工业数字模型驱动引擎(iDME)

最佳实践

文档版本 13

发布日期 2024-06-24

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 基于智能边缘小站手动部署 iDME 应用

1.1 方案概述

应用场景

工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）的应用部署依赖于运行节点，其中数据交互依赖的中间件包括：云数据库、MongoDB、Redis、CSS、Kafka以及文件存储系统。因此，在部署iDME应用前需要确保中间件服务已完成部署，并且部署iDME应用的节点能够正常访问中间件服务。本解决方案能帮助用户快速在华为云上通过智能边缘小站（CloudPond）完成iDME应用部署。

资源环境示例

本方案部署iDME应用所需的CloudPond资源环境组成及版本说明如下：

表 1-1 部署资源

资源	版本	CPU（核数）	内存（G）	数据盘（G）	数量
iDME运行态实例	-	4	8	200	2
分布式消息服务 Kafka	2.3.0	4	8	200	3
云数据库 RDS for PostgreSQL	12.11	32	128	800	2
分布式缓存服务 Redis	5.0.14	4	16	200	2
文档数据库服务 DDS for MongoDB	社区版 4.0.28（自建集群）	4	8	1000	3

资源	版本	CPU（核数）	内存（G）	数据盘（G）	数量
弹性文件系统 SFS-Turbo（电子仓库）	-	-	-	2000	-
云搜索服务 CSS	7.9.3	8	16	1000	3

1.2 实施步骤

操作场景

本文以CentOS 7.6 64位操作系统的华为云弹性云服务器（Elastic Cloud Server，简称ECS）为例，基于智能边缘小站（CloudPond）部署工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）服务。

步骤 1：登录云服务器

登录待部署iDME的云服务器，具体操作请参见**Linux弹性云服务器登录方式概述**。

步骤 2：挂载文件系统

步骤1 执行如下命令，在云服务器上挂载SFS Turbo文件系统。

```
mount -t nfs -o vers=3,nolock 云服务器弹性公网IP地址:/ /mnt/sfs_turbo
```

步骤2 执行如下命令，赋予执行权限。

```
chmod +x /etc/rc.d/rc.local
```

步骤3 执行如下命令，打开并编辑rc.local配置文件。

```
vim /etc/rc.d/rc.local
```

步骤4 按“i”切换至编辑模式，并在最后一行添加如下命令。

```
mount -t nfs -o vers=3,nolock 云服务器弹性公网IP地址:/ /mnt/sfs_turbo
```

步骤5 按“Esc”，输入“:wq”，保存文件返回。

----结束

步骤 3：安装 JDK

步骤1 下载JDK 1.8版本的源码包，您可前往**Java SE 下载**页面选择需要的版本。

建议先将JDK源码包下载到本地，再上传至云服务器，否则会出现解压错误。具体操作请参见**上传文件到云服务器方式概览**。

步骤2 执行如下命令，新建JDK安装目录。

例如，JDK安装目录为“/opt/cloud/tenant-service”。

```
mkdir /opt/cloud/tenant-service
```

步骤3 执行如下命令，将JDK源码包解压到指定位置。

例如，将JDK源码包解压到“/opt/cloud/tenant-service”路径下。

```
tar -xvf jdk-8u221-linux-x64.tar.gz -C /opt/cloud/tenant-service
```

步骤4 执行如下命令，打开profile文件。

```
vim /etc/profile
```

步骤5 按“i”切换至编辑模式，根据您实际使用的JDK版本，在底部添加以下内容。

```
#set java environment
JAVA_HOME=/opt/cloud/tenant-service/jdk/jdk-18.0.x（您的JDK版本）
JRE_HOME=$JAVA_HOME PATH=$JAVA_HOME/bin:$PATH
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JRE_HOME/lib/tools.jar
export JAVA_HOME JRE_HOME PATH CLASSPATH
```

步骤6 按“Esc”，输入“:wq”，保存文件并返回。

步骤7 执行如下命令，读取环境变量。

```
source /etc/profile
```

步骤8 执行如下命令，查看JDK是否已经安装成功。

```
java -version
```

返回如下回显信息，则表示安装成功。

```
[root@ecs-c525-web ~]# java -version java version "18.0.3" 2022-01-18 LTS Java(TM) SE Runtime Environment (build 18.0.3+8-LTS-86) Java HotSpot(TM) 64-Bit Server VM (build 18.0.3+8-LTS-86, mixed mode, sharing)
```

