阿里云
大数据轻量专有云

开发指南

产品版本：V1.1.0

文档版本：20180327

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。
斜体	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 MaxCompute

1.1 SDK介绍

在本文档中，我们仅会对较为常用的MaxCompute核心接口做简短介绍，更多详细信息请参阅获取到的SDK中的SDK Java Doc。

包名	描述
odps-sdk-core	MaxCompute基础功能。 封装了基础的MaxCompute概念及其编程接口，包括odps、project、table等，tunnel相关的功能也在此包。
odps-sdk-core-internal	MaxCompute扩展功能。 封装了一些不常用的MaxCompute概念和操作，例如Event、XFlow等。
odps-sdk-commons	MaxCompute基础设施。 包含TableSchema、Column、Record、OdpsType等基础设施和一些util的封装。
odps-sdk-udf	MaxCompute UDF编程接口。
odps-sdk-mapred	MaxCompute MapReduce作业编程接口。

1.2 AliyunAccount

阿里云认证账号。输入参数为accessId及accessKey，是阿里云用户的身份标识和认证密钥。此类用来初始化MaxCompute。

1.3 Odps

MaxCompute SDK的入口，用户通过此类来获取项目空间下的所有对象集合，包括：Projects、Tables、Resources、Functions、Instances。

用户可以通过传入AliyunAccount实例来构造MaxCompute对象。程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Table t : odps.tables()) {
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.4 Projects

MaxCompute DPS中所有项目空间的集合。

集合中的元素为Project 。程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps  odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Project p = odps.projects().get("my_exists");
p.reload();
Map<String, String> properties = prj.getProperties();
...
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.5 Project

对项目空间信息的描述，可以通过Projects获取相应的项目空间。

1.6 SQLTask

用于运行、处理SQL任务的接口。可以通过run接口直接运行SQL。run接口返回Instance 实例，通过Instance获取SQL的运行状态及运行结果。

程序示例如下，仅供参考：

```
Account account = new AliyunAccount("my_access_id", "my_access_key"); Odps odps = new
Odps(account);
String  odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Instance instance = SQLTask.run(odps, "my_project", "select ...");
String id = instance.getId();
instance.waitforsuccess();
Set<String> taskNames = instance.getTaskNames();
for (String name
taskNames) {
TaskSummary summary = instance.getTaskSummary(name);
String s = summary.getSummaryText();
}
Map<String, String> results = instance.getTaskResults();
Map<String, TaskStatus> taskStatus = instance.getTaskStatus();
for (Entry<String, TaskStatus> status : taskStatus.entrySet()) {
String result = results.get(status.getKey());
}
```

说明：

如果用户想创建表，需要通过SQLTask接口，而不是Table 接口。用户需要将创建表（CREATE TABLE）的语句传入SQLTask。MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.7 Instances

MaxCompute中所有实例（Instance）的集合。

集合中的元素为Instance 。程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Instance i : odps.instances ()) {
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.8 Instance

对实例信息的描述，可以通过Instances获取相应的实例。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps (account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Instance ins = odps.instances().get("instance id");
Date startTime = instance.getStartTime();
Date endTime = instance.getEndTime();
...
Status instanceStatus = instance.getStatus();
String instanceStatusStr = null;
if (instanceStatus == Status.TERMINATED) {
instanceStatusStr = TaskStatus.Status.SUCCESS.toString();
Map<String, TaskStatus> taskStatus = instance.getTaskStatus();
for (Entry<String, TaskStatus> status : taskStatus.entrySet()) {
if (status.getValue().getStatus() != TaskStatus.Status.SUCCESS) {
instanceStatusStr = status.getValue().getStatus().toString();
break;
}
}
} else {
instanceStatusStr = instanceStatus.toString();
}
...
TaskSummary summary = instance.getTaskSummary("instance name");
```

```
String s = summary.getSummaryText();
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.9 Tables

MaxCompute中所有表的集合。集合中的元素为Table。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Table t : odps.tables()) {
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.10 Table

对表信息的描述，可以通过Tables获取相应的表。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Table t = odps.tables().get("table name");
t.reload();
Partition part = t.getPartition(new PartitionSpec(tableSpec[1]));
part.reload();
...
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.11 Resources

MaxCompute中所有资源的集合。集合中的元素为Resource。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Resource r : odps.resources()) {
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.12 Resource

对资源信息的描述，可以通过Resources获取相应的资源。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Resource r = odps.resources().get("resource name");
r.reload();
if (r.getType() == Resource.Type.TABLE) { TableResource tr = new TableResource(r);
String tableSource = tr.getSourceTable().getProject() + "." + tr.getSourceTable().getName();
if (tr.getSourceTablePartition() != null) {
tableSource += " partition(" + tr.getSourceTablePartition().toString() + ")";
}
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

一个创建文件资源的示例：

```
String projectName = "my_porject";
String source = "my_local_file.txt";
File file = new File(source);
InputStream is = new FileInputStream(file);
FileResource resource = new FileResource();
String name = file.getName();
resource.setName(name);
```

```
odps.resources().create(projectName, resource, is);
```

一个创建表资源的示例：

```
TableResource resource = new TableResource(tableName, tablePrj, partitionSpec);
//resource.setName(INVALID_USER_TABLE);
resource.setName("table_resource_name");
odps.resources().update(projectName, resource);
```

1.13 Functions

MaxCompute中所有函数的集合。集合中的元素为Function。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
odps.setDefaultProject("my_project");
for (Function f : odps.functions()) {
....
}
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

1.14 Function

对函数信息的描述，可以通过Functions获取相应的函数。

程序示例如下：

```
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
String odpsUrl = "<your odps endpoint>";
odps.setEndpoint(odpsUrl);
Function f = odps.functions().get("function name");
List<Resource> resources = f.getResources();
```

说明：

MaxCompute提供了两个服务地址供用户选择。详细说明请参考《*Tunnel SDK*简介》。

一个创建函数的示例：

```
String resources = "xxx:xxx";
String classType = "com.aliyun.odps.mapred.open.example.WordCount";
ArrayList<String> resourceList = new ArrayList<String>();
for (String r : resources.split(":")) {
resourceList.add(r);
}
```

```
Function func = new Function();
func.setName(name);
func.setClassType(classType);
func.setResources(resourceList);
odps.functions().create(projectName, func);
```

2 大数据开发套件

2.1 前言

关于本文档

本文档全面系统地给出了阿里云DataWorks中各相关对象的API定义，通过完整的参数说明和使用示例，帮助开发人员了解阿里云大数据开发套件的基本模型的使用方法。

阅读对象

本文档可作为开发人员在对阿里云DataWorks的模型和架构有基本了解的情况下，进行二次开发和数据收集的参考。

文档约定

本文档遵循如下约定：

- **排版约定**；

  下表主要描述了本手册中常用的排版约定：

字体格式或标志	**释义**
粗体	所有标题和功能点均使用**加粗**字体表示。
【表管理】	表示大数据集成服务平台中的具体功能模块。
create table	表示代码示例，如create table s_member_cart。
斜体	命令行参数（代码示例必须由实际值进行替代的部分）采用斜体表示。
【注意】	表示需要读者注意的事项。
【提示】	配置、操作或使用此平台的技巧。
注意事项内容	表示需要读者注意的具体事项说明。

- **符号约定。**

  下表主要描述了本手册中常用的符号约定：

符号	**描述**	**范例**	**释义**
>	表示在平台中的菜单选项	文件>新建>项目	从**文件**菜单中选择**新建**，然后从**新建**子菜单中选择**项目**。

->	表示某流程的先后顺序	申请->审批->赋权	此申请流程需先进行**申请**，再进行**审批**，最后**赋权**。

2.2 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，是用户管理表（Table）、资源（Resource）、自定义函数（UDF）、节点（Node）、权限等的基本单元。

2.3 产品简介

DataWorks是一种简单高效，图形化页面化构建和实现算法逻辑，实施数据处理的调试和定期运维的集成环境。

2.4 API概览

调度系统API

接口名称	接口功能
新建Flow API	在调度系统中创建Flow
更新Flow API	更新调度系统中的Flow信息
.....	

2.5 调用方式

2.5.1 请求结构

2.5.1.1 新建工作流任务

调用方式

```
POST http://baseapi.example.com/v2.0/flow/project/{projectIdentifier}
```

参数

```
{
   FlowCreateDto flowCreateDto;
}
```

FlowUpdateDto属性

```
 {
```

```
String flowName;
String flowDisplayName;
String owner;
String runType; //运行类型，取值: "0", "1", "2"
String scheduleExp; // 调度的cron表达式
Date startEffectDate; // 有效起始时间
Date endEffectDate; // 有效终止时间
String paraValue; // 运行参数，可被每个节点继承
String description; // 描述信息
Boolean isAuto;    //是否自动调度
NodeCreateDto rootNode;  //Flow的根节点
List<NodeCreateDto> includeNodes;//Flow包含的节点
List<String>  sameCycleParents;//依赖的正常周期Flow,格式:"不同 flow 名字用逗号分隔，若为不同project 下的 flow，格式为 ProjectName.FlowName"
String diffCycleDependentType;//跨周期依赖的依赖类型，0 - 不跨周期，1 - 用户自定义跨周期依赖， 2 - 跨一层子节点依赖， 3 - 自依赖
 List<String>  diffCycleParents;//跨周期依赖的Flow，格式:"不同 flow 名字用逗号分隔，若为不同 project 下的 flow，格式为 ProjectName.FlowName"
}
```

NodeCreateDto属性

```
 {
String nodeName; // 节点名称
String nodeDisplayName; // 节点展示名称
Integer type; // 执行代码类型
String runType;//运行类型
Integer dependentType; // 依赖类型，0 - 不跨周期，1 - 用户自定义跨周期依赖， 2 - 跨一层子节点依赖， 3 - 自依赖
String nodeCode;//节点对应的代码
String description; // 描述信息
String paraValue; // 参数信息
Date startEffectDate; // 允许调度的起始日期
Date endEffectDate; // 允许调度的终止日期
String owner; // 负责人账号
String source; // 来源
String connection; // 连接串
String scheduleExp; // 调度的cron表达式
List<String> parents;//上层节点
List<String> customDependentParents;//自定义依赖有父节点
}
```

返回结果

```
 {
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
Boolean returnValue; //执行成功返回true
```

```
}
```

2.5.1.2 更新工作流任务

调用方式

```
PUT  http://baseapi.example.com/v2.0/flow/project/{projectIdentifier}
```

参数

```
{
FlowUpdateDto flowUpdateDto;
}
```

FlowUpdateDto属性

```
{
String flowName;
String flowDisplayName;
String owner;
String runType; //运行类型，取值: "0", "1", "2"
String scheduleExp; // 调度的cron表达式
Date startEffectDate; // 有效起始时间
Date endEffectDate; // 有效终止时间
String paraValue; // 运行参数，可被每个节点继承
String description; // 描述信息
Boolean isAuto;    //是否自动调度
List<String>  sameCycleParents;//依赖的正常周期Flow,格式:"不同 flow 名字用逗号分隔，若为不同project 下的 flow，格式为 ProjectName.FlowName"
String diffCycleDependentType;//跨周期依赖的依赖类型，0 - 不跨周期，1 - 用户自定义跨周期依赖， 2 - 跨一层子节点依赖， 3 - 自依赖
List<String>  diffCycleParents;//跨周期依赖的Flow,格式:"不同 flow 名字用逗号分隔，若为不同project 下的 flow，格式为 ProjectName.FlowName"
}
```

返回结果

```
{
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
Boolean returnValue; //执行成功返回true
```

```
}
```

2.5.1.3 向上查询父节点

调用方式

```
GET  http://baseapi.example.com/v2.0/node/project/{projectIdentifier}/flow/{flowName}/node/{
nodeName}/parents
```

参数

```
{
String projectName; //url中的项目名
String flowName; //url中的flowName
String nodeName; //url中的节点名
Integer depth; //层数
}
```

返回结果

```
 {
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
List<NodeInfo> returnValue [{
    String nodeName; // 节点名称
    String nodeDisplayName; // 节点展示名称
    Integer runType; // 节点类型，参见RunType类
    Long fileId; // 代码编号
    Integer fileVersion; // 代码版本号
    String filePath; // 代码在压缩包的相对路径
    Integer nodeType; // 执行代码类型
    String description; // 描述信息
    String paraValue; // 参数信息
    Date startEffectDate; // 允许调度的起始日期
    Date endEffectDate; // 允许调度的终止日期
    String cronExpress; // cron表达式
    String owner; // 负责人账号
    String flowName;//流程名
    String projectName;//项目名
    Integer dependentType; // 依赖类型，0 - 不跨周期，1 - 用户自定义跨周期依赖， 2 - 跨一层
子节点依赖， 3 - 自依赖
    String source; // 来源
    String connection; // 连接串
    Date createTime; // 创建时间
    String createUser; // 创建人
    Date modifyTime; // 最新修改时间
    String modifyUser; // 最新修改人
    Integer priority; // 优先级
    String resourceGroupIdentifier; // 节点所属资源组标识
    Integer baselineId; // 节点所属基线编号
    Integer cycleType; // 周期类型，参见CycleType类
    List<NodeRelationInfo> parentRelations;//父节点信息
    List<NodeRelationInfo> childRelations;//子节点信息
```

```
},{},{}...]; //执行结果
}
```

