

标签画像平台

<安装部署手册>

|  |  |
| --- | --- |
| **版本** | V2.0 |
| **组织** | 京东科技 |
| **发布日期** | 2021-05-27 |

京东科技控股股份有限公司

**目录**

[1. 法律声明 4](#_Toc48762593)

[2. 产品简介 5](#_Toc48762594)

[2.1. 产品概述 5](#_Toc48762595)

[2.2. 产品部署架构 6](#_Toc48762596)

[3. 部署环境需求 7](#_Toc48762597)

[3.1. 硬件资源需求 7](#_Toc48762598)

[3.2. 部署组件依赖需求 7](#_Toc48762599)

[4. 部署前准备 8](#_Toc48762600)

[4.1. 部署包信息 8](#_Toc48762601)

[4.2. 安装用户说明 8](#_Toc48762602)

[4.3. 部署路径说明 8](#_Toc48762603)

[5. 安装部署步骤操作指南 9](#_Toc48762604)

[5.1. 初始化 9](#_Toc48762605)

[5.2. 后端服务部署 11](#_Toc48762606)

[5.3. 前端服务部署 32](#_Toc48762607)

[6. 冒烟测试 33](#_Toc48762608)

[6.1. 访问配置 33](#_Toc48762609)

[6.2. 访问验证 33](#_Toc48762610)

# 法律声明

京东科技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. 您应当通过京东科技⽹站或京东科技授权通道下载、获取本⽂档，且仅能⽤于自身的合法合规的业务活动。本⽂档的内容视为京东科技的保密信息，您应当严格遵守保密义务；未经京东科技事先书⾯同意，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. 未经京东科技事先书⾯许可，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。京东科技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，并在京东科技网站或京东科技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京东科技网站或京东科技授权渠道下载、获取最新版的用户⽂档。

4. 本文档仅作为用户使⽤京东科技产品及服务的参考性指引，京东科技以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本⽂档。京东科技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，但京东科技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，京东科技不承担任何法律责任。在任何情况下，京东科技均不对任何间接性、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（即使京东科技已被告知该等损失的可能性）。

5. 京东科技⽂档中所有内容，包括但不限于图⽚、架构设计、⻚⾯布局、⽂字描述，均由京东科技和/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。⾮经京东科技和/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、发⾏或公开发表京东科技网站、产品程序或内容。此外，未经京东科技事先书⾯同意，任何⼈不得为了任何营销、⼴告、促销或其他⽬的使⽤、公布或复制京东科技的名称（包括但不限于单独为或以组合形式包含“京东科技”、“JDT”等京东科技和/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、图案表示、标志、标识或通过特定描述使第三⽅能够识别京东科技和/或其关联公司）。

6. 如若发现本⽂档存在任何错误，请与京东科技取得直接联系。

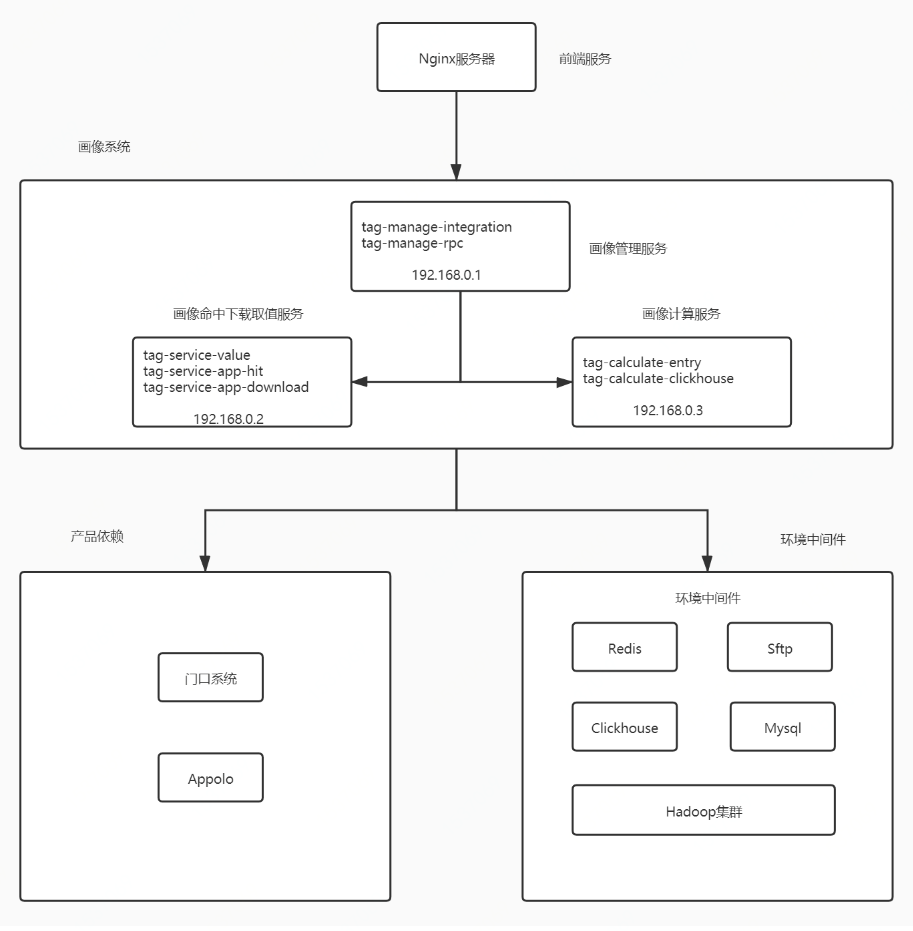
# 产品简介

## 产品概述

标签系统可以为业务人员提供一站式的数据分析、用户画像分析以及可视化门户等功能需求。通过标签系统的建设，可以统一全行的管理应用分析体系以及管理分析指标体系，全面提升各业务条线的管理和经营决策能力。整合不同业务条线的数据资源，满足全行数据资源调度需求，建立统一的数据服务平台，提升营管部的数据服务管理能力。

## 产品部署架构

### 部署架构图



### 架构图节点说明

|  |  |  |
| --- | --- | --- |
| **机器** | **部署模块** | **部署路径** |
| 画像管理前端 | tag.zip | /etc/nginx/apps |
| 标签画像后端 | bdp-tag-app-download-service.zip | /export/App/bdp-tag-app-download-service |
| bdp-tag-value-service.zip | /export/App/bdp-tag-value-service |
| bdp-tag-app-hit-service.zip | /export/App/bdp-tag-app-hit-servic |
| bdp-tag-rpc.zip | /export/App/bdp-tag-rpc |
| bdp-tag-calculate-entry.zip | /export/App/bdp-tag-calculate-entry |
| bdp-tag-manager.zip | /export/App/bdp-tag-manager |

# 部署环境需求

## 硬件资源需求

|  |  |  |
| --- | --- | --- |
| **机器类型** | **配置** | **台数** |
| 虚机 | 4C\*8G\*500G硬盘 | 3 |
| 物理机 |  |  |

## 部署组件依赖需求

|  |  |  |
| --- | --- | --- |
| **模块** | **依赖** | **部署方式** |
| 产品依赖 | 门户 | 提前准备 |
| Apollo配置中心 | 提前准备 |
| 环境中间件 | JDK | 手工部署 |
| Mysql | 天玑 |
| JSF | 提前准备 |
| Hadoop | 天玑 |
| Hbase | 天玑 |
| zookeeper | 天玑 |
| clickHouse | 天玑 |
| spark | 天玑 |
| hive | 天玑 |
| sftp | 提前准备 |
| nginx | 提前准备 |
| Azkaban | 天玑 |
| redis | 天玑 |

# 部署前准备

## 部署包信息

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用包** | **说明** | **版本** |
| 1 | bdp-tag-calculate-clickhouse.zip | 画像计算服务 | V2.0 |
| 2 | bdp-tag-app-download-service.zip | 人群下载服务 | V2.0 |
| 3 | bdp-tag-app-hit-service.zip | 人群命中服务 | V2.0 |
| 4 | bdp-tag-value-service.zip | 标签取值查询服务 | V2.0 |
| 5 | bdp-tag-rpc.zip | RPC接口服务 | V2.0 |
| 7 | bdp-tag-calculate-entry.zip | 计算入口服务 | V2.0 |
| 8 | bdp-tag-manager.zip | 画像管理后台服务 | V2.0 |
| 9 | tag.zip | 画像管理前端 | V2.0 |

