默认情况下，您对自己创建的资源拥有完整的操作权限，可以使用本文档中列举的OpenAPI对资源进行操作。

但在主子账号的场景下，子账号刚创建时没有资格去操作主账号的资源。需要通过 *RAM*授权的方式，给予子账号操作主账号资源的权限。

说明:

如果您不需要跨账户进行实例资源的授权和访问，您可以跳过此章节。跳过这些部分并不影响您对文档中其余部分的理解和使用。

授权策略（Policy）

授权策略（Policy）描述授权的具体内容。授权内容主要包含效果（Effect）、资源（Resource）、对资源所授予的操作权限（Action）以及限制条件（Condition）这几个基本元素。以下内容以OpenAPI网关为例，为您介绍授权策略（Policy）。

- 系统授权策略

  OpenAPI网关已经预置了两个系统权限，`AliyunApiGatewayFullAccess`和`AliyunApiGatewayReadOnlyAccess`，可以在RAM控制台 > 策略管理 进行查看。

  - AliyunApiGatewayFullAccess: 管理员权限，拥有主帐号下包含OpenAPI分组、OpenAPI、流控策略、应用等所有资源的管理权限。
  - AliyunApiGatewayReadOnlyAccess：可以查看主帐号下包含OpenAPI分组、OpenAPI、流控策略、应用等所有资源，但不可以操作。

- 授权策略

  您可以根据需要自定义管理权限，支持更为精细化的授权。可以为某个操作，也可以是某个资源。如：`GetUsers`的编辑权限。

  您可以在RAM控制台 > 策略管理 > 自定义授权策略查看已经定义好的自定义授权。

  

  说明:

  - 自定义授权查看、创建、修改、删除方法请参见授权策略管理。
  - 授权策略内容填写方法请参见Policy 基本元素、Policy语法结构和下文的授权策略。

- 示例:

```
{
```

```
  "Version": "1",
  "Statement": [
    {
  "Action": "apigateway:Describe*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

此示例表示：允许所有的查看操作。

- Action（操作名称列表）格式为：

  `"Action":"<service-name>:<action-name>"`

  其中：

  - `service-name`为：阿里云产品名称，请填写`apigateway`。
  - `action-name` 为：OpenAPI接口名称，请参照下表，支持通配符 `*`。

    ```
    "Action": "apigateway:Describe*" --表示所有的查询操作。
    "Action": "apigateway:*" --表示API网关所有操作。
    ```

- Resource（操作对象列表）

  Resource通常指操作对象， OpenAPI网关中的OpenAPI分组、流控策略、应用都被称为Resource，语法格式：

  `acs:<service-name>:<region>:<account-id>:<relative-id>`

  其中：

  - `acs`：Alibaba Cloud Service的首字母缩写，表示阿里云的公有云平台。
  - `service-name`为：阿里云产品名称，请填写 `apigateway`。
  - `region`：地区信息，可以使用通配符号`*`来表示所有区域。
  - `account-id`：账号ID，比如`1234567890123456`，也可以用`*`代替。
  - `relative-id`：与API 网关相关的资源描述部分，这部分的格式描述支持类似于一个文件路径的树状结构。

  Resource示例：

  `acs:apigateway:$regionid:$accountid:apigroup/$groupId`

  Resource语法：

  `acs:apigateway:*:$accountid:apigroup/`

鉴权规则

以OpenAPI网关为例：

action-name	资源（Resource）
AbolishApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId

4 调用方式

4.1 签名机制

阿里云会对每个访问的请求进行身份验证，所以无论使用HTTP还是HTTPS协议提交请求，都需要在请求中包含签名（Signature）信息。通过使用 `Access Key ID` 和 `Access Key Secret` 进行对称加密的方法来验证请求的发送者身份。`Access Key ID`和`Access Key Secret`由阿里云官方颁发给访问者（可以通过阿里云官方网站申请和管理），其中`Access Key ID`用于标识访问者的身份；`Access Key Secret`是用于加密签名字符串和服务器端验证签名字符串的密钥，仅您和阿里云知道秘钥信息，请严格保密。

说明:

阿里云提供了多种语言的SDK及第三方SDK，可以免去您对签名算法进行编码的工作。详情请参见阿里云开发工具包*#SDK#*。

签名操作

您在访问时，需要按照下面的方法对请求进行签名处理。

1. 使用请求参数构造规范化的请求字符串（Canonicalized Query String）。

   a. 参数排序。 按照参数名称的字典顺序对请求中所有的请求参数（包括公共请求参数和接口的自定义参数，但不包括公共请求参数中的 `Signature` 参数）进行排序。

说明:

当使用GET方法提交请求时，这些参数就是请求URI中的参数部分（即URI中?之后由&连接的部分）。

b. 参数编码。 对排序之后的请求参数的名称和值分别用UTF-8字符集进行URL编码。编码的规则如下。

A. 对于字符A~Z、a~z、0~9 以及字符-、_、.、~不编码；

B. 对于其它字符编码成%XY的格式，其中XY是字符对应ASCII 码的16进制表示。比如英文的双引号”对 应的编码为%22；

C. 对于扩展的UTF-8字符，编码成%XY%ZA...的格式；

D. 英文空格 要编码成 %20，而不是加号+。

该编码方式和一般采用的application/x-www-form-urlencoded MIME格式编码算法（比如Java标准库中的 java.net.URLEncoder的实现）相似，但又有所不同。实现时，可以先用标准库的方式进行编码，然后把编码后的字符串中加号+替换成 %20、星号*替换成 %2A、%7E 替换回波浪号~，即可得到上述规则描述的编码字符串。这个算法可以用下面的percentEncode方法来实现：

```
private static final String ENCODING = "UTF-8";

private static String percentEncode(String value) throws
UnsupportedEncodingException {
return value != null ? URLEncoder.encode(value, ENCODING).replace
("+", "%20").replace("*", "%2A").replace("%7E", "~") : null;
}
```

c. 将编码后的参数名称和值用英文等号=进行连接。

d. 将等号连接得到的参数组合按参数排序排好的顺序依次使用&符号连接，即得到规范化请求字符串。

2. 将上一步构造的规范化字符串按照下面的规则构造成待签名的字符串。

```
StringToSign= HTTPMethod + “&” + percentEncode(“/”) + ”&” +
        percentEncode(CanonicalizedQueryString)
```

其中：

- HTTPMethod是提交请求用的HTTP方法，比如GET。
- percentEncode(“/”) 是按照参数排序中描述的URL编码规则对字符/进行编码得到的值，即%2F。
- percentEncode(CanonicalizedQueryString)是对步骤 1 中构造的规范化请求字符串按参数编码中描述的URL编码规则编码后得到的字符串。

3. 按照RFC2104的定义，计算待签名字符串StringToSign的HMAC值。

> 说明:
> 计算签名时使用的Key就是您持有的Access Key Secret并加上一个&字符（ASCII:38），使用的哈希算法是SHA1。

4. 按照 Base64 编码规则把上面的 HMAC 值编码成字符串，即得到签名值（Signature）。
5. 将得到的签名值作为Signature参数添加到请求参数中，即完成对请求签名的过程。

> 说明:
> 得到的签名值在作为最后的请求参数值提交给ECS服务器时，要和其它参数一样，按照 *RFC3986* 的规则进行URL编码。

示例

以DescribeRegions为例，假设使用的`Access Key Id`为`testid`，`Access Key Secret`为`testsecret`。那么签名前的请求 URL 为：

```
http://ecs.aliyuncs.com/
?TimeStamp=2016-02-23T12:46:24Z
&Format=XML
&AccessKeyId=testid
&Action=DescribeRegions
&SignatureMethod=HMAC-SHA1
&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2014-05-
26
&SignatureVersion=1.0
```

计算得到的待签名字符串 `StringToSign` 为：

```
GET
&%2F
&AccessKeyId%3Dtestid
&Action%3DDescribeRegions
&Format%3DXML&SignatureMethod%3DHMAC-SHA1
&SignatureNonce%3D3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf
&SignatureVersion%3D1.0
&TimeStamp%3D2016-02-23T12%253A46%253A24Z
&Version%3D2014-05-26
```

因为 `Access Key Secret`为`testsecret`，所以用于计算HMAC的Key为 `testsecret&`，计算得到的签名值为：`CT9X0VtwR86fNWSnsc6v8YGOjuE=`

将签名作为 Signature 参数加入到 URL 请求中，最后得到的 URL 为：

```
http://ecs.aliyuncs.com/
?SignatureVersion=1.0
&Action=DescribeRegions
&Format=XML
```

```
&SignatureNonce=3ee8c1b8-83d3-44af-a94f-4e0ad82fd6cf&Version=2014-05-
26
&AccessKeyId=testid
&Signature=CT9X0VtwR86fNWSnsc6v8YGOjuE%3D
&SignatureMethod=HMAC-SHA1
&TimeStamp=2016-02-23T12%3A46%3A24Z
```

4.2 请求结构

本文为您介绍OpenAPI服务的请求结构。

服务地址

API 的服务接入地址，如下所示：

地域	服务地址
杭州	`foas.aliyuncs.com`
cn-shanghai:上海	`foas.[RegionId].aliyuncs.com`

通信协议

为了获得更高的安全性，仅支持使用HTTPS通道发送OpenAPI请求。

字符编码

请求及返回结果都使用UTF-8 字符集进行编码。

4.3 返回结果

返回结果格式

调用OpenAPI服务后返回数据采用统一格式：

- 返回的HTTP状态码为`2xx`，代表调用成功。
- 返回的HTTP状态码为`4xx`或`5xx`，代表调用失败。

说明:

本文档中的返回示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

返回结果成功

调用成功返回的数据格式主要有XML和JSON两种，外部系统可以在请求时传入参数来制定返回的数据格式，默认为XML格式。返回结果成功示例如下：