2.5.1.4 向下查询子节点

调用方式

```
GET  http://baseapi.example.com/v2.0/node/project/{projectIdentifier}/flow/{flowName}/node/{
nodeName}/children
```

参数

```
 {
String projectName; //url中的项目名
String flowName; //url中的flowName
String nodeName; //url中的节点名
Integer depth; //层数
}
```

返回结果

```
{
   String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<NodeInfo> returnValue [{
   String nodeName; // 节点名称
   String nodeDisplayName; // 节点展示名称
   Integer runType; // 节点类型，参见RunType类
   Long fileId; // 代码编号
   Integer fileVersion; // 代码版本号
   String filePath; // 代码在压缩包的相对路径
   Integer nodeType; // 执行代码类型
   String description; // 描述信息
   String paraValue; // 参数信息
   Date startEffectDate; // 允许调度的起始日期
   Date endEffectDate; // 允许调度的终止日期
   String cronExpress; // cron表达式
   String owner; // 负责人账号
   String flowName;//流程名
   String projectName;//项目名
   Integer dependentType; // 依赖类型，参见DependentType类
   String source; // 来源
   String connection; // 连接串
   Date createTime; //建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer priority; // 优先级
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 节点所属基线编号
   Integer cycleType; // 周期类型，参见CycleType类
   List<NodeRelationInfo> parentRelations;//父节点信息
   List<NodeRelationInfo> childRelations;//子节点信息
```

```
},{},{}...]; //执行结果
}
```

2.5.1.5 查询节点的代码

调用方式

```
GET  http://baseapi.example.com/v2.0/node/project/{projectIdentifier}/flow/{flowName}/node/{
nodeName}/code
```

参数

```
 {
String projectName; //url中的项目名
String flowName; //url中的flowName
String nodeName; //url中的节点名
 }
```

返回结果

```
 {
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
String returnValue; //执行结果,返回节点代码
}
```

2.5.1.6 批量更新节点

调用方式

```
GET  http://baseapi.example.com/v2.0/node/update/multiple/config
```

参数

```
{
List<ReplaceNodeConfigDto> replaceDtos;
}
```

ReplaceNodeConfigDto参数

```
{
String projectName;
String flowName;
String nodeName;
Integer baselineId;
String owner;
Integer resgroupId;
```

```
}
```

返回结果

```
 {
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
Boolean returnValue; //执行成功返回true
}
```

2.5.1.7 分页查询节点

调用方式

```
GET  http://baseapi.example.com/v2.0/node
```

参数

```
{
NodeSearchDto condition;
 }
```

NodeSearchDto参数

```
{
String runTypes; // 节点运行类型,多个以逗号分隔
String nodeTypes; // 节点类型,多个以逗号分隔
String projectName;//精确匹配
String flowName; //精确匹配
String owner; // 节点负责人
String modifyTime;//节点发布日期,格式yyyy-mm-dd
String searchText;//节点名称,模糊查询,nodename,displayname
Boolean includeRelation;//返回的结果中,是否包含依赖关系
 }
```

返回结果

```
{
String requestId;//请求的id
String returnCode;//0表示调用成功
String returnMessage;//返回执行的详细信息
List<NodeInfo> returnValue; /
Integer count; //返回结果数
```

```
}
```

2.5.1.8 根据节点名字查询节点

调用方式

```
GET  http://baseapi.example.com/v2.0/project/{projectIdentifier}/flow/{flowName}/node/{
nodeName}
```

返回结果

```
{
  String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    NodeInfo returnValue; //执行成功返回true
 }

NodeInfo
{
    String nodeName; // 节点名称
    String nodeDisplayName; // 节点展示名称
    Integer runType; // 节点类型，参见RunType类
    Long fileId; // 代码编号
    Integer fileVersion; // 代码版本号
    String filePath; // 代码在压缩包的相对路径
    Integer nodeType; // 执行代码类型
    String description; // 描述信息
    String paraValue; // 参数信息
    Date startEffectDate; // 允许调度的起始日期
    Date endEffectDate; // 允许调度的终止日期
    String cronExpress; // cron表达式
    String owner; // 负责人账号
    String flowName;//流程名
    String projectName;//项目名
    Integer dependentType; // 依赖类型，参见DependentType类
    String source; // 来源
    String connection; // 连接串
    Date createTime; // 创建时间
    String createUser; // 创建人
    Date modifyTime; // 最新修改时间
    String modifyUser; // 最新修改人
    Integer priority; // 优先级
    String resourceGroupIdentifier; // 节点所属资源组标识
    Integer baselineId; // 节点所属基线编号
    Integer cycleType; // 周期类型，参见CycleType类
    Integer projectId;// 项目ID
    List<NodeRelationInfo> parentRelations //父节点信息
    List<NodeRelationInfo> childRelations;//子节点信息
    String dqcDescription ;
    Integer dqcType ;
    Boolean isFlowType;//true表节点是Flow类型,false是节点类型
}

NodeRelationInfo
{
```

```
String  projectName; //依赖的Node所属Flow的项目名
String  flowName; //依赖的Node所属Flow名
String  nodeName; //依赖的Node名
Integer relationType; // 依赖方式，参见DependentType类,只有普通，自定义
Integer projectId;// 项目ID
}
```

2.5.1.9 新建节点

调用方式

```
POST  http://baseapi.example.com/v2.0/project/{projectIdentifier}/flow/{flowName}
```

参数

```
{
NodeCreateDto createDto
}
```

NodeCreateDto参数

```
{
String nodeName; // 节点名称
String nodeDisplayName; // 节点展示名称
Integer type; // 执行代码类型
String runType;//运行类型
Integer dependentType; // 依赖类型，参见DependentType类
String nodeCode;//节点对应的代码
String description; // 描述信息
String paraValue; // 参数信息
Date startEffectDate; // 允许调度的起始日期
Date endEffectDate; // 允许调度的终止日期
String owner; // 负责人账号
String source; // 来源
String connection; // 连接串
String scheduleExp; // 调度的cron表达式
List<String> parents;//上层节点
List<String> customDependentParents;//自定义依赖有父节点
}
```

返回结果

```
{
   String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      NodeInfo returnValue; //执行成功返回true
 }

NodeInfo
{
   String nodeName; // 节点名称
   String nodeDisplayName; // 节点展示名称
   Integer runType; // 节点类型，参见RunType类
```

```
    Long fileId; // 代码编号
    Integer fileVersion; // 代码版本号
    String filePath; // 代码在压缩包的相对路径
    Integer nodeType; // 执行代码类型
    String description; // 描述信息
    String paraValue; // 参数信息
    Date startEffectDate; // 允许调度的起始日期
    Date endEffectDate; // 允许调度的终止日期
    String cronExpress; // cron表达式
    String owner; // 负责人账号
    String flowName;//流程名
    String projectName;//项目名
    Integer dependentType; // 依赖类型，参见DependentType类
    String source; // 来源
    String connection; // 连接串
    Date createTime; // 创建时间
    String createUser; // 创建人
    Date modifyTime; // 最新修改时间
    String modifyUser; // 最新修改人
    Integer priority; // 优先级
    String resourceGroupIdentifier; // 节点所属资源组标识
    Integer baselineId; // 节点所属基线编号
    Integer cycleType; // 周期类型，参见CycleType类
    Integer projectId;// 项目ID
    List<NodeRelationInfo> parentRelations //父节点信息
    List<NodeRelationInfo> childRelations;//子节点信息
    String dqcDescription ;
    Integer dqcType ;
    Boolean isFlowType;//true表节点是Flow类型,false是节点类型
}

NodeRelationInfo
{
    String  projectName; //依赖的Node所属Flow的项目名
    String  flowName; //依赖的Node所属Flow名
    String  nodeName; //依赖的Node名
    Integer relationType; // 依赖方式，参见DependentType类,只有普通，自定义
    Integer projectId;// 项目ID
  }
```

2.5.1.10 更新节点

调用方式

```
PUT  http://baseapi.example.com/v2.0/node/project/{projectIdentifier}/flow/{flowName}
```

参数

```
{
   NodeUpdateDto updateDto;
}
```

NodeUpdateDto参数

```
{
```

```
    String nodeName; // 节点名称
    String nodeDisplayName; // 节点展示名称
    Integer type; // 执行代码类型
    String runType;//运行类型
    Integer dependentType; // 依赖类型，参见DependentType类
    String nodeCode;//节点对应的代码
    String description; // 描述信息
    String paraValue; // 参数信息
    Date startEffectDate; // 允许调度的起始日期
    Date endEffectDate; // 允许调度的终止日期
    String owner; // 负责人账号
    String source; // 来源
    String connection; // 连接串
    String scheduleExp; // 调度的cron表达式
    List<String> parents;//上层节点,格式"projectName.flowName.nodeName"
    List<String> customDependentParents;//自定义依赖有父节点
}
```

返回结果

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    Boolean returnValue; //执行成功返回true
}
```

2.5.1.11 删除节点

调用方式

```
DELETE  http://baseapi.example.com/v2.0/node//project/{projectIdentifier}/flow/{flowName}/
node/{nodeName}
```

返回结果

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    Boolean returnValue; //执行成功返回true
}
```

2.5.1.12 根据BaseID删除节点

调用方式

```
DELETE  http://baseapi.example.com/v2.0/node/project/{projectIdentifier}/baseid/{baseid}
```

返回结果

```
{
    String requestId;//请求的id
```

```
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    Boolean returnValue; //执行成功返回true
}
```

2.5.1.13 查询实例运行日志

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/log
```

参数

```
{
    Long nodeInsId; //url中的instanceId
}
```

返回结果

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    String returnValue; //执行结果返回日志
 }
```

2.5.1.14 查询实例历史运行日志

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{historyIntanceId}/historyLog
```

参数

```
{
    Long historyInsId; //url中的historyIntanceId
}
```

返回结果

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    String returnValue; //执行结果返回日志
```

```
}
```

2.5.1.15 更新任务调度配置进入当前实例

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/refresh
```

参数

```
{
  Long instanceId; //url中的instanceId
}
```

返回结果

```
{
  String requestId;//请求的id
  String returnCode;//0表示调用成功
  String returnMessage;//返回执行的详细信息
  Boolean returnValue; //执行结果，成功返回true
}
```

2.5.1.16 终止实例运行

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/kill
```

参数

```
{
  Long instanceId; //url中的instanceId
}
```

返回结果

```
{
  String requestId;//请求的id
  String returnCode;//0表示调用成功
  String returnMessage;//返回执行的详细信息
  Boolean returnValue; //执行结果，成功返回true
```

```
}
```

2.5.1.17 把节点实例设置成功,并唤醒下游实例

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/setSuccess
```

参数

```
{
   Long instanceId; //url中的instanceId
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue; //执行结果，成功返回true
}
```

2.5.1.18 选择一批实例，从指定的实例开始，批量重跑

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/runMultiple
```

参数

```
{
   RunMultipleNodeInsDto runMultipleNodeInsDto;
}
```

RunMultipleNodeInsDto参数

```
{
   List<Long> includeNodeInsIds; //需要重跑的实例ID列表
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue; //执行结果，成功返回true
```

```
}
```

2.5.1.19 重新运行节点实例

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/rerun
```

参数

```
{
   Long instanceId; //url中的instanceId
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue; //执行结果，成功返回true
}
```

2.5.1.20 禁用节点实例

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/disable
```

参数

```
{
   Long instanceId; //url中的instanceId
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue; //执行结果，成功返回true
```

```
}
```

2.5.1.21 恢复指定的暂停的节点实例

调用方式

```
PUT  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/enable
```

参数

```
{
   Long instanceId; //url中的instanceId
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue; //执行结果，成功返回true
}
```

2.5.1.22 分页查询节点实例

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns
```

参数

```
{
   NodeInsSearchDto condition;
}
```

NodeInsSearchDto参数

```
{
   String projectName;
   String flowName;
   String searchText; // 节点名称或displayname
   String instanceIds; // 多个实例ID,以逗号分隔
   Integer dagType; // DAG类型
   String statuses;//多状态
   String instanceTypes;//任务类型：0： 正常 1：一次性任务 2:表示暂停的节点实例 3：空跑

   Integer nodeType;//节点类型
   String owner; // 负责人，统一使用工号
   String bizdate; // 业务日期,格式为yyyy-MM-dd HH:mi:ss
   String bizBeginHour;//格式为yyyy-MM-dd HH:mi:ss
   String bizEndHour;//格式为yyyy-MM-dd HH:mi:ss
   String createTime;//格式为yyyy-MM-dd HH:mi:ss
```