## 安装用户说明

|  |  |  |
| --- | --- | --- |
| **用户** | **是否需要sudo权限** | **备注** |
| admin | 否 |  |

## 部署路径说明

|  |  |  |
| --- | --- | --- |
| 组件 | 路径类型 | 路径 |
| bdp-tag-app-download-service | 安装路径 | /export/App/bdp-tag-app-download-service |
| 日志路径 | /export/log/bdp-tag-app-download-service |
| bdp-tag-value-service | 安装路径 | /export/App/bdp-tag-value-service. |
| 日志路径 | /export/log/bdp-tag-value-service |
| bdp-tag-app-hit- service | 安装路径 | /export/App/bdp-tag-app-hit- service |
| 日志路径 | /export/log/bdp-tag-app-hit- service |
| bdp-tag-rpc | 安装路径 | /export/App/bdp-tag-rpc |
| 日志路径 | /export/log/bdp-tag-rpc |
| bdp-tag-calculate-entry | 安装路径 | /export/App/bdp-tag-calculate-entry |
| 日志路径 | /export/log/bdp-tag-calculate-entry |
| bdp-tag-manager | 安装路径 | /export/App/bdp-tag-manager |
| 日志路径 | /export/log /bdp-tag-manager |

# 安装部署步骤

## 初始化

### 检查kerberos认证，和验证链接hiveserver2

1.在天玑为本服务器安装hadoop（版本2.7.50），hive（版本2.3.2），hbase（版本1.2.6），spark（版本2.3.2）客户端，注意：版本不可错

并下发kerberos的keytab和krb5.conf

2. cd /export/kerberos/1（进入本服务器keytab目录）

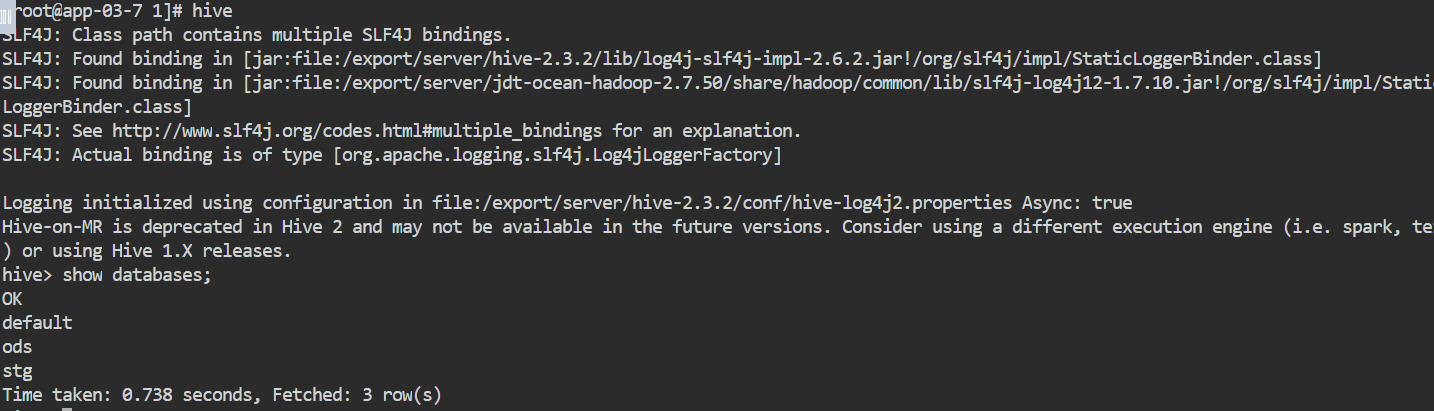
klist -kt hive.keytab （查看key）

例：

3. kinit -kt hive.keytab [hive/ocean@JD.COM](mailto:hive/ocean@JD.COM)

（认证kerberos，[hive/ocean@JD.COM是根据上面klist](mailto:hive/ocean@JD.COM是根据上面klist)查出的）

4.进入hive查看是否会报错

例: 

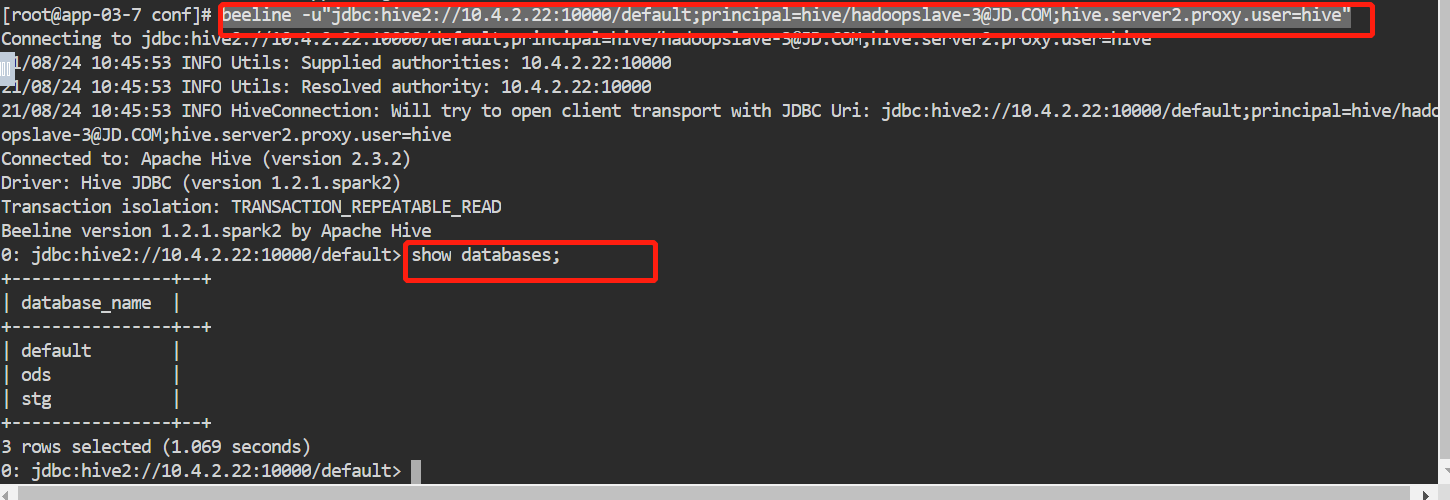
5.进入hiveserver2服务器，查看目录/export/hive/conf下的hive.keytab

例：

6.回到本服务器验证hiveserver2链接，并查看是否报错

beeline -u"jdbc:hive2://10.4.2.22:10000/default;principal=hive/hadoopslave-3@JD.COM;hive.server2.proxy.user=hive"

（[hive/hadoopslave-3@JD.COM，是复制上面步骤查询的hiveserver2](mailto:hive/hadoopslave-3@JD.COM，是复制上面步骤查询的hiveserver2)的key）

例: 

### jdk安装 （正常天玑在安装hadoop客户端时会自动安装jdk）

(注:jdk版本是基于linux的64位的1.8.0\_60，不能低于1.8.0\_60版本，如未安装jdk那么部署的所有服务器都需要安装jdk)，使用天玑进行JDK部署。

1. 检测环境变量

$ java -version

1. 安装JDK

若没有安装,上传jdk-8u261-linux-x64.tar.gz到/export/servers目录下，执行 tar -zxvf jdk-8u261-linux-x64.tar.gz解压

1. 添加以下环境变量（jdk安装路径作为依据）：

export JAVA\_HOME=/export/servers/jdk1.8.0\_261

export JAVA\_BIN=/export/servers/jdk1.8.0\_261/bin

export PATH=$JAVA\_HOME/bin:$PATH

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export JAVA\_HOME JAVA\_BIN PATH CLASSPATH JAVA\_OPTS