----结束

步骤 4：配置 iDME 启动脚本

步骤1 执行如下命令，打开并编辑新创建的startxdm.bash配置文件。

```
vim startxdm.bash
```

步骤2 按“i”切换至编辑模式，将如下内容复制至startxdm.bash中，并将相关参数修改为您实际业务中的参数。

```
# 应用名
export APP_NAME=Demo
# 应用ID：格式（rdm_${应用ID}_app）
export application_subAppId=rdm_01a2b2c4764d4e00f123g345fd9baa9f_app
export BUCKET_NAME=bucket1
export CDM_CLUSTER_ID=67ad4594-d3a2-4c96-a40b-123a4bc5****
export CDM_END_POINT=http://cdm.cn-south-1.myhuaweicloud.com
export ES_PASSWORD=**********
# CSS端口号
export ES_PORT=9200
export ES_SCHEMA=http
# CSS的IP地址
export ES_URL=192.168.50.22,192.168.50.23,192.168.50.25
# CSS用户名
export ES_USERNAME=admin
export IAM_END_POINT=http://iam.cn-south-4.myhuaweicloud.com:31943
# KAFKA集群信息
export KAFKA_BOOTSTRAP_SERVERS=192.168.50.13:9092,192.168.50.14:9092,192.168.50.15:9092
export MONGODB_DATABASE=demoMongdb
# MongoDB连接URL
export MONGODB_URI=mongodb://192.168.50.19:27017,192.168.50.20:27017,192.168.50.21:27017
export MONGODB_INIT=false
export OBS_BUCKET_NAME=bucket1
export OBS_END_POINT=http://192.168.1.94:9000
export PROJECT_ID=0dc1a5fc3700d4b82f0ec1234f56****
# 数据库
export RDS_DATABASE=dme
# 数据库连接IP
export RDS_IP=192.168.50.104
# 数据库连接密码
```

```
export RDS_PASSWORD=**********
# 数据库端口号
export RDS_PORT=5432
# 数据库类型
export RDS_TYPE=postgresql
# Redis连接IP地址
export REDIS_HOST=192.168.50.201
# Redis密码
export REDIS_PASSWORD=********
# Redis端口
export REDIS_PORT=6379
export RES_AK=IJNHPNTX4TO6JHHZ****
export RES_SK=5LWOUtqyb3aGYmjyoQiDR8YG2Y0w8M9A6K0R****
# 是否开启https
export HTTPS_AUTH=false
# 忽略证书
export IIT_IGNORE_HTTPS_VERIFY=true
export rdm_dbType=postgresql
export iit_interceptors_auth_enable=false
export rdm_hibernate_dialect=com.huawei.it.rdm.configuration.XdmPostgresqlDialect
export debug=true
export server_port=8005
export server_ssl_enabled=false
export spring_datasource_driverClassName=org.postgresql.Driver
export spring_datasource_druid_master_url=jdbc:postgresql://${RDS_IP}:${RDS_PORT}/${RDS_DATABASE}?
stringtype=unspecified
export spring_datasource_druid_slave_url=jdbc:postgresql://${RDS_IP}:${RDS_PORT}/${RDS_DATABASE}?
stringtype=unspecified
export spring_datasource_username=root
export spring_profiles_active=impl
export application_tenantId=rdm_01a2b2c4764d4e00f123g345fd9baa9f_app
export server_servlet_context_path=/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services
export spring_redis_database=10
export IAM_LOGIN_ENDPOINT=test
# 文件系统类型
export FILE_STORAGE_TYPE=SFS

#nohup java -Diit.test=true  -jar microserviceTemplate.app-1.0.0-SNAPSHOT.jar >xdm.log 2>&1 &

START_CLASS=com.huawei.microservice.rdm.RdmApplication
PROCESS_PATH=/opt/cloud/tenant-service
LOAD_CLASSPATH=./lib/*:./

# 启动类
cd $PROCESS_PATH
BIND_ADDRESS=`hostname -I`
java -classpath $LOAD_CLASSPATH $START_CLASS --server.address=${BIND_ADDRESS} -Diit.test=true
>xdm.log 2>&1 &
```

步骤3 按“Esc”，输入“:wq”，保存文件并返回。

步骤4 执行如下命令，赋予执行权限。

```
chmod +x startxdm.bash
```

步骤5 执行如下命令，打开并编辑rc.local配置文件。

```
vim /etc/rc.d/rc.local
```

步骤6 按“i”切换至编辑模式，并在最后一行添加如下命令。

```
/opt/cloud/tenant-service/startxdm.bash
```

步骤7 按“Esc”，输入“:wq”，保存文件返回。

步骤8 执行如下命令，运行startxdm.bash配置文件。

```
/opt/cloud/tenant-service/startxdm.bash
```

步骤9 执行如下命令，查看是否存在iDME进程。

```
jps -l | grep com.huawei.microservice.rdm.RdmApplication
```