```
    Long dagId;   //dag实例 ID
}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<NodeInsInfo> returnValue [{
   Long historyId; // 历史实例ID
   Long instanceId; // 任务实例ID
   Long dagId; // 任务实例所属的流程实例ID
   Integer instanceType; // 任务类型,参见InstanceType枚举
   Integer dagType; // DAG类型
   Integer status; // 任务状态
   Long opSeq; // 操作序列号
   Integer opCode; // 操作命令
   String owner; // 负责人，统一使用工号
   Date bizdate; // 业务日期
   Date gmtdate; // 处理日期
   Integer nodeType; // 执行代码类型，如odps_sql等
   Integer priority; // 优先级
   String paraValue; // 参数信息
   String projectName; // 任务实例所属应用ID
   Long relatedDagId; // 任务关联的工作流实例编号
   Date finishTime; // 任务结束时间
   Date beginWaitTimeTime; // 变成等待时间状态的时间
   Date beginWaitResTime; // 变成等待资源状态的时间
   Date beginRunningTime; // 变成运行中状态的时间
   Date createTime; // 创建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer cycleType; // 周期类型
   Date cycleTime; // 周期时间
   Integer dependentType; // 依赖关系类型
   String nodeName; // 节点名称
   String flowName; // 根节点所属工作流定义
   Long inGroupId; // 该周期是当天的第几个周期
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 任务所属基线编号
      },{},{}...]; //执行结果
      Integer count; //查询返回结果数
```

```
}
```

2.5.1.23 根据节点实例ID查询

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{instanceId}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      returnValue {
   Long historyId; // 历史实例ID
   Long instanceId; // 任务实例ID
   Long dagId; // 任务实例所属的流程实例ID
   Integer instanceType; // 任务类型,参见InstanceType枚举
   Integer dagType; // DAG类型
   Integer status; // 任务状态
   Long opSeq; // 操作序列号
   Integer opCode; // 操作命令
   String owner; // 负责人，统一使用工号
   Date bizdate; // 业务日期
   Date gmtdate; // 处理日期
   Integer nodeType; // 执行代码类型，如odps_sql等
   Integer priority; // 优先级
   String paraValue; // 参数信息
   String projectName; // 任务实例所属应用ID
   Long relatedDagId; // 任务关联的工作流实例编号
   Date finishTime; // 任务结束时间
   Date beginWaitTimeTime; // 变成等待时间状态的时间
   Date beginWaitResTime; // 变成等待资源状态的时间
   Date beginRunningTime; // 变成运行中状态的时间
   Date createTime; // 创建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer cycleType; // 周期类型
   Date cycleTime; // 周期时间
   Integer dependentType; // 依赖关系类型
   String nodeName; // 节点名称
   String flowName; // 根节点所属工作流定义
   Long inGroupId; // 该周期是当天的第几个周期
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 任务所属基线编号
}; //执行结果
      Integer count; //查询返回结果数
```

```
}
```

2.5.1.24 按层数查询父节点实例（包含工作流任务和节点）

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/parents?depth={depth}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<NodeInsInfo> returnValue [{
   Long historyId; // 历史实例ID
   Long instanceId; // 任务实例ID
   Long dagId; // 任务实例所属的流程实例ID
   Integer instanceType; // 任务类型,参见InstanceType枚举
   Integer dagType; // DAG类型
   Integer status; // 任务状态
   Long opSeq; // 操作序列号
   Integer opCode; // 操作命令
   String owner; // 负责人，统一使用工号
   Date bizdate; // 业务日期
   Date gmtdate; // 处理日期
   Integer nodeType; // 执行代码类型，如odps_sql等
   Integer priority; // 优先级
   String paraValue; // 参数信息
   String projectName; // 任务实例所属应用ID
   Long relatedDagId; // 任务关联的工作流实例编号
   Date finishTime; // 任务结束时间
   Date beginWaitTimeTime; // 变成等待时间状态的时间
   Date beginWaitResTime; // 变成等待资源状态的时间
   Date beginRunningTime; // 变成运行中状态的时间
   Date createTime; // 创建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer cycleType; // 周期类型
   Date cycleTime; // 周期时间
   Integer dependentType; // 依赖关系类型
   String nodeName; // 节点名称
   String flowName; // 根节点所属工作流定义
   Long inGroupId; // 该周期是当天的第几个周期
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 任务所属基线编号
     },{}..{}]; //执行结果
      Integer count; //查询返回结果数
```

```
}
```

2.5.1.25 按层数查询子节点实例（包含工作流任务和节点）

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/children?depth={depth}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<NodeInsInfo> returnValue [{
   Long historyId; // 历史实例ID
   Long instanceId; // 任务实例ID
   Long dagId; // 任务实例所属的流程实例ID
   Integer instanceType; // 任务类型,参见InstanceType枚举
   Integer dagType; // DAG类型
   Integer status; // 任务状态
   Long opSeq; // 操作序列号
   Integer opCode; // 操作命令
   String owner; // 负责人，统一使用工号
   Date bizdate; // 业务日期
   Date gmtdate; // 处理日期
   Integer nodeType; // 执行代码类型，如odps_sql等
   Integer priority; // 优先级
   String paraValue; // 参数信息
   String projectName; // 任务实例所属应用ID
   Long relatedDagId; // 任务关联的工作流实例编号
   Date finishTime; // 任务结束时间
   Date beginWaitTimeTime; // 变成等待时间状态的时间
   Date beginWaitResTime; // 变成等待资源状态的时间
   Date beginRunningTime; // 变成运行中状态的时间
   Date createTime; // 创建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer cycleType; // 周期类型
   Date cycleTime; // 周期时间
   Integer dependentType; // 依赖关系类型
   String nodeName; // 节点名称
   String flowName; // 根节点所属工作流定义
   Long inGroupId; // 该周期是当天的第几个周期
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 任务所属基线编号
     },{}..{}]; //执行结果
      Integer count; //查询返回结果数
```

```
}
```

2.5.1.26 查询节点的历史实例信息

调用方式

```
GET  http://baseapi.example.com/v2.0/nodeIns/{instanceId}/history?start={start}&limit={limit}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<NodeInsInfo> returnValue [{
   Long historyId; // 历史实例ID
   Long instanceId; // 任务实例ID
   Long dagId; // 任务实例所属的流程实例ID
   Integer instanceType; // 任务类型,参见InstanceType枚举
   Integer dagType; // DAG类型
   Integer status; // 任务状态
   Long opSeq; // 操作序列号
   Integer opCode; // 操作命令
   String owner; // 负责人，统一使用工号
   Date bizdate; // 业务日期
   Date gmtdate; // 处理日期
   Integer nodeType; // 执行代码类型，如odps_sql等
   Integer priority; // 优先级
   String paraValue; // 参数信息
   String projectName; // 任务实例所属应用ID
   Long relatedDagId; // 任务关联的工作流实例编号
   Date finishTime; // 任务结束时间
   Date beginWaitTimeTime; // 变成等待时间状态的时间
   Date beginWaitResTime; // 变成等待资源状态的时间
   Date beginRunningTime; // 变成运行中状态的时间
   Date createTime; // 创建时间
   String createUser; // 创建人
   Date modifyTime; // 最新修改时间
   String modifyUser; // 最新修改人
   Integer cycleType; // 周期类型
   Date cycleTime; // 周期时间
   Integer dependentType; // 依赖关系类型
   String nodeName; // 节点名称
   String flowName; // 根节点所属工作流定义
   Long inGroupId; // 该周期是当天的第几个周期
   String resourceGroupIdentifier; // 节点所属资源组标识
   Integer baselineId; // 任务所属基线编号
     },{}..{}]; //执行结果
      Integer count; //查询返回结果数
```

```
}
```

2.5.1.27 以补数据方式调起一个工作流任务中的一批节点

调用方式

```
POST  http://baseapi.example.com/v2.0/dag/single/flow/multiple/node/manual
```

参数

```
{
   CreateMultipleNodeInsDto dto;
}
```

CreateMultipleNodeInsDto属性

```
{
      String instanceName; // 工作流实例名称,非空
      String projectName;//项目名，非空
      String flowName; //工作流名称，非空
      Date startDayBizdate; //业务起始时间,天任务使用，非空
      Date endDayBizdate; //业务起始时间,天任务使用，非空
      String startHourBizdate; ///小时任务的业务时间,配合startDayBizdate,endDayBizdate一起使
用
      String endHourBizdate; ///小时任务的业务时间
      String startNodeName;//本批次调起的节点,从哪个节点开始运行
      List<NodeDto> nodeList;//批量调起的节点，非空
}
```

返回结果

```
{
      String requestId;//请求的id
      String returnCode;//0表示调用成功
      String returnMessage;//返回执行的详细信息
      List<Long> returnValue;//执行成功返回调起的节点id列表
}
```

2.5.1.28 以冒烟测试的方式调起一个工作流任务中的一个节点

调用方式

```
POST  http://baseapi.example.com/v2.0/dag/single/flow/multiple/node/smoke
```

参数

```
{
   SmokeNodeInsDto dto;
```

```
}
```

SmokeNodeInsDto属性

```
{
   String instanceName; // 工作流实例名称，非空
   String projectName;//项目名，非空
   String flowName;//工作流名称,startFlowId表示本批次从哪个Flow开始触发
   String nodeName; //节点名称
   Date bizdate; //业务起始时间,天任务使用
   String startHourBizdate; ///小时任务的业务时间,配合startDayBizdate,endDayBizdate一起使用

   String endHourBizdate; ///小时任务的业务时间
}
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Long returnValue;//执行成功返回调起的节点id
}
```

2.5.1.29 终止DAG进程

调用方式

```
POST  http://baseapi.example.com/v2.0/dag/{dagId}/kill
```

返回结果

```
{
   String requestId;//请求的id
   String returnCode;//0表示调用成功
   String returnMessage;//返回执行的详细信息
   Boolean returnValue；//成功返回true
}
```

2.5.1.30 提交ZIP包

调用方式

```
POST  http://baseapi.example.com/v2.0/submission/project/{projectIdentifier}
```

参数

```
 {
      InputStream is;        //zip 文件对应流
      FormDataContentDisposition fdcd;
```

```
}
```

返回结果

```
{
    String requestId;//请求的id，本接口为异步提交，需要根据请求id查询提交状态
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    Boolean returnValue;//添加成功返回true
}
```

2.5.1.31 查询ZIP包提交状态

调用方式

```
POST  http://baseapi.example.com/v2.0/submission/project/{projectIdentifier}/{requestId}
```

返回结果

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    AsyncSubmissionLogInfo returnValue [{
  String requestId;
  Date createTime;//接受提交请求时间
  String name;//生成的名称标识,格式为[调用方的系统名-backToCheckId-createTime]
  String status;//success：执行成功，fail:执行失败,running：正在执行
  String message;      }];
}
```

2.5.2 接口鉴权

Header

```
base_id(必填)
tenant_id(必填)
local_name(可选，默认：zh_CN)
```

安全

调用API的接口，都需要经过签名， 调用方需做以下操作。

1. 在url中增加2个固定字段 ：baseKey，timestamp。
2. 生成签名，加到http header中，名为signature。
   - 以GET方法为例：

     假设B提供了http接口为 /getapi，需要的query parameter分别为name，age。A原有的调用url应该是/getapi?name=fengyeng&age=20，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/getapi?name=fengyeng&age=20?baseKey=dsa134&timestamp=123456789， 随后利用新的url中的query parameter结合token生成signature， 再在header中增加一个变量signature:dsafadfsa141fdsaf1 。

- 以POST方法为例：

  假设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&timestamp=123456789，随后利用新的url中的query parameter结合token生成signature，再在header中增加一个变量signature:dsafadfsa141fdsaf1。

2.5.3 返回值

返回结构为JSON格式，包含内容统一为requestId, requestCode, requestMessage, returnValue。如果接口调用失败，则根据requestCode/requestMessage可获知基本报错信息；根据requestId可在系统所有后台日志中追溯接口调用情况，获取详细的调用信息。如果接口调用成功则requestCode为0，如有返回值则作为returnValue返回。

```
{
    String requestId;//请求的id
    String returnCode;//0表示调用成功
    String returnMessage;//返回执行的详细信息
    Object returnValue;
}
```

3 大数据应用加速器

3.1 前言

概述

本文档全面系统地给出了阿里云大数据应用加速器DTBoost中各相关对象的API定义，通过完整的参数说明和使用示例，帮助开发人员了解阿里云数据应用加速器可编程接口的使用。

阅读对象

本文档可作为开发人员在对阿里云大数据应用加速器DTBoost的模型和架构有基本了解的情况下，进行二次开发和数据收集的参考手册。

3.2 产品简介

大数据应用加速器（DTBoost）是阿里云结合阿里巴巴自身大数据应用场景，经过多年总结抽象出的企业级大数据应用平台，它的目标如下所示。

- 让业务人员可以快速的理解数据，应用数据。
- 数据模型设计，重计算模型设计。
- 结构开放，快速支撑数据应用开发。
- 支撑企业内部共建共享，协同开发。

大数据应用加速器除了提供UI交互界面之外，还有一组强大API接口，可以方便的完成系统集成，大大降低应用开发成本。

3.3 API调用方式

3.3.1 接口访问

接口的访问地址为：http://{产品前缀}.example.com/${apipath}

以下*API列表*中列出的API地址，都是 ${apipath} 部分。

3.3.2 接口鉴权

接口的鉴权形式采用私有秘钥签名的形式。

签名算法

获取签名所需的信息。

$HTTP_METHOD

大写，如： POST，GET

$URL

服务器收到的请求path + querystring， querystring 中必须包含 timestamp 参数，签名校验会通过该字段检查签名的时效性。

说明：

- timestamp 为unix-timestamp * 1000 ，即毫秒单位。
- querystring需要urldecode。

$BODY

http请求的body部分。

计算签名

每个租户都有自己的一个 $secretToken，这个token可以在DTBoost的**控制台** > **个人中心**页面找到，如图 *3-1:* 个人中心所示。

图 3-1: 个人中心

签名的方式有两种：

- 带body的场景，如post请求