```
{}
```

返回结果错误

调用接口出错后，将不会返回结果数据。调用方可根据每个接口对应的错误码以及下述公共错误码来定位错误原因。 当调用出错时，HTTP请求返回一个 4xx 或 5xx 的 HTTP 状态码。返回的消息体中是具体的错误代码及错误信息。另外还包含一个全局唯一的请求ID：RequestId 和一个您该次请求访问的站点ID：HostId。在调用方找不到错误原因时，可以联系阿里云客服，并提供该 HostId 和 RequestId，以便尽快为您解决问题。返回结果错误示例如下：

```
null
```

公共错误码

请参见：公共错误码表。

4.4 公共参数

公共参数指的是所有接口调用都需要用到的参数，包含公共请求参数和公共返回参数两种。

公共请求参数

公共请求参数是指每个接口都需要使用到的请求参数。

名称	类型	是否必选	描述
Format	String	否	返回值的类型，支持JSON与XML。默认为XML。
Version	String	是	OpenAPI版本号，为日期形式：YYYY-MM-DD，本版本对应为2018-11-11。
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID。
Signature	String	是	签名结果串，关于签名的计算方法，请参见签名机制。
SignatureMethod	String	是	签名方式，目前支持HMAC-SHA1。

Timestamp	String	是	请求的时间戳。日期格式按照 ISO8601标准表示，并需要使用UTC时间。格式为：`YYYY-MM-DDThh:mm:ssZ`，例如，`2014-05-26T12:00:00Z`（为北京时间2014年5月26日20点0分0秒）。
SignatureVersion	String	是	签名算法版本，目前版本是 `1.0`。
SignatureNonce	String	是	唯一随机数，用于防止网络重放攻击。您在不同请求间要使用不同的随机数值
ResourceOwnerAccount	String	否	本次OpenAPI请求访问到的资源拥有者账户，即登录用户名。此参数的使用方法，详见借助RAM实现子账号对主账号的资源访问（只能在RAM中可对ECS资源进行授权的Action中才能使用此参数，否则访问会被拒绝）。

公共请求参数示例

```
https://foas.aliyuncs.com/
?Format=xml
&Version=2018-11-11
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&Timestamp=2012-06-01T12:00:00Z
…
```

公共返回参数

您发送的每次接口调用请求，无论成功与否，系统都会返回一个唯一识别码 `RequestId`。

- XML示例

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!--结果的根结点-->
<接口名称+Response>
<!--返回请求标签-->
<RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
<!--返回结果数据-->
</接口名称+Response>
```

- JSON示例

```
{
"RequestId": "4C467B38-3910-447D-87BC-AC049166F216",
/* 返回结果数据 */
}
```

5 实例

5.1 BatchGetInstanceRunSummary

`BatchGetInstanceRunSummary`（批量获取实例运行信息）接口用来批量获取某个项目下，某一类作业（批作业或者流作业）的多个Job的运行实例情况。

BatchGetInstanceRunSummary请求参数

参数	类型	是否必选	示例值	描述
jobNames	String	是	`job1,job2`	作业名称，以逗号,分隔。
jobType	String	是	`FLINK_STREAM`	作业类型，批作业使用`FLINK_BATCH`，流作业使用`FLINK_STREAM`。
projectName	String	是	`test_project`	项目名称
RegionId	String	否	`cn-hangzhou-pre`	集团内用户使用： · `cn-hangzhou-pre` 公共云预发； · `cn-hangzhou-internal` 集团内生产； · `cn-hangzhou-internal-pre` 集团内预发。

BatchGetInstanceRunSummary返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，无实际含义，出现问题时方便foas核查。
RunSummarys			作业运行状态详情

└Id	Long	1	运行实例ID
└ActualState	String	RUNNING	作业返回实际运行状态： · WAITING 等待； · RUNNING 正在运行； · PAUSED 暂停； · TERMINATED 停止； · SUCCESS 成功（批作业）； · FAILED 失败（批作业）。
└ExpectState	String	RUNNING	作业期望状态： · WAITING 等待； · RUNNING 正在运行； · PAUSED 暂停； · TERMINATED 停止； · SUCCESS 成功（批作业）； · FAILED 失败（批作业）。
└LastErrorTime	Long	1548397575000	上次出现错误时间（时间戳，精确到毫秒）。
└LastErrorMessage	String	error	上次错误信息
└EngineJobHandler	String	aplication_x | xxxxx	YARN中为作业分配的名称：ApplicationID | JobID（JM分配的ID）。
└InputDelay	Long	300	业务延时
└JobName	String	job1	作业名称

BatchGetInstanceRunSummary请求示例

```
https://foas.aliyuncs.com/api/v2/projects/[project_name]/runsummary?ServiceCode=foas&jobType=FLINK_STREAM&jobNames=[job1]%2[job2]
```

5.2 GetInstance

使用GetInstance（获取运行实例详情） API可以获取作业运行的全部信息。

GetInstance请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	123	InstanceID，可以通过 `ListInstance` 接口、`Startjob` 等接口获得，流作业当前运行实例默认填`-1`。
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstance返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。
Instance			Instance详情
└Id	Long	`123`	InstanceID
└ProjectName	String	`project1`	项目名称
└JobName	String	`job1`	作业名称
└ActualState	String	`RUNNING`	Instance的实际状态： · WAITING：等待 · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（批作业）

└ExpectState	String	RUNNING	期望状态： · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（流作业）
└JobType	String	FLINK_BATCH	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
└ApiType	String	SQL	API类型： · DATASTREAM：Datastream作业 · SQL：SQL作业
└Code	String	code	Instance的运行代码。SQL作业返回SQL，datastream作业返回datastream配置。
└Properties	String	K=V	作业运行参数
└Packages	String	package1.jar	Instance引用的package，引用多个package逗号,分隔，未引用为空。
└Starter	String	starter1	启动人
└StartTime	Long	1548397575000	开始时间
└LastErrorTime	Long	1548397575000	最近错误时间
└LastErrorMessage	String	error	最近一次错误信息
└LastOperator	String	operator1	最近操作人
└LastOperateTime	Long	1548397575000	最近一次操作时间

└PlanJson	String	{a:b}	执行计划
└EngineVersion	String	blink_2.2.4	引擎版本
└EngineJobHandler	String	`application_xxxx | xxxx`	Instance在YARN中的名称：`ApplicaitonID | jobID (JM分配的Job ID)`。
└InputDelay	Long	`100`	业务延迟（毫秒）
└ClusterId	String	`d6wxwo5tnr muamx2ly3m7vkz`	集群ID
└QueueName	String	`root.default`	队列名称
└EndTime	Long	`1548397575000`	作业结束时间，未结束作业返回空值。

GetInstance请求示例

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

5.3 GetInstanceResource

GetInstanceResource API可以获取运行实例使用的CPU、Memory信息和提交时申请的CPU、Memory信息。未运行的作业调用GetInstanceResource OpenAPI会报错。

GetInstanceResource请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填`-1L`，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstanceResource返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Resource			资源情况
└TotalMB	Long	1024	实际运行使用的内存（MB）
└AllocatedMB	Long	2048	作业分配的内存（MB）
└TotalVirtualCores	Long	50	实际运行使用的vcore
└AllocatedVirtualCores	Long	100	作业分配的vcore

GetInstanceResource请求示例

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

5.4 GetInstanceCheckpoint

GetInstanceCheckpoint API可以获得正在运行实例作业的checkpoint，非运行状态的作业调用该方法会返回错误。

GetInstanceCheckpoint请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得。
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstanceCheckpoint返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Checkpoints	String	{a:b}	checkpint的具体信息

GetInstanceCheckpoint请求示例

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

5.5 GetInstanceConfig

GetInstanceConfig API可以获取到运行实例的运行相关参数，返回值Config中是kv对，可以提取自己想要的运行参数。对于没有运行的作业调用本方法会报错。

表 5-1: GetInstanceConfig请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得。
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstanceConfig返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Config	String	{a:b}	作业运行的相关参数。

GetInstanceConfig请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

5.6 GetInstanceDetail

GetInstanceDetail API可以获得运行实例的DAG（有向无环图）图。

GetInstanceDetail请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	`-1`	InstanceID，流作业只有一个运行实例，此处填`-1L`，指在线上运行的，批作业可以通过ListInstance接口或者Startjob接口获得。
jobName	String	是	`job1`	作业名称
projectName	String	是	`project1`	项目名称

GetInstanceDetail返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题
Detail	String	`{a:b}`	作业运行的DAG信息

GetInstanceDetail请求示例

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

5.7 GetInstanceExceptions

GetInstanceExceptions API可以获取作业运行failover的信息。

GetInstanceExceptions请求参数

参数	类型	是否必选	示例值	描述

instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstanceExceptions返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Exceptions	String	xxxxerror	failover的具体信息

GetInstanceExceptions请求示例

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

5.8 GetInstanceFinalState

GetInstanceFinalState API可以查看运行实例在YARN上的最终状态。

GetInstanceFinalState请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得。
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetInstanceFinalState返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Finalstate	String	RUNNING	YARN 上的最终状态

GetInstanceFinalState请求示例

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

5.9 GetInstanceMetric

GetInstanceMetric API可以获取foas上某个运行实例的所有metric信息，

说明:

GetInstanceMetric API可以同时获取多个metric信息。在填写metricjson时会涉及到很多参数，想要了解参数的详细含义可以参考*opentsdb*标准协议。