返回类似如下回显信息，则表示安装成功。

```
[root@ ~]# jps -l | grep com.huawei.microservice.rdm.RdmApplication
18759 com.huawei.microservice.rdm.RdmApplication
```

----结束

步骤 5：验证 iDME 服务

调用任意查询接口（例如TypeDefinition查询接口），如正常调用，则表示部署成功。

1.3 后续操作

完成iDME部署后，您可以进行如下操作。

启动 iDME 服务

iDME提供了两种启动方式。

- 方式一：通过脚本启动iDME

  iDME运行态包分为4个目录：bin、config、lib和suggestion_sql。其中，启动iDME服务涉及的jar包均在lib目录下。iDME的主启动类为com.huawei.microservice.rdm.RdmApplication，启动时需要导入必要的环境变量信息（包括依赖的中间件信息：云数据库、MongoDB、Redis、CSS、Kafka以及文件存储系统）。iDME提供启动脚本模板（startxdm.bash），该脚本默认加载路径为/opt/cloud/tenant-service。通过该脚本配置环境变量，即可通过如下命令一键式启动应用。

  ```
  ./startxdm.bash
  ```

- 方式二（推荐使用）：通过systemd命令启动iDME。

  执行如下命令，启动iDME服务。

  ```
  systemctl start dme.service
  ```

停止 iDME 服务

iDME提供了两种停止方式。

- 方式一：通过Linux命令结束iDME进程。

  a. 执行如下命令，获取当前服务的进程号（PID）。

  ```
  jps -l | grep com.huawei.microservice.rdm.RdmApplication
  ```

  b. 执行如下命令，结束iDME进程。

  ```
  kill PID
  ```

- 方式二（推荐使用）：通过systemd命令停止iDME。

  ```
  systemctl stop dme.service
  ```

监控 iDME 服务

- 通过控制台监控

  登录应用运维管理控制台，实时监控各云服务器的使用情况，包括CPU、内存、网络等各种看板，监控云服务器的负载水平。

- 通过云服务器监控

a. 登录云服务器。

b. 执行如下命令，进入arthas目录。

```
cd /opt/arthas
```

c. 执行如下命令，启动arthas，查看当前的Java进程，如图1-1所示。

```
java -jar arthas-boot.jar
```

图 1-1 iDME 服务进程

```
[root arthas]# java -jar arthas-boot.jar
[INFO] JAVA_HOME: /opt/jdk/jdk1.8/jre
[INFO] arthas-boot version: 3.6.7
[INFO] Found existing java process, please choose one and input the serial number of the process, eg : 1. Then hit ENTER.
* [1]: 1472 org.tanukisoftware.wrapper.WrapperSimpleApp
  [2]: 624 com.huawei.microservice.rdm.RdmApplication
2
```

d. 输入iDME服务进程（com.huawei.microservice.rdm.RdmApplication）所对应的数字，按“Enter”。例如2。

e. 执行如下命令，监控JVM状态（如内存使用情况、GC、线程等）。如图1-2所示。

```
dashboard
```

图 1-2 JVM 状态

```
[ ]$ dashboard
ID   NAME                                GROUP           PRIORITY   STATE      %CPU   DELTA_TIME   TIME       INTERRUPTED   DAEMON
-1                                       -               -1         -          0.0    0.000        0:53.788   false         true
-1                                       -               -1         -          0.0    0.000        0:50.252   false         true
-1                                       -               -1         -          0.0    0.000        0:49.621   false         true
121                                      main            5          RUNNABLE   0.0    0.000        0:44.956   false         false
-1                                       -               -1         -          0.0    0.000        0:17.560   false         true
54                                       main            5          WAITING    0.0    0.000        0:12.119   false         true
51                                       main            5          WAITING    0.0    0.000        0:10.990   false         true
57                                       main            5          WAITING    0.0    0.000        0:10.850   false         true
58                                       main            5          WAITING    0.0    0.000        0:10.597   false         true
50                                       main            5          WAITING    0.0    0.000        0:9.288    false         true
53                                       main            5          WAITING    0.0    0.000        0:9.227    false         true
49                                       main            5          WAITING    0.0    0.000        0:9.167    false         true
52                                       main            5          WAITING    0.0    0.000        0:8.955    false         true
56                                       main            5          WAITING    0.0    0.000        0:8.237    false         true
55                                       main            5          WAITING    0.0    0.000        0:7.673    false         true
-1                                       -               -1         -          0.0    0.000        0:3.524    false         true
-1                                       -               -1         -          0.0    0.000        0:3.484    false         true
-1                                       -               -1         -          0.0    0.000        0:3.470    false         true
-1                                       -               -1         -          0.0    0.000        0:3.462    false         true
-1                                       -               -1         -          0.0    0.000        0:3.459    false         true
-1                                       -               -1         -          0.0    0.000        0:3.448    false         true
-1                                       -               -1         -          0.0    0.000        0:3.436    false         true
Memory                    used    total   max     usage    GC
heap                      1599M   1824M   3531M   45.31%   gc.ps_scavenge.count              144
ps_eden_space             856M    1008M   1197M   71.59%   gc.ps_scavenge.time(ms)           3135
ps_survivor_space         56M     56M     56M     99.99%   gc.ps_marksweep.count             7
ps_old_gen                686M    760M    2648M   25.93%   gc.ps_marksweep.time(ms)          2099
nonheap                   376M    395M    -1      95.38%
code_cache                101M    102M    240M    42.46%
metaspace                 249M    263M    -1      94.38%
compressed_class_space    25M     28M     1024M   2.53%
direct                    3M      3M      -       100.00%
mapped                    0K      0K      -       0.00%
Runtime
os.name                                           Linux
os.version                                        3.10.0-1160.53.1.el7.x86_64
java.version                                      1.8.0_131
java.home                                         /opt/jdk/jdk1.8/jre
systemload.average                                0.13
```