```
$HTTP_METHOD + ‘\n’ +
$URL + ‘\n’ +
```

```
md5($BODY)
```

- 不带body的场景，如get请求

```
$HTTP_METHOD + ‘\n’ +
$URL + ‘\n’ +
```

签名的hash算法为： hmac sha-256，secretkey 为前面提到的$secretToken。

Node.js 版本的签名代码

```
const crypto = require('crypto');
const urllib = require('urllib');
function hmacsha256(str, secret) {
  const hash = crypto.createHmac('sha256', secret);
  hash.update(str, 'utf8');
  return hash.digest('hex');
}
let token = '您的token信息'; // token框架会提供
let timestamp = Date.now();
let str =  'GET\n/otm/api/categories/download?category_id=10001&timestamp='+timestamp;
let signature = hmacsha256(str, token);
let options = {
  headers: {
    signature: signature
  }
};
let url = 'http://$domain/otm/api/categories/download?category_id=10001&timestamp=
1461502358436'
urllib.request(url, options, function (err, result) {
  if (err) {}
  ...
});
```

3.3.3 接口版本控制

当一个接口的 ${apipath} 部分设计了类似 /api/v1/xxxx 结构， 则说明接口存在版本区分，注意查看不同版本的接口说明，并选择合适的版本调用。

当接口路径不包含版本号时，说明该接口始终保持相同的调用方式。

3.3.4 返回格式

请求接口的时候，可能有如下httpCode返回：

- 504：请求超时，请联系阿里云技术支持获得帮助。
- 500：接口异常，请联系阿里云技术支持获得帮助。
- 403：没有权限，可能是资源访问权限受限，请先申请查询对象的访问权限。
- 200：接口请求成功。

当接口httpCode为**200** 的时候，接口返回为一个JSON格式的数据， 格式包含如下：

```
{
"code" :  "SUCCESS",   // {String} 请求返回状态码，字符串类型，大写字母
"data": [],        // {Array} 请求成功之后获得的返回数据
"message": null   // {null|String} 如有异常，返回执行异常时的详细信息
}
```

3.4 API列表

3.4.1 标签中心API

主要是标签数据更新回调接口。

调用方式

```
POST  http://dtboost.example.com/analyse/api/query
```

参数

```
 {
"tql": "select * from object",  // {String} 查询的tql
"extra": {
 "schema_code": "ads_test" , // {String} 云计算资源名称
 "entity": "user" // {String} 实体名称
},
"page_size":  10,  // {Int} [可选]，分页大小，默认不分页
"page_no": 2     // {Int} [可选]页码
 }
```

返回结果

```
 {
"code": "SUCCESS",   // {String} 返回code
"data": [{} ,  {} ,  {}],    // {Array} TQL查询返回的数据集合
"message": null    // {null|String} 如果异常，则返回详细的错误信息String，如果异常来自于计算引擎，请查阅计算引擎的错误码参考资料
}
```

说明：

Code 可以为：

- SUCCESS 查询成功。
- ERROR 查询失败，失败原因在 message中。

3.4.2 整合分析API

3.4.2.1 TQL查询接口

调用方式

POST http://dtboost.example.com/analyse/api/query

参数

```
 {
"tql": "select * from object",  // {String} 查询的tql
"extra": {
 "schema_code": "ads_test" , // {String} 云计算资源名称
 "entity": "user" // {String} 实体名称
},
"page_size":  10,  // {Int} [可选]，分页大小，默认不分页
"page_no": 2     // {Int} [可选]页码
 }
```

返回结果

```
 {
"code": "SUCCESS",   // {String} 返回code
"data": [{} ,  {} ,  {}],    // {Array} TQL查询返回的数据集合
"message": null    // {null|String} 如果异常，则返回详细的错误信息 String，如果异常来自于计算引擎，请查阅计算引擎的错误码参考资料
}
```

说明：

Code 可以为：

- SUCCESS 查询成功。
- ERROR 查询失败，失败原因在 message中。

3.4.2.2 Expr查询接口

调用方式

```
POST  http://dtboost.example.com/analyse/api/query/json
```

参数

```
{
"expr":  {},  // {Object} url中的项目名
"extra": {
 "schema_code": "ads_test", // {String} 云计算资源名称
 "entity": "user"            // {String} 实体名称
},
```

```
"page_size": 10   // {Int} [可选] 分页大小，默认不分页
"page_no": 2      // {Int} [可选] 第几页
}
```

返回结果

```
{
"code": "SUCCESS",        // {String} 返回code
"data": [{} ， {} ， {}],   // {Array} TQL查询返回的数据集合
"message":  null          // {null|String} 如异常，则返回详细的异常信息，如计算引擎异常，请查阅对应计算引擎的异常码手册
}
```

该接口返回数据和 TQL接口一致。

3.4.2.3 JQL查询接口

调用方式

```
POST  http://dtboost.example.com/analyse/api/query/json
```

参数

```
{
"json":  {}, //url中的项目名
"extra": {
 "schema_code": "ads_test", // {String} 云计算资源名称
 "entity": "user"            // {String} 实体名称
},
"page_size": 10,   // {Int} [可选] 分页大小，默认不分页
"page_no": 2       // {Int} [可选] 第几页
}
```

返回结果

```
{
"code": "SUCCESS", // {String} 返回code
"data": [{} ， {} ， {}], // {Array} TQL查询返回的数据集合
"message": null // {null|String} 如异常，则返回详细的异常信息，如异常由于计算引擎引擎，请查阅相应计算引擎的错误码手册
}
```

该接口返回数据和 TQL 接口一致。

3.4.2.4 TQL语法帮助

TQL（Tag Query Language）整合分析提供的基于OLT的标签查询语言，实现基于标签的分析查询。它对用户屏蔽了物理模型和物理存储，用户更容易进行分析查询。

TQL的语法定义如下：

```
SELECT [distinct] tag_expr_list FROM ol_reference [WHERE where_condition] [GROUP BY tag_expr_list] [ORDER BY order_condition] [LIMIT num]
```

简单查询

- 查询某个用户的性别：

```
select user.gender from user where user.uid = xxx;
```

- 计算所有类目下的成交笔数：

```
select trade.cateid,count(*) as cnt from trade group by trade.cateid;
```

使用group by...having

```
select sum(user.age) as s, user.location from user group by location having s > 1000;
```

Inner Join

```
select user.age,count(*)  from user join trade on user.uid=trade.user.uid  where trade.cateid= xxx;
```

Left Join/Right Join

```
select trade.cateid, user.age from user left join trade on (user.uid=trade.user.uid)   where trade.date > '2015-09-07';
```

子查询

```
select
    a.uid, a. amt_pay,
b.collect_cnt
from (
 select
  sum(trade.amt_pay) as amt_pay,
  trade.user.uid as uid
 from
  trade
 group by trade.user.uid
) as a left join (
 select
  count(*) as collect_cnt,
  collect.user.uid as uid
 from
                collect
 group by collect.date
```

```
) as b on (a.uid =b.uid);
```

Function

TQL 对于函数的支持依赖于所使用的执行引擎，即查询执行的数据库系统除此之外，TQL还支持如下函数：

```
* udfprofile 计算百分比sql
```

```
 select user.age_level, udf_profile(user.uid) from user group by        user.age_level;
```

```
udfsegment 分段函数
```

```
select udf_segment(user.age, '1:[0,18];2:(18,28);3:(28,50);4') as age_level from user;等价于sql
 select case when user.age >= 0 and user.age <= 18 then 1 when user.age > 18 and user.age
 <= 28 then 2 when user.age > 28 and user.age <= 50 then 3 else 4 end as age_level from user
;
```

3.4.2.5 Expr语法帮助

查询表达式的结构

表达式是一个JSON结构，共有6个元素：return，target，variable，filter，limit，order by（不区分大小写和先后顺序）。

```
{
   "RETURN":   //数组格式，必选项，申明查询的返回结果
   "TARGET":   //数组格式，必选项，申明分析的对象
   "VARIABLE": //可选，json 格式，子查询的逻辑在此表达，子查询返回的还是Object、Link
"FILTER":   //可选，字符串格式，过滤条件
"limit":    //可选，字符串格式， limit
"orderby"  //可选，字符串格式，排序
}
```

表达式可以嵌套

variable对应于SQL里的子查询，可以嵌套表达。

```
// 这个例子里，$object1、$object2 都来自嵌套的子查询
{
   "TARGET": ["object->$object1->$object2"]
   "RETURN": ["object.tag", "object1.tag1", "object2.tag2"]
   "VARIABLE": {
"object1":{
   "TARGET":["user"],
   "RETURN":["user.tag1"]
},
"object2":{
    "TARGET":["User2->$ojbect3"]  //object2再次嵌套object3
    "RETURN":["User2.tag2","object3.tag3"]
    "VARIABLE":{
        "object3":{
```

```
          "TARGET":["User3"]
          "RETURN":["User3.tag3"]
        }
    }
}
  }
}
```

RETURN：期望返回的结果

其中：

- RETURN里可以加别名。
- 子查询里RETURN的必须加别名。

```
{
  "TARGET":["User2->$ojbect3"]  //object2再次嵌套object3
  "RETURN":["User2.tag2","object3.tag3"] //object3的tag3来自 User3.tag3
  "VARIABLE":{
"object3":{
    "TARGET":["User3"]
        "RETURN":["User3.tag3 as tag3"]
}
  }
}
```

TARGET

TARGET分两类，共三种：object，link，子查询。

TARGET中申明对象的时候可以加过滤条件，取子集：

- 十二月份的成交：trade(month=201512)
- 女装类目的商品： auction(cat_name='女装')
- 上海的女性用户：user(gender='1' and city='shanghai')

TARGET之间的join关系：

- join的类型
  - inner join
  - outer join，且只有left outer join，如需right join，可以通过调整join的方向来转换。
- left join的方向
  - user->trade => user left join trade
  - trade->user => trade left join user
- join的连接点joinKey

- object/link和子查询join，必须指定joinKey。
- link和link之间的join，必须指定joinKey。
- 其他情况，由系统根据OTM的元数据自动决定joinKey。

  Join Cases: :

  - Join Case 1：Object join

    Object 根据外键做join，不需要指定joinKey，由系统根据otm元数据自动决定，前提是注册标签的时候指定了外键。

    ```
    target:["auction-shop"]  // 商品join店铺，auction上有一个外键shop_id(商品所属的店铺)指向shop
    target:["设备->供应商"],  // 设备join供应商，通过设备上的外键(供应商ID) 来和供应商来join  统计每个店铺的商品数量
    {
      "target":["auction-shop"],
       "return":["shop.name", "count(auction.id) as aucs"]
    }
    ```

  - Join Case 2：Object join

    Link 根据外键做join，不需要指定joinKey，由系统根据otm元数据自动决定。

    ```
    target:["auction-trade"] //商品join交易
    target:["auction-trade->shop"] //商品join交易，交易join店铺 target:["auction-trade", "auction-shop"] //商品join交易，商品再join店铺
    ```

  - Join Case 3：Object join

    子查询和子查询进行join，必须要指定 joinKey。

    分析对象为5月份成交大于500的店铺，期望返回结果为统计每个店铺的商品数量，auction 和shop1 这个子查询做join，子查询必须指定joinKey 。

    ```
    {
      "target":["auction-$shop1[shop_id]"],
      "return":["shop1.shopname", "count(auction.id) as aucs"],
      "variable":{
    "shop1":{
       "target":["shop-trade(month=201605)"],
       "return":[
     "shop.shop_id as shop_id",
     "shop.name as shopname",
     "sum(trade.gmv) as month5_total_gmv"
       ],
       "filter":"sum(trade.gmv)>500"
     }
      }
    }
    ```

  - Join Case 4: Link和子查询

Link和子查询之间的join，必须要指定joinKey

```
"target":["trade[user.uid,shop.shopid] -> $search1[user_id, shop_id]"]
```

相当于 TQL：

```
SELECE XX FROM trade FROM (search子查询) as search1 on(trade.user.uid=
search1.user_id     and trade.shop.shopid=search1.shopid )
```

- Join Case 5：子查询和子查询join

  子查询之间的join，必须要指定joinKey

  ```
  "target":["$trade1[uid,shopid] -> $search1[user_id, sellerid]"]
  ```

  相当于 TQL：