GetInstanceMetric请求参数

参数	类型	是否必选	示例值	描述

jobName	String	是	job1	作业名称
metricJson	String	是	{start: 1547637510000, limit: avg:sample:50, end: 1547638420000, queries: [{ downsample: 20s-avg, metric: blink.bayes_team.huayuan_test_job.delay, granularity: 20s, aggregator: max }, {downsample: 20s-avg, metric:blink.bayes_team.huayuan_test_job.fetched_delay, granularity: 20s, aggregator: max }] }	使用特定json来获取metric： · start：metric开始时间（使用13位时间戳，精确到毫秒） · limit：[取各条线值类型：max,avg,min]:[取值方式:bottom,above,sample]:[数目] · end：metric结束时间（使用13位 时间戳，精确到毫秒） · downsample：下采样方式[时间(秒)]-[取值方式：max,avg,min] · metric：blink.[项目名称].[作业名称].[指标名称]。常用指标名称如下表所示 - granularity：采样时间（每隔多少秒取一个点，需要根据start和end时长来取，点数太多会造成调用超时）。 - aggregator：聚合方式（按照采样时间将底层的点聚合后输出为一个点，聚合方式有最大值max，最小值min，平均值avg三种）。
projectName	String	是	project1	项目名称
instanceId	Long	否	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得。

GetInstanceMetric返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题
Metrics			metric信息详情
└MetricName	String	delay	metric名称
└Dps	Map	k:v	时间点和对应的metric 值
└Summary	Float	10.2	聚合后的metric值
└Tags	Map	k:v	metric标记

metric名称对应表

参数名称	指标名称
业务延时（ms）	delay
数据间隔延时（ms）	fetched_delay
数据等待延时（ms）	no_data_delay
作业失败率	task_failover.rate
Source的tps数据输入	tps.rate
Sink的数据输出	sink.outTps.rate
Source的RPS数据输出	parserTps.rate
Source的数据流量输入（byte）	inBps.rate
Source的脏数据比例	parserSkipCount
Checkpoint Duration（ms）	lastCheckpointDuration
Checkpoint大小（byte）	lastCheckpointSize
checkpoint对齐时间（ns）	checkpointAlignmentTime
checkpoint数量	checkpoints
获取state的时间（ns）	rocksdb_get.mean
写入state的时间（ns）	rocksdb_put.mean
seek（ns）	niagara_seek.meam / rocksdb_seek.mean
state大小（byte）	state.state_size / state.state_newsize
GMS GC Time（ms）	Status.JVM.GarbageCollector.ConcurrentMarkSweep.Time

GMS GC Rate	Status.JVM.GarbageCollector.ConcurrentMarkSweep.Count
WaterMark Delay（ms）	watermarkLatency
数据迟到丢弃TPS	lateRecordsDroppedRate
数据迟到累计丢弃数	numLateRecordsDropped
读TT延时（ms）	input.tt.readLatency
Task Input TPS	numRecordsInPerSecond.rate
Task Output TPS	numRecordsOutPerSecond.rate
Input Queue Usage	buffers.inPoolUsage
Output Queue Usage	buffers.outPoolUsage
Time Used In Processing Per Second（ns）	taskLatency.sum
Time Used In Waiting Oputput Per Second（ns）	waitOutput.sum
TaskLatency Histogram Mean（ns）	taskLatency.histogram.mean
WaitOutput Histogram Mean（ns）	waitOutput.histogram.mean
WaitInput Histogram Mean（ns）	waitInput.histogram.mean
PartitionLatency Mean（ns）	partitionLatency.mean
Process MEM Rss（kb）	Status.JVM.Memory.Process.rss
CPU Usage	Status.JVM.CPU.Usage
Memory Heap Used（byte）	Status.JVM.Memory.Heap.Used
Memory NonHeap Used（byte）	Status.JVM.Memory.NonHeap.Used
Threads Count	Status.JVM.Threads.Count
GC(CMS)	Status.JVM.GarbageCollector.ConcurrentMarkSweep.Count

GetInstanceMetric请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

5.10 GetInstanceRunSummary

GetInstanceRunSummary API是getInstance接口的轻量级版本，会返回作业运行时的部分信息，相比getInstance接口更加轻量级。

GetInstanceMetric请求参数

参数	类型	是否必选	示例值	描述
instanceId	Long	是	`-1`	InstanceID，流作业只有一个运行实例，此处填`-1L`，指在线上运行的，批作业可以通过`ListInstance`、`Startjob`等接口获得。
jobName	String	是	`job1`	作业名称
projectName	String	是	`project1`	项目名称

GetInstanceMetric返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题
RunSummary			运行概要信息
└Id	Long	`123`	InstanceID

└ActualState	String	RUNNING	Instance的实际状态： · WAITING：等待 · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（批作业）
└ExpectState	String	RUNNING	期望状态： · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（流作业）
└LastErrorTime	Long	1548397575000	最近错误时间
└LastErrorMessage	String	error	最近错误信息
└EngineJobHandler	String	application_xxxxx | xxxxx	Instance在YARN中的名称： ApplicaitonID | jobID（JM分配的Job ID）。
└InputDelay	Long	20	业务延时（毫秒）
└JobName	String	job1	作业名称

GetInstanceMetric请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

5.11 GetRawPlanJson

GetRawPlanJson API可以获取作业初始的执行计划。

GetRawPlanJson请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称
autoconfEnable	Boolean	否	true	是否开启智能调优，开启或根据作业历史metric，生成一份执行计划；不开启则使用默认执行计划，默认开启
expectedCore	Float	否	1	期望cpu数
expectedGB	Float	否	4	期望内存

GetRawPlanJson返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。
SessionId	String	`xxxxxx`	会话ID，需要传入checkrowplanjson接口中配合使用。

GetRawPlanJson请求示例

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

5.12 ListInstance

ListInstance API可以获取某个project下所有实例的详细信息。

ListInstance请求参数

参数	类型	是否必选	示例值	描述

jobName	String	是	job1	作业名称
jobType	String	是	FLINK_STREAM	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
projectName	String	是	project1	项目名称
actualState	String	否	RUNNING	Instance的实际状态： · WAITING：等待 · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（批作业）
apiType	String	否	SQL	api类型：DATASTREAM，SQL。
endBeginTs	Long	否	1548397575000	停止时间范围上限，和停止时间下限配合使用，搜索停止时间在该范围内的运行实例。使用13位时间戳，单位到毫秒。
endEndTs	Long	否	1548397575000	停止时间范围下限，和停止时间上限配合使用，搜索停止时间在该范围内的运行实例。使用13位时间戳，单位到毫秒。

expectState	String	否	RUNNING	期望状态： · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（流作业）
isArchived	Boolean	否	false	是否搜索归档实例，默认搜索非归档实例（默认值false）。对于流作业来说，同时只存在一个实例，所以当作业启动或者恢复运行的时候，就生成了新的实例，旧实例会被归档；对于批作业，运行结束两天的实例会被归档。
pageIndex	Integer	否	1	分页选项，第几页，从1开始。
pageSize	Integer	否	10	分页选项，每页的实例数
startBeginTs	Long	否	1548397575000	启动时间范围上限，和启动时间下限配合使用，搜索启动时间在该范围内的运行实例。使用13位时间戳，单位到毫秒

startEndTs	Long	否	1548397575000	启动时间范围下限，和启动时间上限配合使用，搜索启动时间在该范围内的运行实例。使用13位时间戳，单位到毫秒

ListInstance返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题
PageIndex	Integer	1	分页属性，第几页，从1开始
PageSize	Integer	10	分页属性，每页多少实例
TotalPage	Integer	5	总页数
TotalCount	Long	50	总实例数目
Instances			实例详情
└Id	Long	123	InstanceID
└ProjectName	String	project1	项目名称
└JobName	String	job1	作业名称
└ActualState	String	RUNNING	Instance的实际状态： · WAITING：等待 · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（批作业）

└ExpectState	String	RUNNING	期望状态： · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（流作业）
└JobType	String	FLINK_STREAM	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
└ApiType	String	SQL	API类型：DATASTREAM；SQL
└Code	String	code	实例代码
└Properties	String	`k=v`	作业运行配置参数
└Packages	String	`package1.jar`	Instance引用的package，多个逗号分隔，未引用为空
└Starter	String	xxxx	启动人
└StartTime	Long	1548397575000	作业启动时间
└LastErrorTime	Long	1548397575000	最近错误时间
└LastErrorMessage	String	error	最近一次错误信息
└LastOperator	String	xxxx	最近操作者
└LastOperateTime	Long	1548397575000	最近操作时间
└PlanJson	String	`{a:b}`	作业上线执行计划
└EngineVersion	String	`blink_2.2.4`	引擎版本

└EngineJobHandler	String	application_xxxx | xxxx	Instance在YARN中的名称：ApplicaitonID | jobID（JM分配的Job ID）。
└InputDelay	Long	20	业务延时（毫秒）
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
└QueueName	String	root.default	队列名称
└EndTime	Long	1548397575000	结束时间

ListInstance请求示例

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

5.13 ModifyInstanceState

ModifyInstanceState API可以修改运行实例的状态，实现对实例的操作

ModifyInstanceState请求参数

参数	类型	是否必选	示例值	描述
expectState	String	是	RUNNING	期望状态： · RUNNING：运行中 · PAUSED：暂停 · TERMINATED：停止 · SUCCESS：成功（批作业） · FAILED：失败（流作业）

instanceId	Long	是	-1	InstanceID，流作业只有一个运行实例，此处填-1L，指在线上运行的，批作业可以通过ListInstance、Startjob等接口获得。
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称
isFlush	Boolean	否	true	作业恢复时使用，确认是否使用最新配置。其他情况使用该参数无效，默认使用。

ModifyInstanceState返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

ModifyInstanceState请求示例

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

6 文件夹

6.1 CreateFolder

CreateFolder API可以为您为在项目对应的路径下创建文件夹，并在foas上保存文件夹ID，路径。

说明:

作业在foas控制台上显示的路径都是在作业开发目录之下，实际填写path的使用并不显示作业开发目录，作业开发目录对应根目录/。

CreateFolder请求参数

参数	类型	是否必选	示例值	描述
path	String	是	/path	文件夹路径
projectName	String	是	project1	项目名称

CreateFolder返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
FolderId	Long	123	文件夹ID