1. 重新刷新

执行source /etc/profile，使环境变量生效

1. 检测是否安装成功

$ java –version

出现java版本号即为安装成功

### 目录初始化

查看linux服务器是否有/export/App目录，没有则创建

$ cd /

$ mkdir export

$ cd export

$ mkdir App

### sftp配置

1. 配置sftp私钥

上传sftp私钥id\_rsa到应用服务器的/export/App/config 目录下,

若目录不存在, 创建目录: mkdir –p /export/App/config

上传后修改秘钥文件权限: chmod 600 id\_rsa

2. 上传应用模板文件excel\_template\_1.xlsx 到/export/App/config下

3. 在sftp中创建应用需要使用的目录及上传模板文件

远程连接sftp服务:

sftp -oIdentityFile=/export/App/config/id\_rsa -oPort=51888 mysftp@192.168.0.9

创建如下目录：(使用ls命令查看目录，不存在则创建):

ls /upload/user-tag/import

ls /upload/user-tag/BitmapStorage/bitmap-rule/user-tag

ls /upload/user-tag/BitmapStorage/bitmap-tag/user-tag

命令：(创建目录前检查层目录是否已存在, 如ls upload, 若存在则进入下一级目录)

$ mkdir upload

$ cd upload;

$ mkdir user-tag;

$ cd user-tag;

$ mkdir import;

$ mkdir BitmapStorage;

$ cd BitmapStorage;

$ mkdir bitmap-rule

$ mkdir bitmap-tag

$ cd bitmap-rule

$ mkdir user-tag

$ cd ../bitmap-tag

$ mkdir user-tag

上传模板文件到sftp的/upload/user-tag/import目录下

$ cd /upload/user-tag/import

$ put excel\_template.xlsx

### hbase初始化

hbase shell（进入hbase客户端）

创建表

create\_namespace 'user\_tag'

create 'user\_tag:cmb\_user\_all\_tag','t'

create 'user\_tag:cmb\_mapping\_jdpin','t'

create 'user\_tag:cmb\_mapping\_id\_no','t'

create 'user\_tag:cmb\_mapping\_mobile\_no','t'

create 'user\_tag:cmb\_mapping\_device\_no','t'

### 数据初始化

使用navicat等工具远程连接给定数据库

|  |
| --- |
| mysql –h IP –u user –p   * create database if not exists bdp\_tag\_manager\_sge; * source /export/App/bdp\_tag\_manager\_sge.sql |

## 后端服务部署

部署顺序依次为：

计算服务(bdp-tag-calculate-clickhouse)

人群下载服务(bdp-tag-app-download-service)

人群命中服务(bdp-tag-app-hit-service)

标签取值查询服务(bdp-tag-value-service)

管理rpc接口服务(bdp-tag-rpc)

计算入口服务(bdp-tag-calculate-entry)

后端服务(bdp-tag-manager)

### 画像计算服务

#### 上传应用包

上传bdp-tag-calculate-clickhouse.zip压缩包到/export/App目录下(若目录不存在, 创建目录mkdir –p /export/App)

解压安装包unzip bdp-tag-calculate-clickhouse.zip，

#### 更改配置文件

vim lib/bdp-tag-calculate-clickhouse.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-calculate-clickhouse apollo.meta=http://10.222.54.129:30002 apollo.bootstrap.enabled=true apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |

#### 在apollo上修改配置

在apollo上创建bdp-tag-calculate-clickhouse项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #clickhouse ip  bdp.clickhouse-tag.host = 10.4.2.19,10.4.2.18,10.4.2.21  #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  bdp.logback.level = DEBUG |

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### 人群下载服务

#### 上传应用包

1. 上传bdp-tag-app-download-service.zip压缩包到/export/App目录下
2. 解压安装包unzip bdp-tag-app-download-service.zip，解压后目录结构为：

#### 更改配置文件

vim lib/bdp-tag-app-download-service.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| app.id=bdp-tag-app-download-service apollo.meta=http://10.222.54.129:30002 apollo.bootstrap.enabled=true apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |

#### 在apollo上修改配置

在apollo上创建bdp-tag-app-download-service项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #r2m address 1  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  # tag  bdp.r2m-tag.token = 06eb096678  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  # offset2pin  bdp.r2m-tagoffset.token = 060c039575  bdp.r2m-tagoffset.name = dtdmp-offset-pin  bdp.r2m-tagoffset.password = PZ7FmzehvT4pVa07  #sftp  bdp.sftp.host = 10.4.2.20  bdp.sftp.port = 51999  bdp.sftp.username.tag = mysftp  #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  #  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes = 10.222.54.123:6379,10.222.54.123:6380,10.222.54.71:6381,10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUG |

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### 人群命中服务

#### 上传应用包

1. 上传bdp-tag-app-hit-service.zip压缩包到/export/App
2. 解压安装包unzip bdp-tag-app-hit-service.zip

#### 更改配置文件

vim lib/bdp-tag-app-hit-service.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-app-hit-service # apllo meta server address apollo.meta=http://10.222.54.129:30002 # will inject 'application' namespace in bootstrap phase apollo.bootstrap.enabled=true apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |
|
|
|
|
|

#### 在apollo上修改配置

在apollo上创建bdp-tag-app-hit-service项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #r2m  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  #tag manager  bdp.r2m-tag.token = 06eb096678  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  #pin2offset  bdp.r2m-tagpin.token = 0bf28c5a6f  bdp.r2m-tagpin.name = dtdmp-pin-offset  bdp.r2m-tagpin.password = 4vHmJ4Yt1vzwBgPJ  #sftp  bdp.sftp.host = 10.4.2.20  bdp.sftp.port = 51999  bdp.sftp.username.tag = mysftp  #jsf config  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  # dubbo  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes = 10.222.54.123:6379,10.222.54.123:6380,10.222.54.71:6381,10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUG |
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### 标签取值服务

#### 上传应用包

1. 上传bdp-tag-value-service.zip压缩包到/export/App目录下
2. 解压安装包unzip bdp-tag-value-service.zip

#### 更改配置文件

vim lib/bdp-tag-value-service.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-value-service # apllo meta server address apollo.meta=http://10.222.54.129:30002 # will inject 'application' namespace in bootstrap phase apollo.bootstrap.enabled=true # will inject 'application', 'application-demo' namespaces in bootstrap phase apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |
|
|
|
|
|
|

#### 在apollo上修改配置

在apollo上创建bdp-tag-value-service项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

注意：bdp.hive.serverprincipal这个参数是去查看hiveserver2服务的keytab（查看方法见初始化5.1.1）

|  |
| --- |
| #r2m address  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  # pin2offset  bdp.r2m-tagpin.token = 0bf28c5a6f  bdp.r2m-tagpin.name = dtdmp-pin-offset  bdp.r2m-tagpin.password = 4vHmJ4Yt1vzwBgPJ  # tag  bdp.r2m-tag.token = 06eb096678  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  #idMapping  hbase.idMapping = 1:cmb\_mapping\_jdpin,2:cmb\_mapping\_id\_no,3:cmb\_mapping\_mobile\_no,4:cmb\_mapping\_device\_no,5:cmb\_mapping\_jdpin,6:cmb\_mapping\_jdpin  # kerberos 开关 默认为true  bdp.kerberos.enable=true  # hbase  bdp.hbase.kerberosenable = true  bdp.hbase.registurl = 10.4.2.6:2181,10.4.2.7:2181,10.4.2.8:2181  bdp.hbase.masterprincipal = hbase/\_HOST@JD.COM  bdp.hbase.regionserverprincipal = hbase/\_HOST@JD.COM  bdp.hbase.krb5conf = /etc/krb5.conf  bdp.hbase.clientprincipal = hbase/ocean@JD.COM  bdp.hbase.keytab = /export/kerberos/1/hbase.keytab  # hive  bdp.hive.kerberosenable = true  bdp.hive.urlconnection = jdbc:hive2://10.4.2.22:10000  bdp.hive.database = default  bdp.hive.others =  bdp.hive.serverprincipal = hive/hadoopslave-3@JD.COM  bdp.hive.clientprincipal = hive/ocean@JD.COM  bdp.hive.keytab = /export/kerberos/1/hive.keytab  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  # dubbo 或者 jsf 配置jsf时扫描jsf-provider.xml，配置dubbo时扫描 dubbo-provider.xml  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes = 10.222.54.123:6379,10.222.54.123:6380,10.222.54.71:6381,10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUG |
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### rpc接口服务

#### 上传应用包

1. 上传bdp-tag-rpc.zip压缩包到/export/App
2. 解压安装包unzip bdp-tag-rpc.zip，

#### 更改配置文件

vim lib/bdp-tag-rpc.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-rpc # apllo meta server address apollo.meta=http://10.222.54.129:30002 # will inject 'application' namespace in bootstrap phase apollo.bootstrap.enabled=true # will inject 'application', 'application-demo' namespaces in bootstrap phase apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |

#### 在apollo上修改配置

在apollo上创建bdp-tag-rpc项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  #datasource  bdp.mysql-app.host = 10.4.2.20:3306  bdp.mysql-tag.database = bdp\_tag\_manager\_sge  bdp.mysql-tag.username = root  bdp.mysql-tag.password = 0gvzJr66iNs5  #r2m  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  bdp.r2m-tag.token = 06eb096678  #tag mamager  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  # dubbo  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes=10.222.54.123:6379,10.222.54.123:6380,10.222.54.71:6381,10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUG |

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### 计算入口服务

#### 上传应用包

1、 上传bdp-tag-calculate-entry.zip压缩包到/export/App

2、 解压安装包unzip bdp-tag-calculate-entry.zip

#### 更改配置文件

vim lib/bdp-tag-calculate-entry.jar 找到配置文件application.properties查看对应的激活文件 spring.profiles.active=xxx

退出编辑器，找到对应的application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-calculate-entry # apllo meta server address apollo.meta=http://10.222.54.129:30002 # will inject 'application' namespace in bootstrap phase apollo.bootstrap.enabled=true # will inject 'application', 'application-demo' namespaces in bootstrap phase apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |
|
|
|
|
|
|

#### 在apollo上修改配置

在apollo上创建bdp-tag-calculate-entry项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #r2m  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  bdp.r2m-tag.token = 06eb096678  #tag mamager  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  # dubbo  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes = 10.222.54.123:6379,10.222.54.123:6380,10.222.54.71:6381,10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUG |
|
|
|
|
|
|
|
|
|

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

### 管理后台服务

#### 上传应用包

1. 上传bdp-tag-manager.zip压缩包到/export/App
2. 解压安装包unzip bdp-tag-manager.zip

#### 更改配置文件

vim config/application-xxx.properties 修改apollo地址

|  |
| --- |
| # apollo app id app.id=bdp-tag-manager # apllo meta server address apollo.meta=http://10.222.54.129:30002 # will inject 'application' namespace in bootstrap phase apollo.bootstrap.enabled=true # will inject 'application', 'application-demo' namespaces in bootstrap phase apollo.bootstrap.namespaces=application-dev.properties apollo.bootstrap.eagerLoad.enabled=true |
|
|
|
|
|
|

#### 在apollo上修改配置

在apollo上创建bdp-tag-calculate-entry项目，并创建application-dev.properties的namespaces

把下面的配置复制到apollo的application-dev

|  |
| --- |
| #jsf  bdp.jsf.registurl = 10.0.2.10:20020  bdp.jsf.alias.tag = bdp-tag  #sso  sso.login.url = http://192.168.0.40:8080/sso/login  sso.exclude.path = /static,/operation/crowd/isHit,/operation/tag/mappingTagValue,/operation/crowd/download  sso.cookie.name = jdfcloud  sso.app.key = test3  sso.app.token = 347c6161e79f4b6a8873202dd5fe7e8f  sso.app.url = http://192.168.0.40:8080/  login.sso.invalid = true  #uas  bdp.uas.appurl = http://10.0.2.10:8080  bdp.uas.asmurl = http://linking-asm-core.ocean.bigdata  bdp.uas.cookiecipherkey = CW9bG0LwftWBr9W5v3gWug==  bdp.uas.cookiedomain = ocean.bigdata  bdp.uas.cookiemaxage = 1728000000  bdp.uas.homeurl = http://linking.ocean.bigdata  bdp.uas.invalid = false  bdp.uas.loginsystem = ocean  bdp.uas.salt = d036c1c6a287456193c05a710e05c414  bdp.uas.uasurl = http://uas.t1.poc.jdpaas  bdp.uas-tag.exeludepath = /static,/operation/crowd/isHit,/operation/tag/mappingTagValue,/operation/crowd/download  #datasource  bdp.mysql-app.host = 10.4.2.20:3306  bdp.mysql-tag.database = bdp\_tag\_manager\_sge  bdp.mysql-tag.username = root  bdp.mysql-tag.password = 0gvzJr66iNs5  #r2m  bdp.r2m.registurl = 10.0.2.62:8175,10.0.2.63:8175,10.0.2.64:8175  #tag manager  bdp.r2m-tag.name = dtdmp-tag  bdp.r2m-tag.token = 06eb096678  bdp.r2m-tag.password = 0JEUtJihIJFTvUNc  bdp.r2m-tagoffset.token = 060c039575  bdp.r2m-tagoffset.name = dtdmp-offset-pin  bdp.r2m-tagoffset.password = PZ7FmzehvT4pVa07  #sftp  bdp.sftp.host = 10.4.2.20  bdp.sftp.port = 51999  bdp.sftp.username.tag = mysftp  bdp.routerurl.gaia = http://internal.bdp-gaia-api-gateway/common-authority/  bdp.routerurl.asm = http://internal.bdp-portal-asm/restful  bdp.hadoop.cluster.id = 52  bdp.cache.use.r2m = true  bdp.cache.use.redis = false  # dubbo  bdp.rpc.use = jsf  bdp.dubbo.registryUrl = 10.222.50.57:2181,10.222.51.69:2181,10.222.54.123:2181  # redis  bdp.redis.nodes=10.222.54.71:6382,10.222.54.70:6383,10.222.54.70:6384  bdp.redis.password = JDDocean2020  bdp.logback.level = DEBUGbdp.r2m-tag.password = 0JEUtJihIJFTvUNc  #sftp  bdp.sftp.host = 10.222.51.68  bdp.sftp.port = 51899  bdp.sftp.username.tag = mysftp  bdp.routerurl.gaia=10.222.54.45:8170  bdp.routerurl.asm=10.222.54.126:17081  bdp.hadoop.cluster.id=52 |
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|

#### 启动服务

sh bin/start.sh （首次启动需格式化shell文件 sed -i 's/\r//' start.sh）

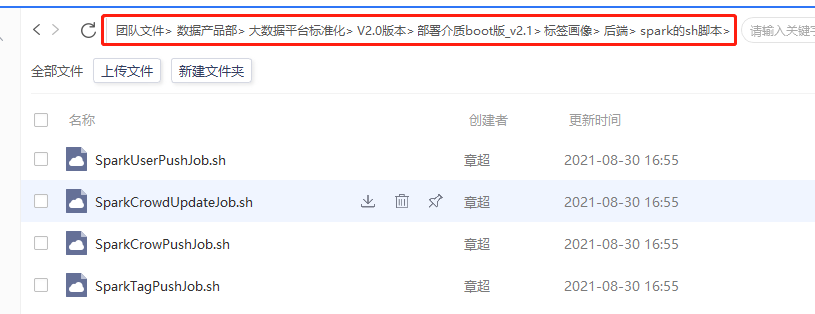
#### 停止服务

sh bin/stop.sh （首次启动需格式化shell文件 sed -i 's/\r//' stop.sh）

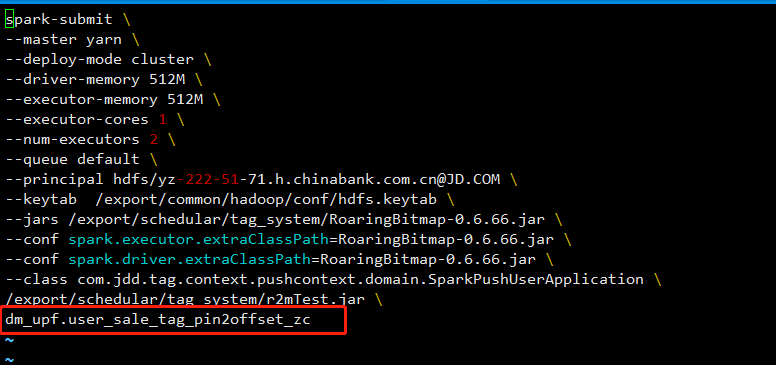
### 离线计算服务

离线计算服务：主要用于clickhouse数据推送，hbase数据推送，人群数据生成，标签数据生成，标签数据合并，标签数据统计

找到安装有spark客户端的服务器, 上传RoaringBitmap-0.6.66.jar；ost\_tag\_bigdata-1.0-SNAPSHOT.jar;4个sh脚本，到/export/schedular/tag\_system目录下, 若目录不存在mkdir –p /export/schedular/tag\_system 创建目录