  ```
  SELECT XX FROM (trade子查询) as trade1 FROM (search子查询) as search1 on(
  trade1.uid=search1.user_id     and trade1.shopid=search1.sellerid  )
  ```

- Join Case 6：Link和Link

  Link之间的join，必须要指定joinKey

  ```
  "target":["trade[user.uid,shop.shopid] -> search[user.uid,shop.shopid]"]
  ```

VARIABLE详解

分析表达式通过variable来承载嵌套的逻辑，可以理解为SQL中的子查询。

- 子查询的return元素，必须加别名，如： user.user*id as user*id。
- target元素里，引用子查询的对象是必须要加$，如：$user1，其他地方引用子查询对象时不能加$。

```
{
  "TARGET":["$shop1"]   //子查询的对象名要加$,
  "RETURN": ["shop1.star", "shop1.id"]  //其他地方引用子查询时不需要加$,
  "FILTER":"shop1.star>1"     ,
  "ORDER BY":"shop1.star"     ,
  "VARIABLE":{
"shop1":{
  "RETURN":["shop.star","sum(trade.amt) as amt"]
  "TARGET":["shop-trade"]
}
  }
}
```

子查询的表达：

- 子查询Case 1：

分析对象：店铺的成交，按店铺星级统计平均成交金额。

子查询：在子查询里汇总店铺的总金额， 最外面求金额的平均值。

```
{
  "RETURN": ["shop.star", "avg(shop.amt) as avg_amt"]
  "TARGET":["$shop"] //子查询的对象要加
  "VARIABLE":{
      shop:{
        "RETURN":["shop.star","sum(trade.amt) as amt"]
        "TARGET":["shop-trade"]
    }
  }
}
```

等价的 TQL：

```
select
  shop.star,
  avg(shop.totoal_amt)
from (     //子查询，算出每个店铺的总金额
  select
shop.id,
shop.star,
sum(trade.amt) as amt
  from shop join trade
on(shop.shop_id=trade.shop.shop_id)
  group by shop.id,shop.star
) shop
group by shop.star
```

- 子查询Case 2：

  分析目标：在线商品数大于10 并且月成交大于10000 的店铺。

  分析结果：各个星级的店铺数。

  子查询：

  1. 店铺的在线商品数：店铺join商品。
  2. 店铺的当月成交金额：店铺join成交。

```
{
  "RETURN":["shop.star","count(shop.shopid) as shops"]
  "TARGET":[
"shop-$shop_join_auctions[shopid]->$shop_join_trade[shopid]"
  ]
  "VARIABLE":{
    "shop_join_auction":{
      "RETURN":[
        "shop.shopid as shopid",
        "shop.star as star",
        "count(auction.id) as items"
      ],
      "TARGET":["shop-auction(is_online='true')"]
```

```
    },
    "shop_join_trade":{
      "RETURN":[ "shop.shop_id as shopid", "sum(trade.amt) as total_amt" ], "TARGET":["
shop-trade"],
      "FILTER":"trade.month='201601'"
    }
  }
  "FILTER":"shop_join_auctions.items > 10 and shop_join_trade.total_amt > 10000"
}
```

等价的 TQL：

```
select
  shop.star,
  count(shop.id) as shops
from shop join(
  select
    shop.shop_id,
    count(auction.id) as items
  from shop join auction
  where
    auction.is_online="true"
  group by shop.shop_id
)shop_join_auc on (shop.shopid= shop_join_auc.shopid)
join(
  select
    shop.shopid,
    sum(trade.amt) as total_amt
  from shop join trade
  group by shop
) shop_join_trade on(shop.shopid= shop_join_trade.shopid)
where
  shop_join_auc.items>10 and
  shop_join_trade.total_amt>10000
group by star
```

3.5 附：术语与缩略语

3.5.1 基本术语

实体（Object）

即客观世界的一个对象，如人员、车辆、飞机、火车、酒店等。

关系（Link）

描述实体和实体之间发生的关联。如用户实体购买了商品实体，购买记录表就是一个 Link，它连接了用户实体和商品实体。

标签（Tag）

标签是实体的属性，如人员实体的标签可以有：性别、年龄等。

3.5.2 缩略词

TQL

标签查询语言（Tag Query Language）是分析接口支持的一种查询语言，基于实体、关系、标签的一种类似SQL的查询语法。其语法和SQL中的select语法非常相识，但不支持update、insert、delete等其他语法。

JQL

结构化标签查询语言（JSON Query Language）是分析接口支持查询输入语言，基于实体、关系、标签的描述，JSON格式的语法。其语法表达的含义和TQL相同。

4 关系网络分析

4.1 前言

概述

本文档详细探讨了阿里云 关系网络分析(Graph Analytics) 支持的 API 操作和相关的示例。

应用范围

使用本产品前，用户需满足必要的技能要求。推荐云产品开发工程师使用本文档。

4.2 简介

欢迎使用阿里云关系网络分析服务（ Graph Analytics Service ），简称（ I+ Service ）。用户可以使用本文档介绍的API对I+服务进行相关操作。

请确保在使用这些接口前，已充分了解了I+产品说明、使用协议和收费方式。

4.2.1 术语表

术语	全称	中文	说明
object		实体	
property		属性	
link		关系	
objectType		实体类型	对应后台管理配置的O编码，例O0001。
linkType		关系类型	对应后台管理配置的L编码，例L0001。
propertyId/propId		属性ID	对应实体与关系的属性编码例：O0001P001，L0001P001。
propertyType		属性查询类型	枚举： string_equel， 条件全词匹配。 string_like，条件模糊匹配。 number，数值查询。

术语	全称	中文	说明
			time，date，时间格式。 dict_select，全词匹配。
propertyList		基本属性条件列表	
derivated		间接关系条件列表	经过一定逻辑组装的属性条件，因编码暂未开放，现在可以使用后台默认配置。
statPropList		累计属性条件列表	经过sum，count计算的累计属性条件。
degree		度数	与某实体有直接联系的关系叫一度关系，两个点中经过一个间接点叫两度，经过两个间接点叫三度，以此类推。
direct/d		关系方向	关系的数据流向，如A给B打电话，则方向为A->B。

4.2.2 业务限制资源规格限制说明

在接口说明部分，凡出现对参数可选值、可用规格方面与官网上给出的资源规格限制发生矛盾时，均以官网上给出的值为准。

4.2.3 其他说明

关系网络中非常重要的过滤条件格式说明：

```
{
        "propertyId":"L0001P001",
        "propertyType":"string_equal",
        "values":[""]
    }
```

所有时间格式的输入包括date/time，都只支持10位时间戳格式，getTime()/1000。

场景一

propertyType=string_equal/string_like，values支持多个值输入，多个值为“或”。

场景二

propertyType=time/date/number，values为两个值的数组。例如：

- [10,20] 表示大于等于10且小于等于20。
- [10,10] 表示等于10。
- [10,*] 表示大于等于10。
- [*,20] 表示小于等于20。

场景三

propertyType=dict_select，values支持多个码值输入，多个值间关系为“或”。

4.3 更新历史

发布时间	更新	说明
2017-05-11	新增I+产品V1.8版本提供的API服务。	接口包括：实体查询服务接口、关联反查服务接口、默认关联反查服务接口、群集分析服务接口、共同邻居服务接口、路径分析服务接口。
2017-08-10	• 调整发布的接口。 • 增加鉴权。 • 增加流量控制。 • 统一了入参格式，增加接口参数说明。 • 增加OLP转义支持。	增加搜索及数据持久化相关的接口，修改了默认关联反查，关联反查，群集分析，共同邻居，路径分析，血缘分析接口。

4.4 API概览

分类列举各个接口。

4.4.1 搜索服务API

API	描述
/rest/search/queryNodes.json 详见5.实体查询服务	按条件查询节点信息。
/rest/search/queryLinks.json 详见6.关系查询服务	按条件查询关系信息。

4.4.2 关系网络服务API

API	描述
/rest/graph/getFastGraph.json 详见7.默认关联反查服务	一键多度查询：单个节点出发，扩展指定度数的关系网络路径。
/rest/graph/getGraph.json 详见8.关联反查服务	关联反查：单个或多个节点出发，获取节点关系网络路径。
/rest/graph/getGroupGraph.json 详见9.群集分析服务	群集分析：获取多个节点之间的直接关联的网络路径。
/rest/graph/commonLinkGraph.json 详见10.共同邻居服务	共同邻居：获取与输入的N个点同时存在关系的路径对象。
/rest/graph/pathAnalysisGraph.json 详见11.路径分析服务	路径分析：获取两个节点之间的最短网络路径或全路径，最多查询6度内的直接关系。
/rest/graph/kinshipGraph.json 详见12.血缘分析服务	血缘分析：多度关系的一种业务场景，查询一个点的同类关系多度扩展。

4.4.3 数据持久化服务API

API	描述
/rest/etl/persistNode.json 详见13.虚拟节点持久化服务	虚拟节点属性持久化。
/rest/etl/persistLink.json 详见14.添加关系持久化服务	增加关系持久化。

4.4.4 OLP转义API

API	描述
/rest/meta/queryOlpMetaConfig.json 详见15.查看OLP配置信息	查看OLP配置信息映射。
/rest/meta/updateOlpMetaConfig.json 详见16.添加关系持久化服务	更新OLP变量与业务变更的映射。

4.5 调用方式

4.5.1 请求结构

4.5.1.1 服务地址

环境	服务地址	备注
内部测试环境	---	无。
生产环境	---	根据实际部署的环境咨询PE。

4.5.1.2 通信协议

通讯协议采用HTTP协议。

4.5.1.3 请求方法

对 I+ API 接口调用是通过向 I+ API 的服务端地址发送HTTP POST请求，并按照接口说明在请求中加入相应请求参数来完成的；根据请求的处理情况，系统会返回处理结果。

4.5.1.4 请求参数

请求参数采用Json格式，application/json协议，requestBody请求。

4.5.1.5 字符编码

字符编码采用UTF-8编码。

4.5.2 公共参数

公共请求参数

I+提供的服务接口请求参数的结构为Json格式，放到请求的requestBody中，示例如下：

```
{
    "objectType": "O0003",
    "objectvalue": [
      "O0003P0004-3714113714164957 95"
    ]
    }
```

公共返回参数

I+提供的服务接口返回参数的结构为Json格式，Json结构中节点data内查询的返回的结果，elapsedTime表示接口的查询时间，noteMsg表示错误信息，success表示接口调用是否成功。

示例

```
{
```

```
  "data": {
    "groups": [],
    "linkCnt": 0,
    "linkDetails": {},
    "linkPropMaps": {},
    "linkProps": {},
    "links": [],
    "nodeCnt": 0,
    "nodePropMaps": {},
    },
    "nodesProps": {}
  },
  "elapsedTime": 59,
   "errorcode": 0,
  "noteMsg": "",
  "success": true
}
```


4.5.3 返回结果

成功结果

I+提供的服务接口返回参数的结构为Json格式，Json结构中success为true表示接口调用成功，成功结果保存在data中。

例如：


```
{
  "data": {
  },
  "elapsedTime": 167,
  "errorcode": 0,
  "noteMsg": "",
  "success": true
}
```


错误结果

I+提供的服务接口返回参数的结构为Json格式，Json结构中success为false表示接口调用失败，失败原因保存在noteMsg中。

例如：


```
{
  "data": {
  },
  "elapsedTime": 167,
  "noteMsg": "011005:用户未登录",
  "success": false
```

```
}
```

公共错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.5.4 接口限制

请求限制

避免计算资源紧张，保证I+系统稳定，每个SID同时接收请求N(10)个，超过的请求轮循等待10S，还没有请求释放，返回系统繁忙，请稍后再试。

4.5.5 鉴权

首次调用或session过期，返回011005错误码，表示没有权限

鉴权请求

```
post http://ip:7001/login.json
{"userNum":"xianyi","password":"e10adc3949ba59abbe56e057f20f883e"}
```

返回：

```
{
  "data": {
    "cookie": "995282b4-82f3-444d-ac6a-50030912e243".....
  },
  "elapsedTime": 167,
  "errorcode": 0,
  "noteMsg": "",
  "success": true
}
```

鉴权验证

后面所有请求，head中增加cookie:sid=995282b4-82f3-444d-ac6a-50030912e243。

4.6 实体查询服务

4.6.1 描述

实体查询API提供客户端通过I+接口，查询关系网络符合条件的实体节点，接口输入为查询类型及查询条件，比如通过姓名户籍查询身份证信息，通过车牌颜色和车牌号查询车辆信息等。

4.6.2 请求参数

名称	类型	是否必须	描述
objectType	String	是	实体节点的类型，跟I+后台配置的实体类型匹配，比如O0001为手机号。
propertyList	Array< PropertyFilter>	是	实体过滤条件。
propertyId	String	否	实体过滤属性的类型。
propertyType	String	否	实体过滤属性的类别。
values	Array<String>	否	实体过滤属性的值。

4.6.3 返回参数

名称	类型	描述
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。

4.6.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.6.5 示例

请求示例