CreateFolder请求示例

```
http(s)://[Endpoint]/api/v2/projects/[projectName]/folders
&<公共请求参数>
```

6.2 DeleteFolder

DeleteFolder API可以删除空文件夹，对于非空文件夹会报错。

DeleteFolder请求参数

参数	类型	是否必选	示例值	描述
path	String	是	/path1/path2	文件夹路径
projectName	String	是	project1	项目名称

DeleteFolder返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

DeleteFolder请求示例

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

6.3 GetFolder

GetFolder API能够获取文件夹信息，保留文件夹ID和路径。

GetFolder请求参数

参数	类型	是否必选	示例值	描述
path	String	是	/path1/path2	文件夹路径
projectName	String	是	project1	项目名称

GetFolder返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Folder			文件夹详情
└FolderId	Long	123	文件夹ID
└Path	String	/path1/path2	文件夹路径

GetFolder请求示例

```
http(s)://[Endpoint]/?path=/aliyun
&projectName=myproject
&RegionId=reg
```

```
&<公共请求参数>
```

6.4 ListChildFolder

ListChildFolder API可以获取某个主目录下所有子目录的ID和路径。

ListChildFolder请求参数

参数	类型	是否必选	示例值	描述
path	String	是	/path	主目录路径
projectName	String	是	project	项目名称

ListChildFolder返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID：方便foas定位问题。
Folders			文件目录详情
└FolderId	Long	123	文件夹ID
└Path	String	/path	文件夹目录

ListChildFolder请求示例

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

6.5 MVFolder

MVFolder API可以移动文件夹，需要注意的是非空文件夹移动时，会将文件夹下的作业一起移动。

MVFolder请求参数

参数	类型	是否必选	示例值	描述
destPath	String	是	/folder1	目标路径
projectName	String	是	p	项目名称
srcPath	String	是	/dets	原路径

MVFolder返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

MVFolder请求示例

```
http(s)://[Endpoint]/?destPath=/folder1
&projectName=p
&RegionId=region
&srcPath=/dets
&<公共请求参数>
```

7 作业

7.1 CheckRawPlanJson

您可以组合调用CheckRawPlanJson和GetRawPlanJson接口，来获取某个作业的planjson。CheckRawPlanJson接口发出调用请求后会返回一个session ID，GetRawPlanJson获取这个session ID后可以查询到planjson的详细情况。

说明:

建议循环调用CheckRawPlanJson方法直到成功获取。

CheckRawPlanJson请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	单个作业名称
projectName	String	是	project1	项目名称
sessionId	String	是	123456	当使用CheckRawPlanJson发送请求后，会返回一个sessionId，将该sessionId填在此处。

CheckRawPlanJson返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID
PlanJsonInfo			PlanJson详情
PlanJsonInfo.Status	String	success	GetRawPlanJson的执行状态： · session in run：执行中 · success：成功 · fail：失败

PlanJsonInfo.PlanJson	String	{a:b}	执行计划
PlanJsonInfo.ErrorMessage	String	error	错误信息

CheckRawPlanJson请求示例

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

7.2 CommitJob

CommitJob API可以提交项目下作业的运行请求。

CommitJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

CommitJob返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas排查问题。

CommitJob请求示例

```
https://[endpoint]/api/v2/projects/[projectName]/jobs/[jobName]/commit
&<公共请求参数>
```

7.3 CreateJob

CreateJob API可以在指定项目下创建作业。

CreateJob请求参数

参数	类型	是否必选	示例值	描述

apiType	String	是	SQL	创建API类型： · DATASTREAM：Datastream作业 · SQL：SQL作业
clusterId	String	是	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
code	String	是	code1	SQL作业填写SQL代码，DataStream作业填写Foas上提交DataStream作业的相关参数。
engineVersion	String	是	`blink_2.2.4`	使用引擎的版本
jobName	String	是	job1	作业名称
jobType	String	是	FLINK_STREAM	作业类型： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
planJson	String	否	`{a:b}`	执行计划
projectName	String	是	project1	项目名称
queueName	String	是	`root.default`	队列名称
description	String	否	test	对作业相关描述备注
folderId	Long	是	123	文件夹ID
packages	String	否	`package1, package2`	jar包全名称，多个使用逗号分隔。
properties	String	否	`{k:v}`	作业运行的相关参数

CreateJob返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas排查问题。

CreateJob请求示例

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

7.4 DeleteJob

DeleteJob API可以删除已经下线的作业，未下线作业会报错。

DeleteJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

DeleteJob返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便Foas定位问题。

DeleteJob请求示例

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

7.5 GetJob

GetJob API可以获得作业的详细信息。

GetJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

GetJob返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
Job			作业详情
└JobName	String	job1	作业名称
└ProjectName	String	project	项目名称
└JobType	String	FLINK_STREAM	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
└ApiType	String	SQL	API类型： · DATASTREAM：Datastream作业 · SQL：SQL作业
└Code	String	code	作业详情： SQL作业返回SQL代码；datastream作业返回foas上datdastream作业的配置。
└PlanJson	String	{a:b}	作业上线的执行计划
└Properties	String	k:v	作业运行配置参数
└Packages	String	package1.jar	引用package名称，多个逗号分隔。
└IsCommitted	Boolean	true	是否上线状态
└Creator	String	xxxx	作业创建者
└CreateTime	Long	1548397575000	作业创建时间
└Modifier	String	xxxx	最近修改者
└ModifyTime	Long	1548397575000	最近修改时间
└Description	String	test	作业备注描述
└EngineVersion	String	blink_2.2.4	引擎版本
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
└QueueName	String	root.default	队列名称

└FolderId	Long	123	文件夹ID

GetJob请求示例

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

7.6 ListJob

ListJob API可以搜索某个project下满足条件的Job信息。

ListJob请求参数

参数	类型	是否必选	示例值	描述
projectName	String	是	project1	项目名称
apiType	String	否	SQL	API类型：DATASTREAM或SQL
clusterId	String	否	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
engineVersion	String	否	`blink_2.2.4`	引擎版本
folderId	Long	否	123	文件夹ID
jobName	String	否	job1	作业名称
jobType	String	否	FLINK_STREAM	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
pageIndex	Integer	否	1	分页属性，第几页
pageSize	Integer	否	10	分页属性，每页返回job数
queueName	String	否	root.default	队列名称

ListJob返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。

PageIndex	Integer	1	分页属性，第几页
PageSize	Integer	10	分页属性，每页多少个job
TotalPage	Integer	5	分页属性，总页数
TotalCount	Long	50	分页属性，总job数
Jobs			作业详情
└JobName	String	job1	作业名称
└ProjectName	String	project1	项目名称
└JobType	String	FLINK_STREAM	作业种类： · FLINK_STREAM：流作业 · FLINK_BATCH：批作业
└ApiType	String	SQL	API类型：DATASTREAM或SQL
└Code	String	code	作业代码
└PlanJson	String	{a:b}	执行计划
└Properties	String	k:v	作业运行配置
└Packages	String	package1.jar	package名称
└IsCommitted	Boolean	true	作业是否已经提交运行
└Creator	String	xxxx	创建者
└CreateTime	Long	1548397575000	作业创建时间
└Modifier	String	xxxx	最近修改者
└ModifyTime	Long	1548397575000	最近修改时间
└Description	String	test	作业备注描述
└EngineVersion	String	blink_2.2.4	引擎版本
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
└QueueName	String	root.default	队列名称
└FolderId	Long	123	文件夹ID

ListJob请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

7.7 OfflineJob

OfflineJob API可以下线作业，作业需要没有运行实例。

ListJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

ListJob返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。

ListJob请求示例

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

7.8 StartJob

StartJob API可以启动指定项目下某个作业。

StartJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称
parameterJson	Json	否	`{"a":"b"}`	作业启动相关参数，使用json格式，参数有如startOffset（启动位点，13位时间戳）。

StartJob返回参数

参数	类型	示例值	描述

RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
instanceId	Long	123	实例ID

StartJob请求示例

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

7.9 UpdateJob

UpdateJob API可以实现对作业属性的更新。

UpdateJob请求参数

参数	类型	是否必选	示例值	描述
clusterId	String	否	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
code	String	否	code	作业代码
engineVersion	String	否	blink_2.2.4	引擎版本
jobName	String	是	job1	作业名称
planJson	String	否	{a:b}	作业执行计划
projectName	String	是	project1	项目名称
queueName	String	否	root.default	队列名称
description	String	否	test	作业备注描述
folderId	Long	否	123	文件夹ID
packages	String	否	package1.jar	package名称
properties	String	否	k:v	作业运行配置参数

UpdateJob返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

UpdateJob请求示例

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

7.10 ValidateJob

ValidateJob API可以对作业进行语法检查。

ValidateJob请求参数

参数	类型	是否必选	示例值	描述
jobName	String	是	job1	作业名称
projectName	String	是	project1	项目名称

ValidateJob返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。
JobInOut			作业输入输出
└Inputs			输入数组信息
└Type	String	sls	表类型
└Workspace	String	xxxx	表所属项目
└Name	String	xxxx	表名
└Properties	Map	k:v	配置组
└Outputs			返回数组信息
└Type	String	sls	表类型
└Workspace	String	xxxx	表所属项目
└Name	String	xxxx	表名
└Properties	Map	k:v	配置组

ValidateJob请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

8 Package

8.1 CreatePackage

CreatePackage API可以获取到您在OSS中的package，将该package传送到作业的执行节点上来。

说明:

package的版本信息由您自行维护，上传时可以选填md5值，foas协助校验。公有云共享集群无法使用CreatePackage API，独享集群用户只能使用创建集群时的OSS信息，填写时无需填写完整信息（foas可以自动读取），只需要填写Osspath。