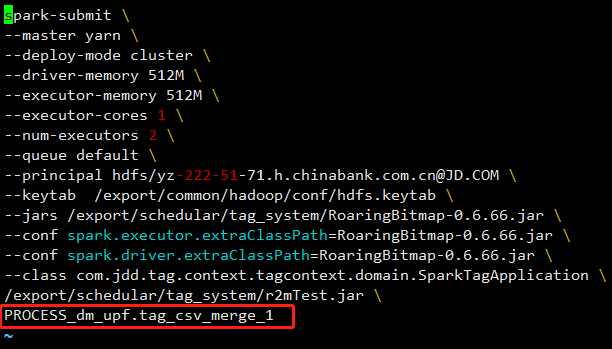
SparkUserPushJob.sh：偏移量推送任务（多主题，需要修改脚本中偏移量表名）



SparkCrowdUpdateJob.sh：人群更新任务

SparkCrowPushJob.sh：人群推送任务

SparkTagPushJob.sh：标签推送，（多主题，需要修改宽表表名）



修改4个sh脚本中的principal和keytab（/export/kerberos/1/hdfs.keytab）

例：

修改jar包里的配置文件：vim ost\_tag\_bigdata-1.0-SNAPSHOT.jar，找到根目录下的context.properties文件，修改以下配置项后保存即可

|  |  |  |
| --- | --- | --- |
| 配置项名称 | 业务含义 | 备注 |
| base.estimate.hostpost | CK IP端口 | 例：172.25.61.25:8123,172.25.61.20:8123 |
| filesys.user.store | sftp用户名 |  |
| filesys.password.store | sftp密钥内容 | 取出/export/App/config/ id\_rsa文件内容, 换行处用\n替代 |
| filesys.host.store | sftp IP |  |
| filesys.port.store | sftp端口 |  |
| cache.app.name | 标签redis应用名 |  |
| cache.hostports | redis连接URL |  |
| cache.pwd.app | 标签redis密码 |  |
| cache.app.u2o | pin-offset redis应用 |  |
| cache.pwd.u2o | pin-offset redis密码 |  |
| cache.app.o2u | offset-pin redis应用 |  |
| cache.pwd.o2u | offset-pin redis密码 |  |
| kv.store.host | hbase quorum |  |
| kv.store.port | hbase zk 端口 |  |
| kv.kerberos.enable | Kerber开关 默认为true |  |
| kv.client.kerberos.principal | hbase 客户端 principal |  |
| kv.server.kerberos.principal | hbase 服务端 principal |  |
| keytab.file | kerberos keytab文件名 |  |
| krb5.file | kerberos krb5配置文件名 |  |

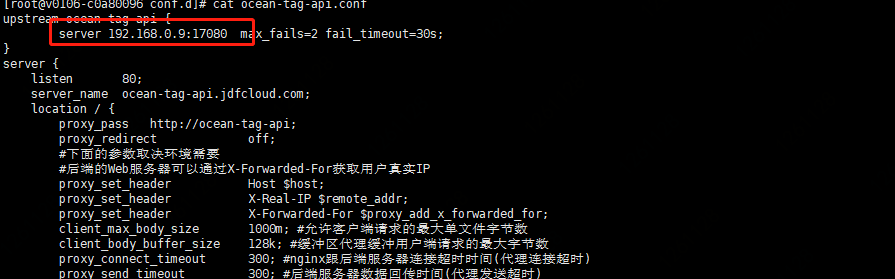
修改4个脚本中的

## 前端服务部署

* 1. 登录部署门户前端服务器, 一般为nginx服务器
  2. $ cd /usr/local/nginx/apps

(注意: /usr/local/nginx为nginx安装路径, 但要根据实际nginx安装路径来定)

* 1. 上传前端介质tag.zip压缩包到/etc/nginx/apps目录下
  2. $ unzip tag.zip
  3. 上传附件中的ocean-tag-api.conf到/usr/local/nginx/conf/domains目录下, 并修改ip和端口(tag-manage-integration应用部署的ip和服务端口)



* 1. 修改配置nginx.conf

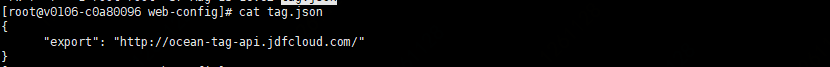
$ vim /usr/local/nginx/nginx.conf

在原server下追加如下配置：

|  |
| --- |
| location /tag{  root /usr/local/nginx/apps/; #前端介质存在目录  access\_log on;  try\_files $uri $uri/ /index.html =404;  }  location /tag/core {  # tag-manage-integration应用部署的ip和服务端口  proxy\_pass http://192.168.0.9:17080/;  proxy\_redirect off;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  } |

* 1. 配置 web-config

上传附件中的tag.json到/usr/local/nginx/web-config目录下：



* 1. 重新加载nginx: nginx -s reload

# 冒烟测试

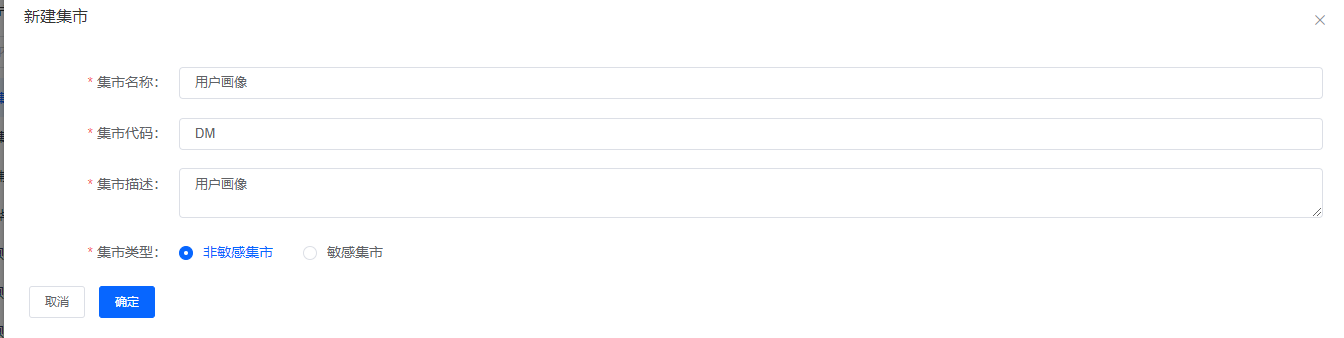
## 画像使用的hive库

库名：dm\_upf, 若没有先建库, 可在数据标准中新建集市和分层

1. 新建集市



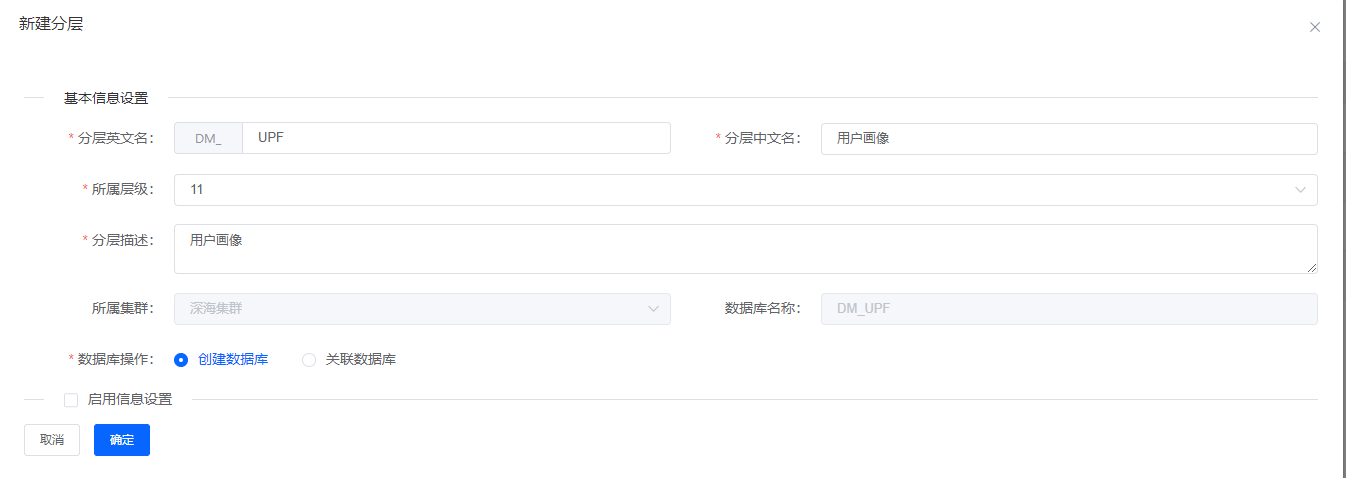
填写集市信息:



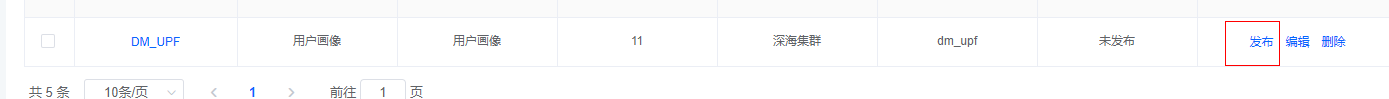
1. 新建分层



填写分层信息:



发布分层:



发布成功后dm\_upf库就已建好

## 用户id-offset表

（1）此表唯一, 若已有且数据已维护则忽略此步,若需导入新数据,需与开发确认表名,编码起点,不能覆盖已有数据

（1）开发背景：为提升计算效率，画像系统开发用户id与offset（用户编码）对应表，对用户id进行编码

（2）表结构

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| jdpin | string | 用户id |
| jdpin\_offset | bigint | 用户唯一编码 |
| dt | string | 日期分区 |

建表测试语句参考如下:



（3）注意事项：用户id与offset一一对应且唯一, 用户id为画像系统所用全量有效用户id；offset必须为数字

（4）导入数据

（5）更新策略: 增量表, 所有分区总和为一份全量数据

## 明确标签基本信息

（1）标签编码：根据自定规则确定标签编码，标签编码唯一

（2）标签名称：根据标签意义确定标签名称，标签名称唯一

（3）标签逻辑：明确数据来源及口径

（4）标签数据类型：string/bigint

## 注意事项

考虑数据存储及计算环境，标签数据类型需谨慎设置, 尽量选用bigint

## 标签表

1. 表结构

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| jdpin | string | 用户id |
| p0000001 | bigint | 标签字段1(如:性别) |
| p0000002 | bigint | 标签字段2(如:年龄段) |
| p000000\* | bigint | 标签字段n |
| dt | string | 日期分区 |

建表测试语句参考如下:



(2)表内包含数据

标签值：根据标签口径获取；空值(null)及无效值统一用-9999替代；标签值不要使用汉字，用数字或字母替代

用户id：关联用户id-offset对应表，获取offset

(3)注意事项

表内用户id唯一，不能出现重复id，否则会导致数据不准确问题

(4)更新策略: 快照表, 每个分区存储一份全量数据

## 标签宽表

1. 表结构

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| jdpin | string | 用户id |
| dt | string | 日期分区 |

建表测试语句参考如下:



(2)注意事项

a)宽表禁止导入数据

b)宽表为画像系统数据加工的基本单元, 所以不同周期,业务以及不希望相互影响的标签尽量指定不同的宽表

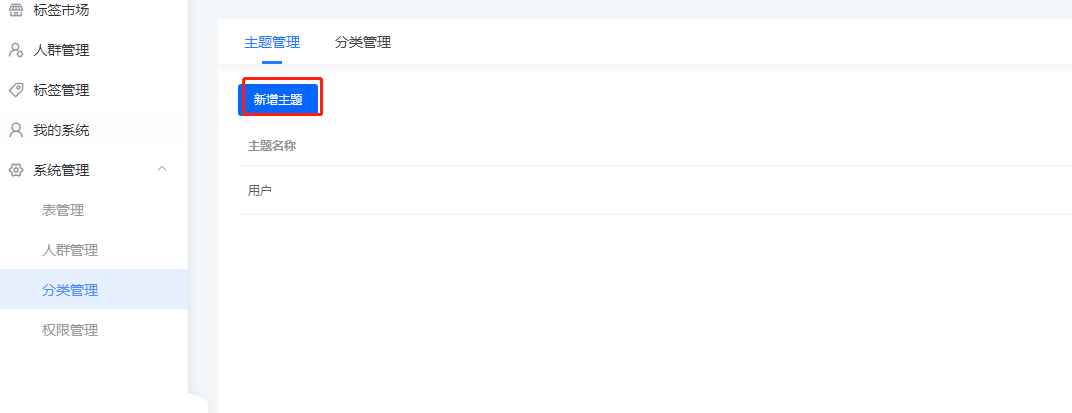
c)宽表末尾建议增加merge等关键字,方便与标签表区分

d)一套环境中的所有宽表的用户id和日期分区字段名需要一致

## 导入标签数据

向标签表导入数据, 标签表里需要做好数据才能去注册, 否则没有意义

## 创建主题和分类管理

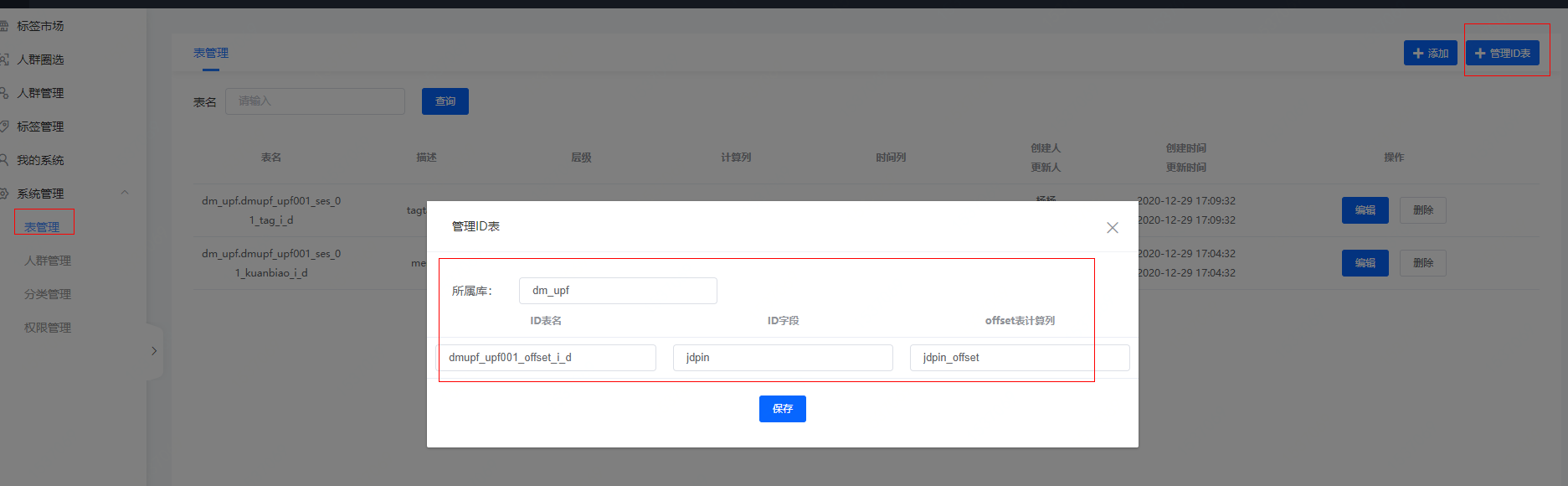




## 在对应主题下注册偏移量表和标签表

1).注册id-offset表, 该表注册一次即可, 若已注册则忽略:

如下图,添加hive库里的id-offset表名，用户id字段, 编码字段名称



2).注册宽表

3).注册标签表, 选择宽表要注意,不同周期,业务以及不希望相互影响的标签尽量选择不同的宽表

4.).注册标签

## 测试阶段

1.执行标签计算任务(spark任务), 测试能否加工成功

执行方式有两种:

1) 在命令行执行任务

登入部署离线计算服务的服务器,确保/export/schedular/tag\_system目录下的程序包已存在并且环境配置都已修改好, 执行spark命令SparkUserPushJob.sh偏移量任务, SparkTagPushJob.sh标签推送任务， 执行后登录yarn管理端,查看任务执行日志.