```
{
   "objectType":"O0001",
   "propertyList":[
      {
         "propertyId":"O0001P001",
         "propertyType":"string_equal"
         "values":["321321321"]
      }
   ]
    }
```

返回示例

```
{
  "data": {
    "nodes": {
      "O0003P0004-893751893750806906": {
        "id": "O0003P0004-893751893750806906",
        "label": "张三",
        "type": "O0003",
        "virtual": false
      }
    },
    "nodesProps": {
     "O0003P0004-893751893750806906": {
      "O0003P0001":"张三",
      "O0003P0002":"男",
      "O0003P0003":"浙江省",
      "O0003P0004":"893751893750806906"
     }
    }
  },
  "elapsedTime": 50,
```

```
  "noteMsg": "",
  "success": true
      }
```

4.7 关系查询服务

4.7.1 描述

关系查询API提供客户端通过I+接口，查询关系网络符合条件的关系信息，接口输入为查询类型及查询条件，比如通过乘车时间和出发站等信息查询一次人与火车的关系等。

4.7.2 请求参数

名称	类型	是否必须	描述
linkType	String	是	关系节点的类型，跟I+后台配置的关系类型匹配，比如L0001为通话号。
propertyList	Array< PropertyFilter>	是	
propertyId	String	否	关系过滤属性的类型。
propertyType	String	否	关系过滤属性的类别。
values	Array<String>	否	关系过滤属性的值。

4.7.3 返回参数

名称	类型	描述
links	Array<Link>	关系边列表。
id	String	关系边ID。
source	String	关系边的源实体ID。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体ID。
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。

名称	类型	描述
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的ID。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.7.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.7.5 示例

请求示例

```
{
    "linkType":"L0001",
    "propertyList":[
        {
            "propertyId":"L0001P001",
            "propertyType":"string_equal",
            "values":[""]
        }
    ]
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
     } ,
     "L0003^49052e268251cf1b8252c407961e132a": {
      "L0003P0001":"",
      "L0003P0002":"",
     }
    },
    "links": [
```

```
      {
        "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
        ],
        "source": "O0003P0004-893751893750806906",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      },
      {
        "id": "O0003#O0003P0004-371411371416495795^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^49052e268251cf1b8252c407961e132a"
        ],
        "source": "O0003P0004-371411371416495795",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      }
    ],
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```


4.8 默认关联反查服务

4.8.1 描述

默认关联反查API提供客户端可以通过I+接口查询，从一个实体节点出发，查询默认关系的关系网络路径。

- 默认关系通过I+后台配置，支持关系类型配置，关系过滤属性配置，目标节点属性。
- 如果默认关系没有配置，默认查询实体类型可以查询的所有一度直接关系。

4.8.2 请求参数

名称	类型	是否必须	描述
objectType	String	是	实体节点的类型，跟I+后台配置的实体类型匹配，比如O0001为手机号。
objectValue	String	是	实体的主键属性类型和主键值，主键属性类型与I+后台配置的实体主键类型一致，主键和主键值以-连接，比

名称	类型	是否必须	描述
			如：身份证-身份证号码，O0003P0004-371411371416495795。
degree	int	是	查询度数，支持1-5度，建议不超过5度。

4.8.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
linkCnt	Integer	网络中关系边个数。
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。

名称	类型	描述
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.8.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.8.5 示例

请求示例

```
{
  "workspaceId":"",
  "objectType":"O0001",
  "objectValue":"O0001P001",
  "degree":3,
  "defaultTimeCond":{
    "propertyId":"",
    "propertyType":"time",
    "values":[null,null]
    }
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
```

```
  } ,
  "L0003^49052e268251cf1b8252c407961e132a": {
   "L0003P0001":"",
   "L0003P0002":"",
  }
 },
 "links": [
  {
   "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
   "linkDetails": [
    "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
   ],
   "source": "O0003P0004-893751893750806906",
   "sourceType": "O0003",
   "target": "O0004P0002-HB1163",
   "targetType": "O0004"
  },
  {
   "id": "O0003#O0003P0004-371411371416495795^O0004#O0004P0002-HB1163",
   "linkDetails": [
    "L0003^49052e268251cf1b8252c407961e132a"
   ],
   "source": "O0003P0004-371411371416495795",
   "sourceType": "O0003",
   "target": "O0004P0002-HB1163",
   "targetType": "O0004"
  }
 ],
 "nodeCnt": 3,
 "nodes": {
  "O0003P0004-371411371416495795": {
   "id": "O0003P0004-371411371416495795",
   "label": "李四",
   "type": "O0003",
   "virtual": false
  },
  "O0003P0004-893751893750806906": {
   "id": "O0003P0004-893751893750806906",
   "label": "张三",
   "type": "O0003",
   "virtual": false
  },
  "O0004P0002-HB1163": {
   "id": "O0004P0002-HB1163",
   "label": "HB1163",
   "type": "O0004",
   "virtual": false
  }
 },
 "nodesProps": {
  "O0003P0004-371411371416495795": {
   "O0003P0001":"李四",
   "O0003P0002":"",
   "O0003P0003":"",
   "O0003P0004":"371411371416495795"
  },
  "O0003P0004-893751893750806906": {
   "O0003P0001":"张三",
   "O0003P0002":"",
   "O0003P0003":"",
   "O0003P0004":"893751893750806906"
  },
```

```
    "O0004P0002-HB1163": {
     "O0003P0001":"",
     "O0003P0002":"HB1163",
     "O0003P0003":""
    }
   }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```

4.9 关联反查服务

4.9.1 描述

关联反查API提供客户端可以通过I+接口查询，从一个或多个实体节点出发，查询指定关系的关系网络路径。

- 支持多个不同类型的节点出发。
- 支持查询多个关系。
- 支持定义关系属性的过滤条件。
- 支持定义目标节点属性的过滤条件。

4.9.2 请求参数

名称	类型	是否必须	描述
objects	String,Array<String>	是	起始节点列表,KV结构，K为实体类型，和I+后台配置相同，V为实体ID数组，参见请求示例。
links	Array<Link>	是	图查询的需要关系。
linkType	String	是	关系类型和I+后台配置相同。
direct	int	否	关系出入度，0:出度，1:入度，2:出入度，3:无向，不传默认无向/出入度（支持关联反查，群集分析，共同邻居）。

名称	类型	是否必须	描述
queryProps	Array<String>	否	指定想要查询的属性，默认查询所有属性。
propertyList	Array<PropertyFilter>	否	需要过滤关系的属性，详见示例。
propertyId	String	否	关系过滤属性的类型。
propertyType	String	否	关系过滤属性的查询类型。
values	Array<String>	否	详见其他说明。
derivated	Array<PropertyFilter>	否	同字类关系的过滤属性，详见示例，同类为后台管理配置的间接关系，不输入使用后面默认配置。
statPropList	Array<statPropFilter>	否	累计属性条件过滤，用于数值类型sum,count值过滤，属性后台配置，支持关联反查，群集分析，共同邻居。
propertyId	string	否	固定值，一般为linkType+T+基础属性ID(次数为P000)，例：L0001TP000。
propertyType	string	否	固定为number。
values	List<String>	否	同基础属性。
statType	int	否	统计条件类型，0按属性值/次数查询，1 按属性值/次数的排序号查询。
targetObjects	Array<ObjectFilter>	否	目标节点的属性过滤，详见示例。
objectType	String	是	节点类型和I+后台配置相同。

名称	类型	是否必须	描述
propertyList	Array<PropertyFilter>	否	需要目标节点的属性，详见示例。
propertyId	String	否	目标节点过滤属性的类型。
propertyType	String	否	目标节点过滤属性的类别。
values	Array<String>	否	目标节点过滤属性的值。

4.9.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
linkCnt	Integer	网络中关系边个数。
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。

名称	类型	描述
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.9.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.9.5 示例

请求示例

```
{
  "data": {
    "objects": {
      "O0003": [
        "O0003P0004-3714113714164957 95",
        "O0003P0004-893751893750806906"
      ],
      "O0004":[
       "O0004P0002-HB1163",
       "O0004P0002-HB1985",
       "O0004P0002-HB4999"
      ]
    },
    "links": [
      {
        "linkType": "L0003",
        "propertyList": [],
        "derivated": []
      },
      {
        "linkType": "L0004",
        "propertyList": [
          {
            "propertyId": "L0004CC002",
            "propertyType": "string_like",
            "values": [
              "11"
            ]
          }
        ],
        "derivated": [
          {
            "count": {
              "propertyType": "string_equal",
              "values": [
```

```
            "1"
          ]
        },
        "propIds": [
          "L0004CC002"
        ],
        "intervaltime": []
      }
    ]
   }
  ],
  "targetObjects": [
   {
    "objectType": "O0003",
    "propertyList": [
     {
      "propertyId": "O0003P0002",
      "propertyType": "string_like",
      "values": [
       "22"
      ]
     }
    ]
   }
  ]
 }
}
```

返回示例

```
{
 "data": {
  "linkCnt": 2,
  "linkDetails": {
   "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
    "label": "乘火车",
    "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
    "linkType": "L0003",
   },
   "L0003^49052e268251cf1b8252c407961e132a": {
    "label": "乘火车",
    "linkId": "L0003^49052e268251cf1b8252c407961e132a",
    "linkType": "L0003"
   }
  },
  "linkProps": {
   "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
    "L0003P0001":"",
    "L0003P0002":"",
   } ,
   "L0003^49052e268251cf1b8252c407961e132a": {
    "L0003P0001":"",
    "L0003P0002":"",
   }
  },
  "links": [
   {
    "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
    "linkDetails": [
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
    ],
```

```
      "source": "O0003P0004-893751893750806906",
      "sourceType": "O0003",
      "target": "O0004P0002-HB1163",
      "targetType": "O0004"
    },
    {
      "id": "O0003#O0003P0004-371411371416495795^O0004#O0004P0002-HB1163",
      "linkDetails": [
        "L0003^49052e268251cf1b8252c407961e132a"
      ],
      "source": "O0003P0004-371411371416495795",
      "sourceType": "O0003",
      "target": "O0004P0002-HB1163",
      "targetType": "O0004"
    }
  ],
  "nodeCnt": 3,
  "nodes": {
    "O0003P0004-371411371416495795": {
      "id": "O0003P0004-371411371416495795",
      "label": "李四",
      "type": "O0003",
      "virtual": false
    },
    "O0003P0004-893751893750806906": {
      "id": "O0003P0004-893751893750806906",
      "label": "张三",
      "type": "O0003",
      "virtual": false
    },
    "O0004P0002-HB1163": {
      "id": "O0004P0002-HB1163",
      "label": "HB1163",
      "type": "O0004",
      "virtual": false
    }
  },
  "nodesProps": {
   "O0003P0004-371411371416495795": {
    "O0003P0001":"李四",
    "O0003P0002":"",
    "O0003P0003":"",
    "O0003P0004":"371411371416495795"
   },
   "O0003P0004-893751893750806906": {
    "O0003P0001":"张三",
    "O0003P0002":"",
    "O0003P0003":"",
    "O0003P0004":"893751893750806906"
   },
   "O0004P0002-HB1163": {
    "O0003P0001":"",
    "O0003P0002":"HB1163",
    "O0003P0003":""
   }
  }
},
"elapsedTime": 0,
"noteMsg": "",
"success": true
```

```
}
```

4.10 群集分析服务

4.10.1 描述

群集分析API提供客户端可以通过I+接口查询，从多个实体节点出发，查询实体之间指定关系的一度关系网络路径。

- 支持多个不同类型的节点出发。
- 支持查询多个关系。
- 支持定义关系的过滤条件。
- 目标节点在输入节点范围内。

4.10.2 请求参数

名称	类型	是否必须	描述
group	Array<group>	否	输入节点分组，用于合并节点的群集分析。
id	String	否	合并节点ID，值唯一就可以。
children	Array<String>	否	一个合并节点ID包含的节点。
objects	String,Array<String>	是	起始节点列表,KV结构，K为实体类型，和I+后台配置相同，V为实体ID数组，参见请求示例。
links	Array<Link>	是	图查询的需要关系。
linkType	String	是	关系类型和I+后台配置相同。
direct	int	否	关系出入度，0:出度，1:入度，2:出入度，3:无向，不传默认无向/出入度（支持关联反查，群集分析，共同邻居）。

名称	类型	是否必须	描述
queryProps	Array<String>	否	指定想要查询的属性，默认查询所有属性。
propertyList	Array<PropertyFilter>	否	需要过滤关系的属性，详见示例。
propertyId	String	否	关系过滤属性的类型。
propertyType	String	否	关系过滤属性的查询类型。
values	Array<String>	否	详见其他说明。
derivated	Array<PropertyFilter>	否	同字类关系的过滤属性，详见示例，同类为后台管理配置的间接关系，不输入使用后面默认配置。
statPropList	Array<statPropFilter>	否	累计属性条件过滤，用于数值类型sum,count值过滤，属性后台配置，支持关联反查，群集分析，共同邻居。
propertyId	string	否	固定值，一般为linkType+T+基础属性ID(次数为P000)，例：L0001TP000。
propertyType	string	否	固定为number。
values	List<String>	否	同基础属性。
statType	int	否	统计条件类型，0按属性值/次数查询，1 按属性值/次数的排序号查询。

4.10.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。

名称	类型	描述
linkCnt	Integer	网络中关系边个数。
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属

名称	类型	描述
		性值，比如"L0003P0001":"乘车时间"。

4.10.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.10.5 示例

请求示例