CreatePackage请求参数

参数	类型	是否必选	示例值	描述
ossBucket	String	否	`blinktest2.oss-cn-hangzhou-internal.aliyuncs.com`	OSSBucket
ossEndpoint	String	否	`oss-cn-hangzhou-internal.aliyuncs.com`	OSS接入点
ossOwner	String	否	user1	OSS所有者
ossPath	String	是	`path1/path2/a.jar`	package在oss中的路径
packageName	String	是	`package1.jar`	package名称
projectName	String	是	project1	项目名称

type	String	是	JAR	package 类型： · JAR：jar包 · DICTIONARY：普通文件 · SCRIPT：脚本 · PYTHON：python文件或者zip包
description	String	否	test	package的备注描述
md5	String	否	md5值	package的md5值
originName	String	否	package2	package别名

CreatePackage返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

CreatePackage请求示例

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

8.2 DeletePackage

DeletePackage API可以删除foas上通过createpackage保存的oss信息，删除后foas无法再获取对应的package信息。

> 说明:
> 使用DeletePackage API删除foas上不存在的oss package信息时系统会报错。已经上线的作业如已经引用该package，则删除时也会报错。

DeletePackage请求参数

参数	类型	是否必选	示例值	描述
packageName	String	是	package.jar	package名称

projectName	String	是	project1	项目名称

DeletePackage返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题。

DeletePackage请求示例

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

8.3 GetPackage

GetPackage API可以获得package的详细信息。

> 说明:
> GetPackage API只能获取到通过web或CreatePackage接口在foas上保存过相关信息的package。

GetPackage请求参数

参数	类型	是否必选	示例值	描述
packageName	String	是	package1.jar	package名称
projectName	String	是	project1	项目名称

GetPackage返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题
Package			package详情
└ProjectName	String	project1	项目名称
└PackageName	String	package1.jar	package名称
└Creator	String	xxxx	创建者
└Modifier	String	xxxx	最近修改者

└CreateTime	Long	1548397575000	创建时间（13位时间戳，精确到毫秒）。
└ModifyTime	Long	1548397575000	最近修改时间
└OriginName	String	package2.jar	package别名
└Type	String	JAR	package 类型： · JAR：jar包 · DICTIONARY：普通文件 · SCRIPT：脚本 · PYTHON：python文件或者zip包
└Md5	String	md5zhi	package的md5值
└Description	String	test	package备注描述
└OssEndpoint	String	oss-cn-hangzhou-internal.aliyuncs.com	Oss接入点
└OssBucket	String	blinktest2.oss-cn-hangzhou-internal.aliyuncs.com	Ossbucket
└OssOwner	String	xxxx	Oss所有者
└OssPath	String	path1/path2/a.jar	Oss路径

GetPackage请求示例

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

8.4 GetRefPackageJob

GetRefPackageJob API可以获取引用指定package的所有作业详情。

GetRefPackageJob请求参数

参数	类型	是否必选	示例值	描述
packageName	String	是	package1.jar	package名称
projectName	String	是	project1	项目名称

GetRefPackageJob返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题
PageIndex	Integer	1	返回第几页
PageSize	Integer	10	每页包含的job数
TotalCount	Long	50	总计多少job
TotalPage	Integer	5	总计多少页
Jobs			作业详情
└JobName	String	job1	作业名称
└ProjectName	String	project1	项目名称
└JobType	String	FLINK_STREAM	作业类型：FLINK_STREAM（流作业）或FLINK_BATCH（批作业）
└ApiType	String	SQL	API类型：DATASTREAM或SQL
└Code	String	code	作业code
└PlanJson	String	{a:b}	作业上线的执行计划
└Properties	String	k:v	作业运行参数配置
└Packages	String	package1.jar	作业引用的package名称，多个逗号分隔
└IsCommitted	Boolean	true	是否上线
└Creator	String	xxxx	创建者
└CreateTime	Long	1548397575000	作业创建时间
└Modifier	String	xxxx	最近修改者
└ModifyTime	Long	1548397575000	最近修改时间
└Description	String	test	作业备注描述
└EngineVersion	String	blink_2.2.4	引擎版本
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群ID

└QueueName	String	root.default	队列名称
└FolderId	Long	123	文件夹ID

GetRefPackageJob请求示例

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

8.5 ListPackage

ListPackage API可以搜索出指定project下满足特定条件的package信息。

ListPackage请求参数

参数	类型	是否必选	示例值	描述
packageName	String	是	package1	package名称
projectName	String	是	project1	项目名称
pageIndex	Integer	否	1	分页属性，第几页
pageSize	Integer	否	10	分页属性，每页包含package数量
type	String	否	JAR	package 类型： · JAR：jar包 · DICTIONARY：普通文件 · SCRIPT：脚本 · PYTHON：python文件或者zip包

ListPackage返回参数

参数	类型	示例值	描述
RequestId	String	FD0FF8C0-779A-45EB-9674-FF3E127B10D2	请求ID，方便foas定位问题
PageIndex	Integer	1	分页属性，第几页
PageSize	Integer	10	分页属性，每页包含package数
TotalPage	Integer	5	分页属性，总页面数

TotalCount	Long	50	分页属性，package总数
Packages			package详情
└ProjectName	String	project1	项目名称
└PackageName	String	`package1.jar`	package名称
└Creator	String	xxxx	创建者
└Modifier	String	xxxx	最近修改者
└CreateTime	Long	1548397575000	创建时间（13位时间戳，精确到毫秒）
└ModifyTime	Long	1548397575000	最近修改时间
└OriginName	String	`package2.jar`	package别名
└Type	String	JAR	package 类型： · JAR：jar包 · DICTIONARY：普通文件 · SCRIPT：脚本 · PYTHON：python文件或者zip包
└Md5	String	md5值	package的md5值
└Description	String	test	package备注描述
└OssEndpoint	String	`oss-cn-hangzhou-internal.aliyuncs.com`	Oss接入点
└OssBucket	String	`blinktest2.oss-cn-hangzhou-internal.aliyuncs.com`	OssBucket
└OssOwner	String	xxxx	Oss拥有者
└OssPath	String	`path1/path2/a.jar`	Oss路径

ListPackage请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

```
&<公共请求参数>
```

8.6 UpdatePackage

UpdatePackage API可以更新foas上保存的指定项目下package相关信息。

UpdatePackage请求参数

参数	类型	是否必选	示例值	描述
ossBucket	String	否	`blinktest2.oss-cn-hangzhou-internal.aliyuncs.com`	ossBucket
ossEndpoint	String	否	`oss-cn-hangzhou-internal.aliyuncs.com`	oss接入点
ossOwner	String	否	xxxx	oss所有者
ossPath	String	否	`path1/path2/a.jar`	oss路径
packageName	String	是	`package1.jar`	package名称
projectName	String	是	project1	项目名称
description	String	否	test	package注释描述
md5	String	否	md5值	package的md5值
originName	String	否	`package2.jar`	package别名
tag	String	否	xxxx	package的标记

UpdatePackage返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。

UpdatePackage请求示例

```
http(s)://[Endpoint]/?Action=undefined
```

&<公共请求参数>

9 Queue

9.1 ListProjectBindQueue

ListProjectBindQueue API可以查看指定project对应的队列相关的信息。

ListProjectBindQueue请求参数

参数	类型	是否必选	示例值	描述
clusterId	String	否	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
projectName	String	是	project1	项目名称
queueName	String	否	root.default	队列名称

ListProjectBindQueue返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题。
Queues			队列信息
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
└QueueName	String	`root.default`	集群名称

ListProjectBindQueue请求示例

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

9.2 ListProjectBindQueueResource

ListProjectBindQueueResource API可以查询project关联的队列的资源信息，返回对应Queue资源情况的具体信息。

ListProjectBindQueueResource请求参数

参数	类型	是否必选	示例值	描述

clusterId	String	否	d6wxwo5tnrmuamx2ly3m7vkz	集群ID
projectName	String	是	project1	项目名称
queueName	String	否	`root.default`	队列名称

ListProjectBindQueueResource返回参数

参数	类型	示例值	描述
RequestId	String	`FD0FF8C0-779A-45EB-9674-FF3E127B10D2`	请求ID，方便foas定位问题
Queues			队列详情
└ClusterId	String	d6wxwo5tnrmuamx2ly3m7vkz	集群名称
└QueueName	String	`root.default`	队列名称
└MinGpu	Integer	2	最小GPU数
└MaxGpu	Integer	10	GPU最大值
└MinMem	Integer	1024	最小内存数（MB）
└MaxMem	Integer	2048	内存最大值（MB）
└MinVCore	Integer	50	最小Vcore数
└MaxVCore	Integer	100	最大Vcore数
└UsedVCore	Integer	50	已使用Vcore
└UsedGpu	Integer	5	已使用GPU
└UsedMem	Integer	512	已使用内存（MB）

ListProjectBindQueueResource请求示例

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

10 OpenAPI DEMO

以下为公有云环境，使用实时计算的OpenAPI方式来操作实时计算产品上的所有项目作业的POM依赖和DEMO。

POM依赖如下。

```
<dependencies>

        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-foas</artifactId>
            <version>1.0.0</version>
        </dependency>

        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.5</version>
        </dependency>

    </dependencies>
```

说明:

若需要依赖aliyun-java-sdk-core，请使用4.3.0及以上版本。4.3.0版本依赖示例如下。

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.3.0</version>
</dependency>
```

DEMO如下。

```
package sample;

import com.aliyuncs.AcsRequest;
import com.aliyuncs.AcsResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.foas.model.v20181111.*;
import com.aliyuncs.http.FormatType;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.profile.IClientProfile;
import com.google.gson.Gson;

import java.util.HashMap;
import java.util.Map;