2) 在作业中心配置任务手动执行进行测试, 步骤参考下一节

2.执行成功后, 测试标签能否正常

1)测试计算人数功能

画像系统-人群圈选-选择建好的标签-保存筛选-计算人群数量-保存人群

2)测试人群生成功能

命令行执行SparkCrowdUpdateJob.sh人群更新任务或者在作业中心手动执行相应任务

执行成功后查看人群是否变为有效状态

3)测试画像分析等功能

若测试失败, 根据测试失败原因调整标签表数据

举例：

（1）枚举值不匹配问题

筛查标签表中枚举值是否存在与配置的标签枚举值不匹配的问题，修复后重新测试即可

（2）标签表或者标签注册错误问题

检查标签表、标签宽表、标签注册是否正确

若测试成功,再重新执行标签加工任务,成功后即可

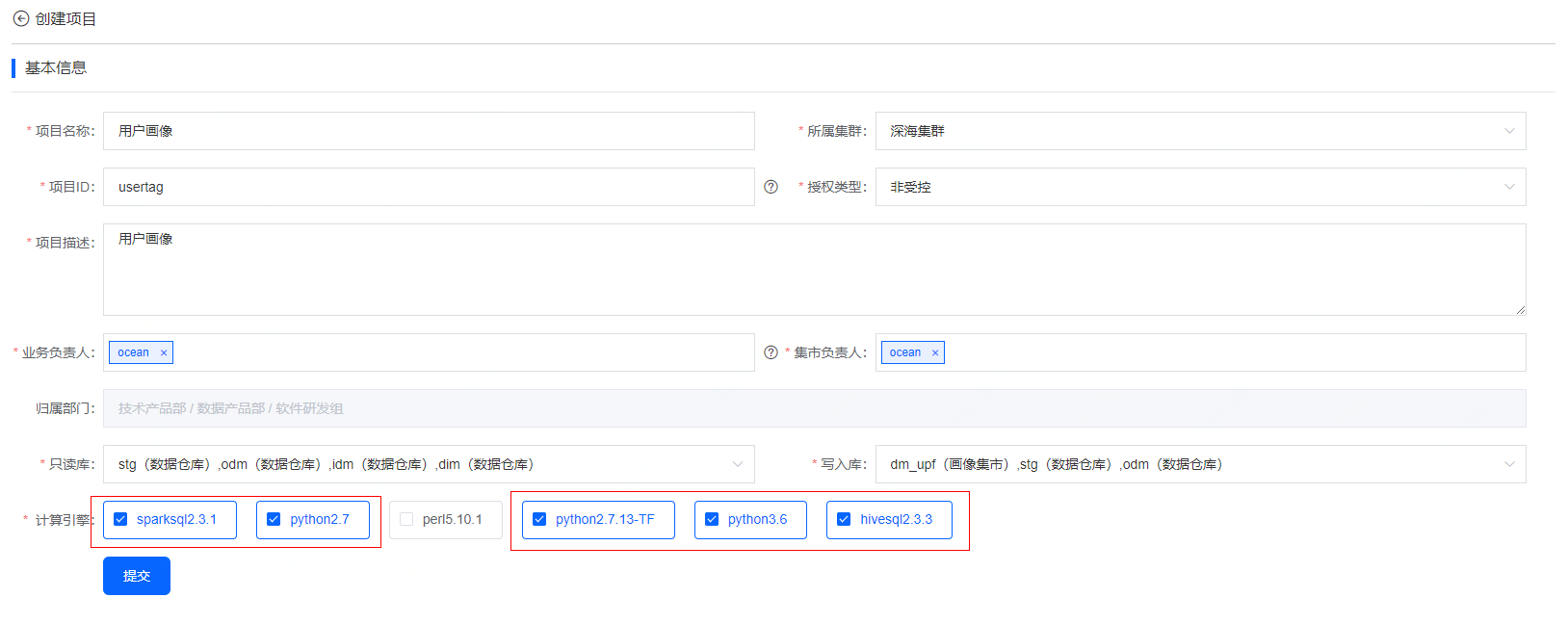
## 画像计算任务配置

根据任务的不同执行周期,配置任务定时执行

1.在大数据平台作业中心配置标签计算作业和偏移量计算任务

新建作业:

1).大数据平台-项目管理-全部项目-新建项目, 如下图:



其中计算引擎里必选标红出的5个

2).新建模型

数据模型-新建模型:



4).新建实体表

目前作业中心是一个作业对应一个表, 所以每建一个作业时需要先建一个表

表实体中文名和描述中添加改作业的相关关键信息即可

数据模型-新建实体表:



添加表字段:



5).新建作业

数据模型-新建作业:



填写参数,上传zip包:



(a)标签计算任务配置,如上图:

脚本类型:zip包

文件名称: tag\_context.sh(zip包里的执行文件名称)

参数: PROCESS\_库名.标签宽表名称

格式:前缀+库名.表名, 如: PROCESS\_dm\_upf.tag\_test\_merge\_3

命令:sh

文件上传: 文件ost\_tag\_bigdata-1.0-SNAPSHOT.zip(部署文档中离线计算模块里的zip包,确保包里相关配置已修改)



依赖作业:选择标签表的抽数作业, 若没有可暂时不配置

注意:1)标签计算作业需要每个合成宽表单独配置一个作业任务

2)作业的执行周期按照标签表的更新周期配置, 测试或者演示环境的标签数据一般不更新, 所以该任务可选择手动执行,避免由于环境不稳定导致执行失败,影响使用

(2)偏移量计算任务配置:

新建表并新建作业,与上图类似,区别如下

文件名称: push\_user\_context.sh

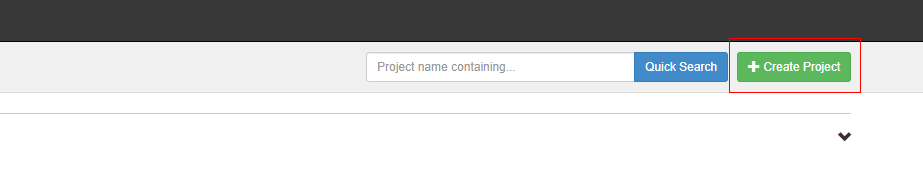
参数:库名.表名 如 dm\_upf. user\_sale\_tag\_pin2offset

依赖作业:选择偏移量表的数据计算作业, 若没有可暂时不配置

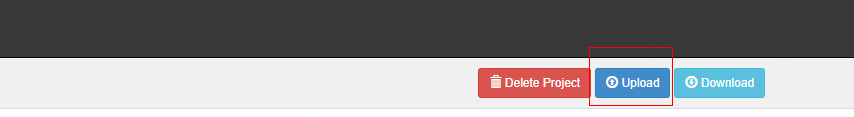
注意: 作业的执行周期按照pinoffset表的更新周期配置, 测试或者演示环境的pinoffset表数据一般不更新, 所以该任务可选择手动执行,避免由于环境不稳定导致执行失败,影响使用

2.在azkaban中配置人群创建,人群推送任务

1).登入azkaban,新建项目



2).上传zip包

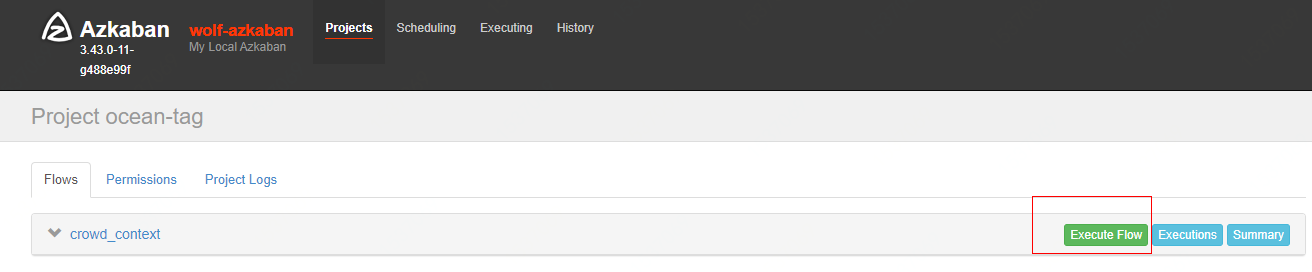


zip包格式:下载ost\_tag\_bigdata-1.0-SNAPSHOT.zip文件,取出以下3个文件,并确认principal和keytab已正确修改,选择2个文件一起压缩为zip文件, 命名为tag.zip, 然后点击Upload按钮上传

a).crowd\_context.job文件:人群生成任务, 建议配置执行周期为5到30分钟之间

b).push\_context.job文件:人群推送任务,建议配置执行周期为一小时或者一天

3).执行任务



点击Execute Flow按钮进入执行页面

Execute:可单次执行任务

Schedule:可配置定时任务

## 其它表规范

（1）idTypeMapping表, 不同类型id的映射关系表

注意:该表只作为测试使用,生产环境真正使用的表及数据是由数据分析师加工

表主键: jdpin

CREATE TABLE `dm\_upf\_id\_type\_mapping`(

`jdpin` string COMMENT '京东pin',

`idcard` string COMMENT '证件号',

`phone` string COMMENT '手机号',

`device` string COMMENT '设备号')