```
{
  "data": {
    "objects": {
      "O0003": [
        "O0003P0004-3714113714164957 95",
        "O0003P0004-893751893750806906"
      ],
      "O0004":[
       "O0004P0002-HB1163",
       "O0004P0002-HB1985",
       "O0004P0002-HB4999"
      ]
    },
    "links": [
      {
        "linkType": "L0003",
        "propertyList": [],
        "derivated": []
      },
      {
        "linkType": "L0004",
        "propertyList": [
          {
            "propertyId": "L0004CC002",
            "propertyType": "string_like",
            "values": [
              "11"
            ]
          }
        ],
        "derivated": [
          {
            "count": {
              "propertyType": "string_equal",
              "values": [
                "1"
              ]
            },
            "propIds": [
              "L0004CC002"
            ],
            "intervaltime": []
          }
        ]
      }
    ],
    "targetObjects": [
      {
        "objectType": "O0003",
        "propertyList": [
          {
            "propertyId": "O0003P0002",
            "propertyType": "string_like",
            "values": [
              "22"
            ]
          }
```

```
      ]
    }
   ]
  }
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
     } ,
     "L0003^49052e268251cf1b8252c407961e132a": {
      "L0003P0001":"",
      "L0003P0002":"",
     }
    },
    "links": [
     {
       "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
       "linkDetails": [
         "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
       ],
       "source": "O0003P0004-893751893750806906",
       "sourceType": "O0003",
       "target": "O0004P0002-HB1163",
       "targetType": "O0004"
     },
     {
       "id": "O0003#O0003P0004-371411371416495795^O0004#O0004P0002-HB1163",
       "linkDetails": [
         "L0003^49052e268251cf1b8252c407961e132a"
       ],
       "source": "O0003P0004-371411371416495795",
       "sourceType": "O0003",
       "target": "O0004P0002-HB1163",
       "targetType": "O0004"
     }
    ],
    "nodeCnt": 3,
    "nodes": {
      "O0003P0004-371411371416495795": {
        "id": "O0003P0004-371411371416495795",
        "label": "李四",
        "type": "O0003",
```

```
        "virtual": false
      },
      "O0003P0004-893751893750806906": {
        "id": "O0003P0004-893751893750806906",
        "label": "张三",
        "type": "O0003",
        "virtual": false
      },
      "O0004P0002-HB1163": {
        "id": "O0004P0002-HB1163",
        "label": "HB1163",
        "type": "O0004",
        "virtual": false
      }
    },
    "nodesProps": {
     "O0003P0004-3714113714164957795": {
      "O0003P0001":"李四",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"3714113714164957795"
     },
     "O0003P0004-893751893750806906": {
      "O0003P0001":"张三",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"893751893750806906"
     },
     "O0004P0002-HB1163": {
      "O0003P0001":"",
      "O0003P0002":"HB1163",
      "O0003P0003":""
     }
    }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```


4.11 共同邻居服务

4.11.1 描述

关联反查API提供客户端可以通过I+接口查询，从多个实体节点出发，查询指定关系，与输入节点直接相关节点的关系网络路径。

- 支持多个不同类型的节点出发。
- 支持查询多个关系。
- 支持定义关系的过滤条件。
- 支持定义目标节点的过滤条件。
- 支持配置共同邻居个数的配置。

4.11.2 请求参数

名称	类型	是否必须	描述
objects	String,Array<String>	是	起始节点列表，KV结构，K为实体类型，和I+后台配置相同，V为实体ID数组，参见请求示例。
links	Array<Link>	是	图查询的需要关系。
linkType	String	是	关系类型和I+后台配置相同。
direct	int	否	关系出入度，0:出度，1:入度，2:出入度，3:无向，不传默认无向/出入度（支持关联反查，群集分析，共同邻居）。
queryProps	Array<String>	否	指定想要查询的属性，默认查询所有属性。
propertyList	Array<PropertyFilter>	否	需要过滤关系的属性，详见示例。
propertyId	String	否	关系过滤属性的类型。
propertyType	String	否	关系过滤属性的查询类型。
values	Array<String>	否	详见其他说明。
derivated	Array<PropertyFilter>	否	同字类关系的过滤属性，详见示例，同类为后台管理配置的间接关系，不输入使用后面默认配置。
statPropList	Array<statPropFilter>	否	累计属性条件过滤，用于数值类型sum,count值过滤，属性后台配置，支持关联反查，群集分析，共同邻居。

名称	类型	是否必须	描述
propertyId	string	否	固定值，一般为linkType+T+基础属性ID(次数为P000)，例：L0001TP000。
propertyType	string	否	固定为number。
values	List<String>	否	同基础属性。
statType	int	否	统计条件类型，0按属性值/次数查询，1 按属性值/次数的排序号查询。
targetObjects	Array<ObjectFilter>	否	目标节点的属性过滤，详见示例。
objectType	String	是	节点类型和I+后台配置相同。
propertyList	Array<PropertyFilter>	否	需要目标节点的属性，详见示例。
propertyId	String	否	目标节点过滤属性的类型。
propertyType	String	否	目标节点过滤属性的类别。
values	Array<String>	否	目标节点过滤属性的值。
degree	int	否	用于路径分析，共同邻居，血缘分析。

4.11.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
linkCnt	Integer	网络中关系边个数。
nodes	Array<Node>	节点列表。
id	String	节点id。

名称	类型	描述
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.11.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.11.5 示例

请求示例

```
{
  "data": {
    "degree": "3",
    "objects": {
      "O0003": [
```

```
      "O0003P0004-3714113714164957 95",
      "O0003P0004-893751893750806906"
    ],
    "O0004":[
     "O0004P0002-HB1163",
     "O0004P0002-HB1985",
     "O0004P0002-HB4999"
    ]
  },
  "links": [
    {
      "linkType": "L0003",
      "propertyList": [],
      "derivated": []
    },
    {
      "linkType": "L0004",
      "propertyList": [
        {
          "propertyId": "L0004CC002",
          "propertyType": "string_like",
          "values": [
            "11"
          ]
        }
      ],
      "derivated": [
        {
          "count": {
            "propertyType": "string_equal",
            "values": [
              "1"
            ]
          },
          "propIds": [
            "L0004CC002"
          ],
          "intervaltime": []
        }
      ]
    }
  ],
  "targetObjects": [
    {
      "objectType": "O0003",
      "propertyList": [
        {
          "propertyId": "O0003P0002",
          "propertyType": "string_like",
          "values": [
            "22"
          ]
        }
      ]
    }
  ]
}
```

```
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
     } ,
     "L0003^49052e268251cf1b8252c407961e132a": {
      "L0003P0001":"",
      "L0003P0002":"",
     }
    },
    "links": [
      {
        "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
        ],
        "source": "O0003P0004-893751893750806906",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      },
      {
        "id": "O0003#O0003P0004-3714113714164957 95^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^49052e268251cf1b8252c407961e132a"
        ],
        "source": "O0003P0004-371411371416495795",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      }
    ],
    "nodeCnt": 3,
    "nodes": {
      "O0003P0004-371411371416495795": {
        "id": "O0003P0004-3714113714164957 95",
        "label": "李四",
        "type": "O0003",
        "virtual": false
      },
      "O0003P0004-893751893750806906": {
        "id": "O0003P0004-893751893750806906",
```

```
        "label": "张三",
        "type": "O0003",
        "virtual": false
      },
      "O0004P0002-HB1163": {
        "id": "O0004P0002-HB1163",
        "label": "HB1163",
        "type": "O0004",
        "virtual": false
      }
    },
    "nodesProps": {
     "O0003P0004-371411371416495795": {
      "O0003P0001":"李四",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"371411371416495795"
     },
     "O0003P0004-893751893750806906": {
      "O0003P0001":"张三",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"893751893750806906"
     },
     "O0004P0002-HB1163": {
      "O0003P0001":"",
      "O0003P0002":"HB1163",
      "O0003P0003":""
     }
    }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```


4.12 路径分析服务

4.12.1 描述

关联反查API提供客户端可以通过I+接口查询，查询两个实体之间指定关系的一度或多度关系网络路径。

- 支持多个不同类型的节点出发。
- 支持查询多个关系。
- 支持定义关系的过滤条件。
- 支持定义目标节点的过滤条件。
- 支持设置路径分析的路径查询数量。
- 最多支持6度的关系拓展。
- 路径分析不支持间接关系。

4.12.2 请求参数

名称	类型	是否必须	描述
objects	String,Array<String>	是	起始节点列表,KV结构，K为实体类型，和I+后台配置相同，V为实体ID数组，参见请求示例。
links	Array<Link>	是	图查询的需要关系。
linkType	String	是	关系类型和I+后台配置相同。
direct	int	否	关系出入度，0:出度，1:入度，2:出入度，3:无向，不传默认无向/出入度（支持关联反查，群集分析，共同邻居）。
queryProps	Array<String>	否	指定想要查询的属性，默认查询所有属性。
propertyList	Array<PropertyFilter>	否	需要过滤关系的属性，详见示例。
propertyId	String	否	关系过滤属性的类型。
propertyType	String	否	关系过滤属性的查询类型。
values	Array<String>	否	详见其他说明。
degree	int	否	传值则返回指定度数内的所有路径，不传值默认查询最短路径。

4.12.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
linkCnt	Integer	网络中关系边个数。

名称	类型	描述
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.12.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.12.5 示例

请求示例

```
{
  "data": {
    "type": "remote",
    "degree": "3",
    "objects": {
```

```
    "O0003": [
      "O0003P0004-371411371416495795",
      "O0003P0004-893751893750806906"
    ],
    "O0004":[
     "O0004P0002-HB1163",
     "O0004P0002-HB1985",
     "O0004P0002-HB4999"
    ]
  },
  "links": [
    {
      "linkType": "L0003",
      "propertyList": [],
      "derivated": []
    },
    {
      "linkType": "L0004",
      "propertyList": [
        {
          "propertyId": "L0004CC002",
          "propertyType": "string_like",
          "values": [
            "11"
          ]
        }
      ],
      "derivated": [
        {
          "count": {
            "propertyType": "string_equal",
            "values": [
              "1"
            ]
          },
          "propIds": [
            "L0004CC002"
          ],
          "intervaltime": []
        }
      ]
    }
  ],
  "targetObjects": [
    {
      "objectType": "O0003",
      "propertyList": [
        {
          "propertyId": "O0003P0002",
          "propertyType": "string_like",
          "values": [
            "22"
          ]
        }
      ]
    }
  ]
}
```

```
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
     } ,
     "L0003^49052e268251cf1b8252c407961e132a": {
      "L0003P0001":"",
      "L0003P0002":"",
     }
    },
    "links": [
      {
        "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
        ],
        "source": "O0003P0004-893751893750806906",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      },
      {
        "id": "O0003#O0003P0004-3714113714164957 95^O0004#O0004P0002-HB1163",
        "linkDetails": [
          "L0003^49052e268251cf1b8252c407961e132a"
        ],
        "source": "O0003P0004-371411371416495795",
        "sourceType": "O0003",
        "target": "O0004P0002-HB1163",
        "targetType": "O0004"
      }
    ],
    "nodeCnt": 3,
    "nodes": {
      "O0003P0004-371411371416495795": {
        "id": "O0003P0004-3714113714164957 95",
        "label": "李四",
        "type": "O0003",
        "virtual": false
      },
      "O0003P0004-893751893750806906": {
        "id": "O0003P0004-893751893750806906",
```

```
        "label": "张三",
        "type": "O0003",
        "virtual": false
      },
      "O0004P0002-HB1163": {
        "id": "O0004P0002-HB1163",
        "label": "HB1163",
        "type": "O0004",
        "virtual": false
      }
    },
    "nodesProps": {
     "O0003P0004-371411371416495795": {
      "O0003P0001":"李四",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"371411371416495795"
     },
     "O0003P0004-893751893750806906": {
      "O0003P0001":"张三",
      "O0003P0002":"",
      "O0003P0003":"",
      "O0003P0004":"893751893750806906"
     },
     "O0004P0002-HB1163": {
      "O0003P0001":"",
      "O0003P0002":"HB1163",
      "O0003P0003":""
     }
    }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```


4.13 血缘分析服务

4.13.1 描述

血缘分析API提供客户端可以通过I+接口查询，特定业务关系的多度扩展。

- 支持单个实体。
- 特定关系是指在后台管理-业务参数-血缘关系配置的关系，一般如果血缘，同户号等。
- 不支持条件过滤。
- 支持设置查询度数，不超过5度。

4.13.2 请求参数

名称	类型	是否必须	描述
objects	String,Array<String>	是	起始节点列表,KV结构，K为实体类型，和

名称	类型	是否必须	描述
			I+后台配置相同，V为实体ID数组，参见请求示例。
degree	int	否	传值则返回指定度数内的所有路径，不传值默认查询最短路径。