/**
 * 适用于公共云环境
 *
 * 作业的一般流程是:
 * 如果有依赖package的话，先创建package
```

```
 * 创建job->获取planjson->更新planjson到job->上线job->启动job->停止
instance
 * instance的状态一般如下(流作业为例，括号里的为动作):
 * job上线->UNKNOWN[启动job]->WAITING[等待]->RUNNING[暂停]->PAUSED[恢复]-
>RUNNING[停止]->TERMINATED
 */
public class PublicSample {

    //填写购买的foas所在的regionId
    private static final String regionId = "cn-shanghai";

    //云账号ak
    private static final String accessId = "xxxxxxxxx";
    private static final String accessKey = "yyyyyyyyy";

    private static final String projectName = "your project name";
    private static final String jobName = "your job name";

    // 文件夹与linux文件系统格式相同，以 / 开始，表示根目录，只支持绝对路径，不支
持相对路径
    private static final String folderName = "/新手任务";
    private static final String clusterId = "et2bts";//可以通过
listProjectBindQueue接口来获得
    private static final String queueName = "root.default";//可以通过
listProjectBindQueue接口来获得
    private static final String engineVersion = "current";
    private static final String packageName = "xxx.jar";

    private static final String sql =
            "--SQL\n" +
                    "--
*************************************************************************--
\n" +
                    "--CreateTime: 2018-12-29 14:35:04\n" +
                    "--Comment: 请输入业务注释信息\n" +
                    "--
*************************************************************************--
\n" +
                    "\n" +
                    "CREATE TABLE datahub_source (`name` INT, num
VARCHAR) WITH (\n" +
                    "    type = 'random'\n" +
                    ");\n" +
                    "CREATE TABLE datahub_sink (`name` INT, cnt BIGINT
) WITH (\n" +
                    "    type = 'PRINT'\n" +
                    ");\n" +
                    "INSERT INTO\n" +
                    "    datahub_sink\n" +
                    "SELECT\n" +
                    "    `name`,\n" +
                    "    COUNT (1)\n" +
                    "FROM\n" +
                    "    datahub_source\n" +
                    "GROUP BY\n" +
                    "    `name`";

    private static final String properties =
            "#checkpoint模式: EXACTLY_ONCE 或者 AT_LEAST_ONCE\n" +
                    "#blink.checkpoint.mode=EXACTLY_ONCE\n" +
                    "#checkpoint间隔时间，单位毫秒\n" +
                    "#blink.checkpoint.interval.ms=180000\n" +
                    "#rocksdb的数据生命周期，单位毫秒\n" +
                    "#state.backend.rocksdb.ttl.ms=129600000\n";
```

```
    private static <T extends AcsResponse> T getResponse(IAcsClient
client, AcsRequest<T> request) {
        AcsResponse response = null;
        try {
            //必须设置为json
            request.setHttpContentType(FormatType.JSON);
            request.setAcceptFormat(FormatType.JSON);

            response = client.getAcsResponse(request);
        } catch (Exception e) {
            //处理自己的异常逻辑，请注意异常中的requestId字段，排查问题时，需要
提供该值给服务端
            System.out.println(e.getMessage());
        }
        return (T) response;
    }

    public static void main(String[] args) throws Exception {

        IClientProfile profile = DefaultProfile.getProfile(regionId,
accessId, accessKey);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        client.setAutoRetry(false);//不重试

        //获取文件夹列表
        ListChildFolderRequest listChildFolderRequest = new ListChildF
olderRequest();
        listChildFolderRequest.setPath("/");
        listChildFolderRequest.setProjectName(projectName);
        ListChildFolderResponse listChildFolderResponse = PublicSample
.getResponse(client, listChildFolderRequest);
        System.out.println(new Gson().toJson(listChildFolderResponse
));

        //获取文件夹
        GetFolderRequest getFolderRequest = new GetFolderRequest();
        getFolderRequest.setPath(folderName);
        getFolderRequest.setProjectName(projectName);
        GetFolderResponse getFolderResponse = PublicSample.getResponse
(client, getFolderRequest);
        System.out.println(new Gson().toJson(getFolderResponse));

        Long folderId;
        GetFolderResponse.Folder folder = getFolderResponse.getFolder
();

        if (folder == null || folder.getFolderId() == null) {
            //创建文件夹
            CreateFolderRequest createFolderRequest = new CreateFold
erRequest();
            createFolderRequest.setPath(folderName);
            createFolderRequest.setProjectName(projectName);
            CreateFolderResponse createFolderResponse = PublicSample.
getResponse(client, createFolderRequest);
            System.out.println(new Gson().toJson(createFolderResponse
));
            folderId = createFolderResponse.getFolderId();
        } else {
            folderId = folder.getFolderId();
        }

        // sdk的package存在用户自己的oss上，此处需要将meta信息提供给foas,
```

```
        // 并对foas授权bucket读取权限(role:AliyunStreamDefaultRole), foas
在使用时会从用户指定的oss和bucket上下载package
        // 请自行保证oss的region与foas所在的region一致
        // 如果project建在小集群上，则不需要提供endpoint, bucket等信息，只需要
提供package的ossPath即可,
        // 因为用户的oss信息在创建小集群的时候已经提供过了，不需要重复提供
        CreatePackageRequest createPackageRequest = new CreatePack
ageRequest();
        createPackageRequest.setProjectName(projectName);
        createPackageRequest.setPackageName(packageName);
        createPackageRequest.setType("jar");
        createPackageRequest.setDescription("test package");
        createPackageRequest.setOssEndpoint("oss-cn-hangzhou.aliyuncs.
com");//提供的endpoint需要保证网络可通达
        createPackageRequest.setOssBucket("your bucket");
        createPackageRequest.setOssOwner("111111111");//oss的主账号uid
        createPackageRequest.setOssPath("aaa/bbb/ccc.jar");
        createPackageRequest.setMd5("123456");//可不填，不填的话就会跳过
md5校验，提供了MD5的话，在下载时会做完整性校验
        CreatePackageResponse createPackageResponse = PublicSample.
getResponse(client, createPackageRequest);
        System.out.println(new Gson().toJson(createPackageResponse));

        //更新package
        UpdatePackageRequest updatePackageRequest = new UpdatePack
ageRequest();
        updatePackageRequest.setProjectName(projectName);
        updatePackageRequest.setPackageName(packageName);
        updatePackageRequest.setOssPath("aaa/bbb/ddd.jar");
        UpdatePackageResponse updatePackageResponse = PublicSample.
getResponse(client, updatePackageRequest);
        System.out.println(new Gson().toJson(updatePackageResponse));

        //获取package详情
        GetPackageRequest getPackageRequest = new GetPackageRequest();
        getPackageRequest.setProjectName(projectName);
        getPackageRequest.setPackageName(packageName);
        GetPackageResponse getPackageResponse = PublicSample.
getResponse(client, getPackageRequest);
        System.out.println(new Gson().toJson(getPackageResponse));

        //根据条件搜索package
        ListPackageRequest listPackageRequest = new ListPackageRequest
();
        listPackageRequest.setProjectName(projectName);
        listPackageRequest.setPackageName("aa");//模糊匹配
        listPackageRequest.setType("jar");
        listPackageRequest.setPageIndex(1);//可不填，默认1
        listPackageRequest.setPageSize(30);//可不填，默认10
        ListPackageResponse listPackageResponse = PublicSample.
getResponse(client, listPackageRequest);
        System.out.println(new Gson().toJson(listPackageResponse));

        //列举project绑定的cluster和queue
        ListProjectBindQueueRequest listProjectBindQueueRequest = new
ListProjectBindQueueRequest();
        listProjectBindQueueRequest.setProjectName(projectName);
        ListProjectBindQueueResponse listProjectBindQueueResponse =
PublicSample.getResponse(client, listProjectBindQueueRequest);
        System.out.println(new Gson().toJson(listProjectBindQueue
Response));

        //查询project下绑定的queue的资源
```

```
        ListProjectBindQueueResourceRequest listProjectBindQueue
ResourceRequest = new ListProjectBindQueueResourceRequest();
        listProjectBindQueueResourceRequest.setProjectName(projectName
);
        ListProjectBindQueueResourceResponse listProjectBindQueue
ResourceResponse = PublicSample.getResponse(client, listProjec
tBindQueueResourceRequest);
        System.out.println(new Gson().toJson(listProjectBindQueue
ResourceResponse));

        //获取作业
        GetJobRequest getJobRequest = new GetJobRequest();
        getJobRequest.setProjectName(projectName);
        getJobRequest.setJobName(jobName);
        GetJobResponse getJobResponse = PublicSample.getResponse(
client, getJobRequest);
        System.out.println(new Gson().toJson(getJobResponse));

        GetJobResponse.Job job = getJobResponse.getJob();
        if (job == null || job.getJobName() == null) {
            //创建作业
            CreateJobRequest createJobRequest = new CreateJobRequest
();
            createJobRequest.setProjectName(projectName);
            createJobRequest.setJobName(jobName);
            createJobRequest.setCode(sql);
            createJobRequest.setProperties(properties);//无则不用设置
            //clusterId和queueName可以不用配置，不配置，会随机从项目中绑定的
queue中选择一个
            createJobRequest.setClusterId(clusterId);
            createJobRequest.setQueueName(queueName);
            //engineVersion可以不用配置，不配置，会使用current版本
            createJobRequest.setEngineVersion(engineVersion);
            createJobRequest.setDescription("test");
            createJobRequest.setPackages(packageName);//多个package之间
用英文逗号分隔
            createJobRequest.setJobType("flink_stream");//flink_stream
/flink_bacth 大小写不敏感
            createJobRequest.setApiType("sql");//datatream/sql 大小写不
敏感
            createJobRequest.setFolderId(folderId);

            CreateJobResponse createJobResponse = PublicSample.
getResponse(client, createJobRequest);
            System.out.println(new Gson().toJson(createJobResponse));
        } else {
            //修改作业
            UpdateJobRequest updateJobRequest = new UpdateJobRequest
();
            updateJobRequest.setProjectName(projectName);
            updateJobRequest.setJobName(jobName);
            updateJobRequest.setCode(sql);//设置想要修改的字段，不修改的字
段不用设置