COMMENT 'idMapping表'

PARTITIONED BY (

`dt` string COMMENT '日期分区')

ROW FORMAT SERDE

'org.apache.hadoop.hive.ql.io.orc.OrcSerde'

STORED AS INPUTFORMAT

'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat';

该表需要通过作业中心导入数据, 并将数据推送到hbase中

以下为bhase中需要被推入数据的表及字段(列族为t),需要配置的推送

任务数与ID类型数一致,每种ID类型作为rowkey推送一份

a)user\_tag:cmb\_mapping\_jdpin表需推送全量字段

目标字段为:t:idcard, t:device, t:jdpin, t:phone

以jdpin作为rowkey, 选择md5加密

b)user\_tag:cmb\_mapping\_mobile\_no表只需推送phone,jdpin字段

目标字段为: t:phone, t:jdpin

以phone作为rowkey, 选择md5加密

c)user\_tag:cmb\_mapping\_id\_no表只需推送idcard,jdpin字段,

目标字段为: t:idcard, t:jdpin

以idcard作为rowkey, 选择md5加密

d)user\_tag:cmb\_mapping\_device\_no表只需推送device,jdpin字段,

目标字段为: t:device, t:jdpin

以device作为rowkey, 选择md5加密

（2）人群推送表规范, 人群推送功能使用

表名规范:表名中包含关键字: tag\_push

表结构:

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| id | string | 人群ID |
| userid | string | 用户ID |
| version | string | 人群版本号 |
| dt | string | 日期分区 |

建表测试语句参考如下:



上传人群需要在集群所有datanode节点创建 /export/data/tag-rule-download

## 进入门户

[http:// linking.101bank.sh/](http://ocean.jdfcloud.com/)



# 接口测试

## **1、判断人群命中接口**

说明：判断人群命中，入参： 用户PIN、人群编码（可多个），出参：返回人群是否命中

1)请求方式http：get

API：<http://ocean-tag-api.jdfcloud.com/operation/crowd/isHit>

(域名请以实际环境域名为准)

请求示例：http://ocean-tag-api.jdfcloud.com/operation/crowd/isHit?busId=jd\_6474cfff0d748&crowds=3d2730e4-8c9e-4e1f-b9df-ac37fa0feaea&token=175c0a45458d4870bbc1472c90a57e36

返回示例：{"stateCode":"3000","retMessage":"成功","result":"","data":{"3d2730e4-8c9e-4e1f-b9df-ac37fa0feaea":"1"}}

参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项** | **类型** | **是否可空** | **说明** |
| token | String | 不可空 | token认证 |
| busId | String | 不可空 | 京东用户PIN |
| crowds | String | 不可空 | 人群ID集合, 逗号分隔（必须是同一个主题下的人群，如果不是同一个主题下的人群只会以第一个人群的主题为准，其他主题的人群不查询） |

返回参数

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **数据项** | **数据项名称** | | **类型** | | **是否可空** | **说明** | |
| result | 无意义 | | String | | 空 |  | |
| stateCode | 状态码 | | String | | 不可空 | 3000: 成功  3001：第三方调用异常  3002：参数错误，无效的请求  3003：失败  3009：未知错误 | |
| retMessage | 描述信息 | | String | | 不可空 | 描述信息 | |
| data | 返回结果 | | T | | 可空 | 业务参数Map，详见：输出参数【data】说明 | |
| **数据项** | | **数据项名称** | | **类型** | | **可空** | **说明** |
| crowdId:value | | 人群ID:命中结果 | | Map<String,String> | | 否 | 输出规则ID，及其是否命中标识  1：命中，  0：不命中，  -1：人群不存在/人群未应用到命中服务  -2：pin不存在 |

## **2、标签取值接口**

说明：查询用户标签，入参：用户PIN、标签编码（多个），出参：返回用户PIN的标签值

1)请求方式http：get

API：<http://ocean-tag-api.jdfcloud.com/operation/tag/mappingTagValue>

(域名请以实际环境域名为准)

请求示例：http://ocean-tag-api.jdfcloud.com/operation/tag/mappingTagValue?busId=jd\_6bbde5a419e8b&tagCodes=tagOff0000461&token=574623c7782f48ac8c91952d5bbbd247&idType=1

返回示例：{"stateCode":"3000","retMessage":"成功","result":"","data":{"tagOff0000461":"5"}}

参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项** | 类型 | 是否可空 | 说明 |
| token | String | 不可空 | token认证 |
| busId | String | 不可空 | 京东用户PIN |
| tagCodes | String | 不可空 | 标签编码串,逗号分隔多个（必须是同一个主题下的标签，如果不是同一个主题下的人群只会以第一个标签的主题为准，其他主题的不查询） |
| idType | Integer | 不可空 | id类型,根据系统中实际支持的类型id确定,如测试数据  用户id的类型为1 |

返回参数

| **数据项** | **数据项名称** | | **类型** | | **可空** | **说明** | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| result | 无意义 | | String | | 可空 |  | |
| stateCode | 错误码 | | String | | 不可空 | 3000: 成功  3001：第三方调用异常  3002：参数错误，无效的请求  3003：失败  3009：未知错误 | |
| retMessage | 错误信息 | | String | | 不可空 | 描述信息 | |
| data | 返回结果 | | T | | 可空 | 业务参数Map，详见：输出参数说明 | |
| **数据项** | | **数据项名称** | | **类型** | | **可空** | **说明** | |
| tagCode：tagValue | | 标签编码：标签码值 | | Map<String,String> | | 否 | 如果入参“tagCode”有值，则按入参的标签输出指定的码值  注：标签值为"-9999",则表示标签值未知 | |

## **3、人群明细下载接口**

说明：下载人群明细，入参：人群编码、id偏移量，出参：返回用户PIN集合

1)请求方式http：get

API：<http://ocean-tag-api.jdfcloud.com/operation/crowd/download>

(域名请以实际环境域名为准)

请求示例：http://ocean-tag-api.jdfcloud.com/operation/crowd/download?startOffset=0&crowdCode=8e9715a0-9b19-4810-b44e-00a35efcf26e&token=7ac587ef2c984c40afbd7e04680ff549&num=5000

返回示例：{"stateCode":"3000","retMessage":"成功","result":"","data":{"total":2,"currentOffset":19,"ids":["105978104","1062698937\_m"]}}

参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项** | 类型 | 是否可空 | 说明 |
| token | String | 不可空 | token认证 |
| crowdCode | String | 不可空 | 要下载的人群编码（必须是同一个主题下的人群，如果不是同一个主题下的人群只会以第一个人群的主题为准，其他主题的人群不查询） |
| startOffset | Long | 不可空 | 人群中id的偏移量，初次请求时为0，下次请求时参考返回值中的  currentOffset（currentOffset + 1，即为下次请求的初始偏移量startOffset） |
| num | Integer | 可空 | 默认5000, 每次请求人群的数量,min:5000，max:10000 |

返回参数

| **数据项** | **数据项名称** | **类型** | **可空** | **说明** |
| --- | --- | --- | --- | --- |
| result | 无意义 | String | 可空 |  |
| stateCode | 错误码 | String | 不可空 | 3000: 成功  3001：第三方调用异常  3002：参数错误，无效的请求  3003：失败  3009：未知错误 |
| retMessage | 错误信息 | String | 不可空 | 描述 |
| data | 返回结果 | T | 可空 | 业务参数Map，详见：输出参数说明 |
| **数据项** | **数据项名称** | **类型** | **可空** | **说明** |
| total | 人群总数量 | long | 可空 |  |
| currentOffset | 本次查询id的最后一位偏移量 | long | 可空 | 返回的偏移量+1是下次请求下载同一个人群的startOffset |
| ids | ids集合 | List<String> | 可空 |  |