4.13.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
linkCnt	Integer	网络中关系边个数。
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。

名称	类型	描述
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.13.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.13.5 示例

请求示例

```
{
   "degree": "3",
   "objects": {
     "O0003": [
       "O0003P0004-371411371416495795"
     ]
   }
}
```

返回示例

```
{
  "data": {
    "linkCnt": 2,
    "linkDetails": {
      "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "乘火车",
        "linkId": "L0003^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0003",
      },
      "L0003^49052e268251cf1b8252c407961e132a": {
       "label": "乘火车",
        "linkId": "L0003^49052e268251cf1b8252c407961e132a",
        "linkType": "L0003"
      }
    },
    "linkProps": {
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0003P0001":"",
      "L0003P0002":"",
     } ,
     "L0003^49052e268251cf1b8252c407961e132a": {
      "L0003P0001":"",
```

```
    "L0003P0002":"",
   }
  },
  "links": [
   {
    "id": "O0003#O0003P0004-893751893750806906^O0004#O0004P0002-HB1163",
    "linkDetails": [
     "L0003^27b44b00fcf663b8fdfad968f8a2eba9"
    ],
    "source": "O0003P0004-893751893750806906",
    "sourceType": "O0003",
    "target": "O0004P0002-HB1163",
    "targetType": "O0004"
   },
   {
    "id": "O0003#O0003P0004-371411371416495795^O0004#O0004P0002-HB1163",
    "linkDetails": [
     "L0003^49052e268251cf1b8252c407961e132a"
    ],
    "source": "O0003P0004-371411371416495795",
    "sourceType": "O0003",
    "target": "O0004P0002-HB1163",
    "targetType": "O0004"
   }
  ],
  "nodeCnt": 3,
  "nodes": {
   "O0003P0004-371411371416495795": {
    "id": "O0003P0004-371411371416495795",
    "label": "李四",
    "type": "O0003",
    "virtual": false
   },
   "O0003P0004-893751893750806906": {
    "id": "O0003P0004-893751893750806906",
    "label": "张三",
    "type": "O0003",
    "virtual": false
   },
   "O0004P0002-HB1163": {
    "id": "O0004P0002-HB1163",
    "label": "HB1163",
    "type": "O0004",
    "virtual": false
   }
  },
  "nodesProps": {
   "O0003P0004-371411371416495795": {
    "O0003P0001":"李四",
    "O0003P0002":"",
    "O0003P0003":"",
    "O0003P0004":"371411371416495795"
   },
   "O0003P0004-893751893750806906": {
    "O0003P0001":"张三",
    "O0003P0002":"",
    "O0003P0003":"",
    "O0003P0004":"893751893750806906"
   },
   "O0004P0002-HB1163": {
    "O0003P0001":"",
    "O0003P0002":"HB1163",
```

```
      "O0003P0003":""
     }
    }
   },
   "elapsedTime": 0,
   "noteMsg": "",
   "success": true
}
```

4.14 虚拟节点持久化服务

4.14.1 描述

虚拟节点持久化API提供客户端持久化一个系统数据库中不存在的点到数据库。

持久化系统不存在的用户添加的节点信息。

4.14.2 请求参数

名称	类型	是否必须	描述
objectType	String	是	实体类型，和I+后台配置相同。
objectValue	String	是	节点值，属性标识与主键属性值的组合。
properties	Array<Properties>	是	实体需要添加的属性。
propId	string	是	实体属性ID，与后台管理配置对应。
propValue	string	是	实体属性对应的值。

4.14.3 返回参数

名称	类型	描述
nodeCnt	Integer	网络中实体节点个数。
nodes	Array<Node>	节点列表。
id	String	节点id。
label	String	节点标签，跟I+后台配置的实体属性一致。
type	String	节点类型，如O0001。
virtual	Boolean	节点在网络中是否存在。

名称	类型	描述
nodesProps	Array<Property>	节点属性列表。
<Property>	<String,String>	节点属性KV值，K为节点类型，和I+后台配置的实体类型一致，V为节点属性，比如"O0003P0001":"张三"。

4.14.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信	200	

错误代码	描述	Http状态码	语义
	息在noteMsg中详细解释。		

4.14.5 示例

请求示例

```
{
   "objectType":"O0001",
   "objectValue":"O0001P0001-1323432543",
   "properties":[
      {
         "propId":"O0001P001",
         "propValue":"1323432543"
      },
      {
         "propId":"O0001P002",
         "propValue":"陈大"
      },
      {
         "propId":"O0001P003",
         "propValue":"C类"
      },
      {
         "propId":"O0001P006",
         "propValue":"中国联通"
      },
      {
         "propId":"O0001P007",
         "propValue":"1340005435"
      }
   ]
}
```

返回示例

```
{
  "data": {
    "nodeCnt": 1,
    "nodes": {
      " O0001P0001-1323432543": {
        "id": " O0001P0001-1323432543",
        "label": "陈大",
        "avatar": "/static/img/id.png",
        "type": "O0001",
        "virtual": false
      }
    },
    "nodesProps": {
     " O0001P0001-1323432543": {
      " O0001P0001":"1323432543",
      " O0001P002":"陈大",
      " O0001P003":"C类",
      " O0001P006":"中国联通" ,
          " O0001P007":"2017-01-01 00:00:01"
```

```
    }
   }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```

4.15 添加关系持久化服务

4.15.1 描述

添加关系持久化API提供用户持久化两点之间的关系数据到数据库。

持久化系统不存在的，用户自己的关系数据。

4.15.2 请求参数

名称	类型	是否必须	描述
linkType	String,Array<String>	是	起始节点列表，KV结构，K为实体类型，和I+后台配置相同，V为实体ID数组，参见请求示例。
source	Object	是	关系的其中一个实体。
objectType	String	是	实体类型。
objectValue	int	否	实体值。
target	Object	是	关系的另外一个实体。
objectType	String	否	实体类型。
objectValue	String	否	实体值。
properties	Array<Properties>	是	实体需要添加的属性。
propId	string	是	实体属性ID，与后台管理配置对应。
propValue	string	是	实体属性对应的值。

4.15.3 返回参数

名称	类型	描述
linkCnt	Integer	网络中关系边个数。

名称	类型	描述
links	Array<Link>	关系边列表。
id	String	关系边id。
source	String	关系边的源实体id。
sourceType	String	关系边的源实体类型，如O0003。
target	String	关系边的目标实体id。
targetType	String	关系边的目标实体类型，如O0004。
linkDetails	Array<String>	关系边包含的边明细记录id列表。
linkDetails	Array<LinkDetail>	关系边明细列表。
label	String	关系边明细的label。
linkId	String	关系边明细的id。
linkType	String	关系边明细类型。
linkProps	Array<Property>	关系边明细属性列表。
<Property>	<String,String>	关系属性KV值，K为关系边属性类型，和I+后台配置的关系边属性类型一致，V为关系边属性值，比如"L0003P0001":"乘车时间"。

4.15.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信	200	

错误代码	描述	Http状态码	语义
	息在noteMsg中详细解释。		
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.15.5 示例

请求示例

```
{
    "linkType":"L00001",
    "source":{
        "objectType":"O0001",
        "objectValue":"O0001P001-13100000003"
    },
    "target":{
        "objectType":"O0001",
        "objectValue":"O0001P001-13100000004"
    },
    "properties":[
        {
            "propId":"L00001P013",
            "propValue":"13100000003"
        },
        {
            "propId":"L00001P015",
            "propValue":"13100000004"
        },
        {
            "propId":"L00001P012",
```

```
            "propValue":"14204320000"
        },
        {
            "propId":"L00001P022",
            "propValue":"1"
        },
        {
            "propId":"L00001P023",
            "propValue":"20"
        }
    ]
}
```

返回示例

```
{
  "data": {
    "linkCnt": 1,
    "linkDetails": {
      "L00001^27b44b00fcf663b8fdfad968f8a2eba9": {
        "label": "通话",
        "linkId": "L0001^27b44b00fcf663b8fdfad968f8a2eba9",
        "linkType": "L0001",
        "d": "1",
      }
    },
    "linkProps": {
     "L0001^27b44b00fcf663b8fdfad968f8a2eba9": {
      "L0001P0001":"",
      "L0001P0002":"",
     }
    },
    "links": [
      {
        "id": "O0001#O0001P0001-13100000003^O0001#O0001P0001-13100000004",
        "linkDetails": [
          "L0001^27b44b00fcf663b8fdfad968f8a2eba9"
        ],
        "source": "O0001P0001-13100000003",
        "sourceType": "O0001",
        "target": "O0001P0001-13100000004",
        "targetType": "O0001"
      }
    ]
    }
  },
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
```

```
}
```

4.16 查看olpmeta配置映射

4.16.1 描述

查看I+ OLP模型的配置信息，用于其他系统用户配置业务变量映射。

4.16.2 请求参数

无。

4.16.3 返回参数

名称	类型	描述
objects	Array<ObjectDef>	所有实体定义。
objDefId	String	实体定义ID，如：O0001。
objName	String	实体名称。
bizVar	String	业务变量。
objProps	Array<propertyDef>	属性列表。
propDefId	String	属性定义ID，如：O0001P001。
propName	String	属性中文名。
bizVar	String	属性业务变量。
links	Array<LinkDef>	所有关系定义。
linkDefId	String	关系定义ID，如：L0001。
linkName	String	实体名称。
bizVar	String	业务变量。
linkProps	Array<propertyDef>	属性列表。
propDefId	String	属性定义ID，如：L0001P001。
propName	String	属性中文名。
bizVar	String	属性业务变量。

4.16.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.16.5 示例

返回示例

```
{
        "data": {
   "objects":[
      {
         "objDefId":"O0001",
```

```
            "objName":"手机",
            "bizVar":"",
            "objProps":[
               {
                  "propDefId":"O0001P001",
                  "propName":"手机号",
                  "bizVar":""
               },{
                  "propDefId":"O0001P002",
                  "propName":"运营商",
                  "bizVar":""
               }
            ]
         },

      ],
      "links":[
         {
            "linkDefId":"L00001",
            "linkName":"通话",
            "bizVar":"tell",
            "linkProps":[
               {
                  "propDefId":"L0001P001",
                  "propName":"手机号",
                  "bizVar":"phone"
               }
            ]
         }
      ]
           },
   "elapsedTime": 0,
   "noteMsg": "",
   "success": true
}
```


4.17 更新olpmeta配置映射

4.17.1 描述

更新I+ OLP模型的配置信息，用于其他系统用户配置业务变量映射。

4.17.2 请求参数

名称	类型	描述
objects	Array<ObjectDef>	所有实体定义。
objDefId	String	实体定义ID，如：O0001。
objName	String	实体名称。
bizVar	String	业务变量。
objProps	Array<propertyDef>	属性列表。

名称	类型	描述
propDefId	String	属性定义ID，如：O0001P001。
propName	String	属性中文名。
bizVar	String	属性业务变量。
links	Array<LinkDef>	所有关系定义。
linkDefId	String	关系定义ID，如：L0001。
linkName	String	实体名称。
bizVar	String	业务变量。
linkProps	Array<propertyDef>	属性列表
propDefId	String	属性定义ID，如：L0001P001。
propName	String	属性中文名。
bizVar	String	属性业务变量。

4.17.3 返回参数

无。

4.17.4 错误码

错误代码	描述	Http状态码	语义
00XXXX	00开头的异常为系统通用异常，错误信息在noteMsg中详细解释。	200	
10XXXX	10开头的异常为鉴权认证异常，错误信息在noteMsg中详细解释。	200	
20XXXX	20开头的异常为用户及角色管理异常，错误信息在noteMsg中详细解释。	200	
30XXXX	30开头的异常为数据源配置异常，错误信息在noteMsg中详细解释。	200	

错误代码	描述	Http状态码	语义
40XXXX	40开头的异常为meta配置异常，错误信息在noteMsg中详细解释。	200	
50XXXX	50开头的异常为缓存刷新异常，错误信息在noteMsg中详细解释。	200	
60XXXX	60开头的异常为系统参数配置异常，错误信息在noteMsg中详细解释。	200	
70XXXX	70开头的异常为业务参数配置异常，错误信息在noteMsg中详细解释。	200	

4.17.5 示例

请求示例

```
{
   "objects":[
      {
         "objDefId":"O0001",
         "objName":"手机",
         "bizVar":"",
         "objProps":[
            {
               "propDefId":"O0001P001",
               "propName":"手机号",
               "bizVar":""
            },{
               "propDefId":"O0001P002",
               "propName":"运营商",
               "bizVar":""
            }
         ]
      },

   ],
   "links":[
      {
         "linkDefId":"L00001",
         "linkName":"通话",
         "bizVar":"tell",
         "linkProps":[
            {
               "propDefId":"L0001P001",
               "propName":"手机号",
```

```
                "bizVar":"phone"
            }
        ]
    }
  ]
}
```

返回示例

```
{
        "data": "OK"
  "elapsedTime": 0,
  "noteMsg": "",
  "success": true
}
```