            UpdateJobResponse updateJobResponse = PublicSample.
getResponse(client, updateJobRequest);
            System.out.println(new Gson().toJson(updateJobResponse));
        }

        //查找指定的package被哪些job所引用，该接口查询的是上线后的job引用关
系，未上线的job不会返回
        GetRefPackageJobRequest getRefPackageJobRequest = new
GetRefPackageJobRequest();
        getRefPackageJobRequest.setProjectName(projectName);
```

```
        getRefPackageJobRequest.setPackageName(packageName);
        GetRefPackageJobResponse getRefPackageJobResponse = PublicSamp
le.getResponse(client, getRefPackageJobRequest);
        System.out.println(new Gson().toJson(getRefPackageJobResponse
));

        //校验job是否存在语法错误
        ValidateJobRequest validateJobRequest = new ValidateJobRequest
();
        validateJobRequest.setProjectName(projectName);
        validateJobRequest.setJobName(jobName);
        ValidateJobResponse validateJobResponse = PublicSample.
getResponse(client, validateJobRequest);
        System.out.println(new Gson().toJson(validateJobResponse));

        //获取job的planjson，该接口是异步接口，提交完成之后，需要调用下面的
CheckRawPlanJson接口来获取结果:Session in run/success/fail
        GetRawPlanJsonRequest getRawPlanJsonRequest = new GetRawPlan
JsonRequest();
        getRawPlanJsonRequest.setProjectName(projectName);
        getRawPlanJsonRequest.setJobName(jobName);
        getRawPlanJsonRequest.setAutoconfEnable(true);//可以不填，默认为
false
        //设置这个job预期使用多少资源来运行，这个参数会影响到生成plan的大小，不提
供的话，底层引擎会默认生成资源量
        getRawPlanJsonRequest.setExpectedCore(2f);//这个参数与下面的参数可
以同时提供，也可以同时不提供，不允许一个提供，一个不提供
        getRawPlanJsonRequest.setExpectedGB(9f);//
        GetRawPlanJsonResponse getRawPlanJsonResponse = PublicSample.
getResponse(client, getRawPlanJsonRequest);

        CheckRawPlanJsonRequest checkRawPlanJsonRequest = new
CheckRawPlanJsonRequest();
        checkRawPlanJsonRequest.setProjectName(projectName);
        checkRawPlanJsonRequest.setJobName(jobName);
        checkRawPlanJsonRequest.setSessionId(getRawPlanJsonResponse.
getSessionId());

        String planJson;
        CheckRawPlanJsonResponse checkRawPlanJsonResponse;
        while (true) {
            checkRawPlanJsonResponse = PublicSample.getResponse(client
, checkRawPlanJsonRequest);
            System.out.println(checkRawPlanJsonResponse.getPlanJso
nInfo().getStatus());
            if (checkRawPlanJsonResponse.getPlanJsonInfo().getStatus
().equals("success")) {
                planJson = checkRawPlanJsonResponse.getPlanJsonInfo().
getPlanJson();
                break;
            } else if (checkRawPlanJsonResponse.getPlanJsonInfo().
getStatus().equals("fail")) {
                System.out.println(checkRawPlanJsonResponse.getPlanJso
nInfo().getErrorMessage());
                return;
            }
            Thread.sleep(1000);//每秒查询一次，不用过快
        }

        //修改作业planjson
        UpdateJobRequest updateJobRequest = new UpdateJobRequest();
        updateJobRequest.setProjectName(projectName);
        updateJobRequest.setJobName(jobName);
        updateJobRequest.setPlanJson(planJson);
```

```
        UpdateJobResponse updateJobResponse = PublicSample.getResponse
(client, updateJobRequest);
        System.out.println(new Gson().toJson(updateJobResponse));

        //上线作业
        CommitJobRequest commitJobRequest = new CommitJobRequest();
        commitJobRequest.setProjectName(projectName);
        commitJobRequest.setJobName(jobName);
        CommitJobResponse commitJobResponse = PublicSample.getResponse
(client, commitJobRequest);
        System.out.println(new Gson().toJson(commitJobResponse));

        //获取实例
        GetInstanceRequest getInstanceRequest = new GetInstanceRequest
();
        getInstanceRequest.setProjectName(projectName);
        getInstanceRequest.setJobName(jobName);

        //-1表示获取流作业的当前运行实例，注意批作业不可这样填，批作业需要填实际的
运行实例id
        getInstanceRequest.setInstanceId(-1L);
        GetInstanceResponse getInstanceResponse = PublicSample.
getResponse(client, getInstanceRequest);
        System.out.println(new Gson().toJson(getInstanceResponse));

        GetInstanceResponse.Instance instance = getInstanceResponse.
getInstance();

        {
            //启动作业，实例状态机参考文档
            if (instance == null || "UNKNOWN".equals(instance.
getActualState()) || "TERMINATED".equals(instance.getActualState())) {
                StartJobRequest startJobRequest = new StartJobRequest
();
                startJobRequest.setProjectName(projectName);
                startJobRequest.setJobName(jobName);

                Map map = new HashMap<String, String>();
                //startOffset表示启动点位
                map.put("startOffset", String.valueOf(System.
currentTimeMillis()));
                startJobRequest.setParameterJson(new Gson().toJson(map
));//json格式参数
                StartJobResponse startJobResponse = PublicSample.
getResponse(client, startJobRequest);
                System.out.println(new Gson().toJson(startJobResponse
));

                //等待启动成功
                while (true) {
                    GetInstanceRunSummaryRequest getInstanceRunSummar
yRequest = new GetInstanceRunSummaryRequest();
                    getInstanceRunSummaryRequest.setProjectName(
projectName);
                    getInstanceRunSummaryRequest.setJobName(jobName);
                    getInstanceRunSummaryRequest.setInstanceId(-1L);
                    GetInstanceRunSummaryResponse getInstanceRunSummar
yResponse = PublicSample.getResponse(client, getInstanceRunSummar
yRequest);
                    System.out.println(new Gson().toJson(getInstanc
eRunSummaryResponse));

                    if ("WAITING".equals(getInstanceRunSummaryResponse
.getRunSummary().getActualState())) {
```

```
                        System.out.println(String.format("lastErrorT
ime[%s], lastErrorMessage[%s]",
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorTime(),
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorMessage()));
                        Thread.sleep(1000);
                    } else {
                        break;
                    }
                }
            }
        }

        {
            // 作业启动成功后，可以调用如下的一些接口进行运维
            //获取instance的运行dag图
            GetInstanceDetailRequest getInstanceDetailRequest = new
GetInstanceDetailRequest();
            getInstanceDetailRequest.setProjectName(projectName);
            getInstanceDetailRequest.setJobName(jobName);
            getInstanceDetailRequest.setInstanceId(-1L);
            GetInstanceDetailResponse getInstanceDetailResponse =
PublicSample.getResponse(client, getInstanceDetailRequest);
            System.out.println(new Gson().toJson(getInstanceDetailRes
ponse));

            //获取instance的运行配置
            GetInstanceConfigRequest getInstanceConfigRequest = new
GetInstanceConfigRequest();
            getInstanceConfigRequest.setProjectName(projectName);
            getInstanceConfigRequest.setJobName(jobName);
            getInstanceConfigRequest.setInstanceId(-1L);
            GetInstanceConfigResponse getInstanceConfigResponse =
PublicSample.getResponse(client, getInstanceConfigRequest);
            System.out.println(new Gson().toJson(getInstanceConfigRes
ponse));

            //获取instance的checkpoint信息
            GetInstanceCheckpointRequest getInstanceCheckpointRequest
 = new GetInstanceCheckpointRequest();
            getInstanceCheckpointRequest.setProjectName(projectName);
            getInstanceCheckpointRequest.setJobName(jobName);
            getInstanceCheckpointRequest.setInstanceId(-1L);
            GetInstanceCheckpointResponse getInstanceCheckpoin
tResponse = PublicSample.getResponse(client, getInstanceCheckpoin
tRequest);
            System.out.println(new Gson().toJson(getInstanceCheckpoin
tResponse));

            //获取instance的运行时异常信息(failover信息)
            GetInstanceExceptionsRequest getInstanceExceptionsRequest
 = new GetInstanceExceptionsRequest();
            getInstanceExceptionsRequest.setProjectName(projectName);
            getInstanceExceptionsRequest.setJobName(jobName);
            getInstanceExceptionsRequest.setInstanceId(-1L);
            GetInstanceExceptionsResponse getInstanceException
sResponse = PublicSample.getResponse(client, getInstanceException
sRequest);
            System.out.println(new Gson().toJson(getInstanceException
sResponse));

            //查询instance的运行时各项指标的数据曲线信息
```

```
            //当前只支持查询最近两小时的曲线，metric的格式为：blink.{
projectName}.{jobName}.{metricName}
            String metricJson = "{\n" +
                    "\t\"start\": 1547637620000,\n" +
                    "\t\"limit\": \"avg:sample:50\",\n" +
                    "\t\"end\": 1547638420000,\n" +
                    "\t\"queries\": [{\n" +
                    "\t\t\"downsample\": \"20s-avg\",\n" +
                    "\t\t\"metric\": \"blink.bayes_team.huayuan_te
st_job.delay\",\n" +
                    "\t\t\"granularity\": \"20s\",\n" +
                    "\t\t\"aggregator\": \"max\"\n" +
                    "\t}, {\n" +
                    "\t\t\"downsample\": \"20s-avg\",\n" +
                    "\t\t\"metric\": \"blink.bayes_team.huayuan_te
st_job.fetched_delay\",\n" +
                    "\t\t\"granularity\": \"20s\",\n" +
                    "\t\t\"aggregator\": \"max\"\n" +
                    "\t}]\n" +
                    "}";

            GetInstanceMetricRequest getInstanceMetricRequest = new
GetInstanceMetricRequest();
            getInstanceMetricRequest.setProjectName(projectName);
            getInstanceMetricRequest.setJobName(jobName);
            getInstanceMetricRequest.setInstanceId(-1L);
            getInstanceMetricRequest.setMetricJson(metricJson);
            GetInstanceMetricResponse getInstanceMetricResponse =
PublicSample.getResponse(client, getInstanceMetricRequest);
            System.out.println(new Gson().toJson(getInstanceMetricRes
ponse.getMetrics()));

            //获取instance实际使用的资源信息（cpu/memory）
            GetInstanceResourceRequest getInstanceResourceRequest =
new GetInstanceResourceRequest();
            getInstanceResourceRequest.setProjectName(projectName);
            getInstanceResourceRequest.setJobName(jobName);
            getInstanceResourceRequest.setInstanceId(-1L);
            GetInstanceResourceResponse getInstanceResourceResponse =
PublicSample.getResponse(client, getInstanceResourceRequest);
            System.out.println(new Gson().toJson(getInstanceResourceR
esponse.getResource()));

            //批量获取多个作业的运行时状态
            BatchGetInstanceRunSummaryRequest batchGetInstanceRunS
ummaryRequest = new BatchGetInstanceRunSummaryRequest();
            batchGetInstanceRunSummaryRequest.setProjectName(
projectName);
            batchGetInstanceRunSummaryRequest.setJobType("flink_stream
");
            batchGetInstanceRunSummaryRequest.setJobNames("job1,job2,
job3");//多个job之间用英文逗号分隔，请确保都是stream job，并且job已存在且已上
线，否则会忽略错误的job
            BatchGetInstanceRunSummaryResponse batchGetInstanceRunS
ummaryResponse = PublicSample.getResponse(client, batchGetInstanceRunS
ummaryRequest);
            System.out.println(new Gson().toJson(batchGetInstanceRunS
ummaryResponse.getRunSummarys()));
        }

        {
            //暂停作业，实例状态机参考文档
            GetInstanceRunSummaryRequest getInstanceRunSummaryRequest
 = new GetInstanceRunSummaryRequest();
```

```
            getInstanceRunSummaryRequest.setProjectName(projectName);
            getInstanceRunSummaryRequest.setJobName(jobName);
            getInstanceRunSummaryRequest.setInstanceId(-1L);
            GetInstanceRunSummaryResponse getInstanceRunSummar
yResponse = PublicSample.getResponse(client, getInstanceRunSummar
yRequest);
            System.out.println(new Gson().toJson(getInstanceRunSummar
yResponse));

            if ("RUNNING".equals(getInstanceRunSummaryResponse.
getRunSummary().getActualState())) {
                ModifyInstanceStateRequest modifyInstanceStateRequest
 = new ModifyInstanceStateRequest();
                modifyInstanceStateRequest.setProjectName(projectName
);
                modifyInstanceStateRequest.setJobName(jobName);
                modifyInstanceStateRequest.setInstanceId(-1L);
                modifyInstanceStateRequest.setExpectState("PAUSED");
                ModifyInstanceStateResponse modifyInstanceStateR
esponse = PublicSample.getResponse(client, modifyInstanceStateRequest
);
                System.out.println(new Gson().toJson(modifyInst
anceStateResponse));

                //等待暂停成功
                while (true) {
                    getInstanceRunSummaryRequest = new GetInstanc
eRunSummaryRequest();
                    getInstanceRunSummaryRequest.setProjectName(
projectName);
                    getInstanceRunSummaryRequest.setJobName(jobName);
                    getInstanceRunSummaryRequest.setInstanceId(-1L);
                    getInstanceRunSummaryResponse = PublicSample.
getResponse(client, getInstanceRunSummaryRequest);
                    System.out.println(new Gson().toJson(getInstanc
eRunSummaryResponse));

                    if ("RUNNING".equals(getInstanceRunSummaryResponse
.getRunSummary().getActualState())) {
                        System.out.println(String.format("lastErrorT
ime[%s], lastErrorMessage[%s]",
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorTime(),
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorMessage()));
                        Thread.sleep(1000);
                    } else {
                        break;
                    }
                }
            }
        }

        {
            //恢复作业，实例状态机参考文档
            GetInstanceRunSummaryRequest getInstanceRunSummaryRequest
 = new GetInstanceRunSummaryRequest();
            getInstanceRunSummaryRequest.setProjectName(projectName);
            getInstanceRunSummaryRequest.setJobName(jobName);
            getInstanceRunSummaryRequest.setInstanceId(-1L);
            GetInstanceRunSummaryResponse getInstanceRunSummar
yResponse = PublicSample.getResponse(client, getInstanceRunSummar
yRequest);
```

```
            System.out.println(new Gson().toJson(getInstanceRunSummar
yResponse));

            if ("PAUSED".equals(getInstanceRunSummaryResponse.
getRunSummary().getActualState())) {
                ModifyInstanceStateRequest modifyInstanceStateRequest
 = new ModifyInstanceStateRequest();
                modifyInstanceStateRequest.setProjectName(projectName
);
                modifyInstanceStateRequest.setJobName(jobName);
                modifyInstanceStateRequest.setInstanceId(-1L);
                modifyInstanceStateRequest.setExpectState("RUNNING");
                ModifyInstanceStateResponse modifyInstanceStateR
esponse = PublicSample.getResponse(client, modifyInstanceStateRequest
);
                System.out.println(new Gson().toJson(modifyInst
anceStateResponse));

                //等待恢复成功
                while (true) {
                    getInstanceRunSummaryRequest = new GetInstanc
eRunSummaryRequest();
                    getInstanceRunSummaryRequest.setProjectName(
projectName);
                    getInstanceRunSummaryRequest.setJobName(jobName);
                    getInstanceRunSummaryRequest.setInstanceId(-1L);
                    getInstanceRunSummaryResponse = PublicSample.
getResponse(client, getInstanceRunSummaryRequest);
                    System.out.println(new Gson().toJson(getInstanc
eRunSummaryResponse));

                    if ("PAUSED".equals(getInstanceRunSummaryResponse.
getRunSummary().getActualState())) {
                        System.out.println(String.format("lastErrorT
ime[%s], lastErrorMessage[%s]",
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorTime(),
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorMessage()));
                        Thread.sleep(1000);
                    } else {
                        break;
                    }
                }
            }
        }

        {
            //停止作业，实例状态机参考文档
            GetInstanceRunSummaryRequest getInstanceRunSummaryRequest
 = new GetInstanceRunSummaryRequest();
            getInstanceRunSummaryRequest.setProjectName(projectName);
            getInstanceRunSummaryRequest.setJobName(jobName);
            getInstanceRunSummaryRequest.setInstanceId(-1L);
            GetInstanceRunSummaryResponse getInstanceRunSummar
yResponse = PublicSample.getResponse(client, getInstanceRunSummar
yRequest);
            System.out.println(new Gson().toJson(getInstanceRunSummar
yResponse));

            if ("RUNNING".equals(getInstanceRunSummaryResponse.
getRunSummary().getActualState())) {
                ModifyInstanceStateRequest modifyInstanceStateRequest
 = new ModifyInstanceStateRequest();
```

```
                modifyInstanceStateRequest.setProjectName(projectName
);
                modifyInstanceStateRequest.setJobName(jobName);
                modifyInstanceStateRequest.setInstanceId(-1L);
                modifyInstanceStateRequest.setExpectState("TERMINATED
");
                ModifyInstanceStateResponse modifyInstanceStateR
esponse = PublicSample.getResponse(client, modifyInstanceStateRequest
);
                System.out.println(new Gson().toJson(modifyInst
anceStateResponse));

                //等待停止成功
                while (true) {
                    getInstanceRunSummaryRequest = new GetInstanc
eRunSummaryRequest();
                    getInstanceRunSummaryRequest.setProjectName(
projectName);
                    getInstanceRunSummaryRequest.setJobName(jobName);
                    getInstanceRunSummaryRequest.setInstanceId(-1L);
                    getInstanceRunSummaryResponse = PublicSample.
getResponse(client, getInstanceRunSummaryRequest);
                    System.out.println(new Gson().toJson(getInstanc
eRunSummaryResponse));

                    if ("RUNNING".equals(getInstanceRunSummaryResponse
.getRunSummary().getActualState())) {
                        System.out.println(String.format("lastErrorT
ime[%s], lastErrorMessage[%s]",
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorTime(),
                                getInstanceRunSummaryResponse.
getRunSummary().getLastErrorMessage()));
                        Thread.sleep(1000);
                    } else {
                        break;
                    }
                }
            }
        }

        //下线作业
        OfflineJobRequest offlineJobRequest = new OfflineJobRequest();
        offlineJobRequest.setProjectName(projectName);
        offlineJobRequest.setJobName(jobName);
        OfflineJobResponse offlineJobResponse = PublicSample.
getResponse(client, offlineJobRequest);
        System.out.println(new Gson().toJson(offlineJobResponse));

        //删除作业
        DeleteJobRequest deleteJobRequest = new DeleteJobRequest();
        deleteJobRequest.setProjectName(projectName);
        deleteJobRequest.setJobName(jobName);
        DeleteJobResponse deleteJobResponse = PublicSample.getResponse
(client, deleteJobRequest);
        System.out.println(new Gson().toJson(deleteJobResponse));

        //删除package
        DeletePackageRequest deletePackageRequest = new DeletePack
ageRequest();
        deletePackageRequest.setProjectName(projectName);
        deletePackageRequest.setPackageName(packageName);
        DeletePackageResponse deletePackageResponse = PublicSample.
getResponse(client, deletePackageRequest);
```

```
        System.out.println(new Gson().toJson(deletePackageResponse));
    }
}
```