1.4 故障处理

在CloudPond使用iDME时，可能会出现一些故障问题。本文介绍一些常见故障问题的排查及定位。

表 1-2 现象描述

现象	处理措施
在调用接口时，发现异常（如后端无响应但进程还在）。	• 方式一：执行**systemctl reload dme.service**命令，重启iDME服务。 • 方式二：先执行Kill命令结束iDME进程，再通过启动startxdm.bash脚本，启动iDME服务。
在监测时，发现JVM异常（如OOM（Out of Memory）等）。	

现象	处理措施
云服务器运行正常，且存在进程，但进程不可用。	执行**systemctl reload dme.service**命令，重启iDME服务。
云服务器宕机了。	1. 执行**reboot**命令，重启云服务器。 2. 执行**systemctl reload dme.service**命令，重启iDME服务。

如通过以上操作均无法解决您的问题，可将日志信息、发生问题时的操作内容、操作接口和参数、报错信息等信息反馈至华为云技术支撑人员。

1.5 常见问题

部署 iDME 时，MongoDB 连接失败怎么办?

由于CloudPond没有下沉MongoDB服务，因此MongoDB采用自建集群方式。当前支持4.x版本MongoDB，并生成类似于“mongodb://192.168.50.19:27017,192.168.50.20:27017,192.168.50.21:27017”格式的URL。若选择5.x版本，则会提示如下报错。

```
[cluster-ClusterId{value='6380a0889bcb6f2d7d2ba480',description='null'}-192.168.50.19: 27017] INFO
[org.mongodb.driver.cluster: info] [76] - Exception in monitor thread while connecting to server
192.168.50.19:27017com.mongodb.MongoSocketOpenException: Exception opening socket
ERROR [o.s.boot.SpringApplication: reportFailure] [830] - Application run failed
com.mongodb.MongoSocketReadException: Prematurely reached end of stream
```

部署 iDME 时，提示启动时间过长怎么办?

在启动Linux服务器时，由于安全随机数导致的线程阻塞，会使得启动时间过长（半小时左右）。因此在CloudPond启动iDME时，需要将安全随机数生成函数由Windows下的“SecureRandom secureRandom = SecureRandom.getInstanceStrong();”调整为：“SecureRandom secureRandom = SecureRandom.getInstance("NativePRNGNonBlocking");”。

综合后可调整为：

```
SecureRandom secureRandom = null;
if (isWindows()) {
    secureRandom = SecureRandom.getInstanceStrong();
} else if (isLinux()) {
    secureRandom = SecureRandom.getInstance("NativePRNGNonBlocking");
}
```

部署 iDME 后，其接口调用方式是什么?

前缀采用*http://${iDME服务器IP地址}/rdm_应用ID_app/services*，后缀再拼接需要调用的接口mapping地址。

例如调用获取所有非内部的数据实体模型的接口，采用前缀拼接上后缀“/rdm/basic/api/DataModelManagement/getAllDMEntity”，即调用：

“http://${iDME服务器IP地址}/rdm_应用ID_app/services/rdm/basic/api/DataModelManagement/getAllDMEntity”。

2 xDM-F 的查询相关接口

2.1 查询接口概述

数据实体和关系实体是工业数字模型驱动引擎-数据建模引擎（xDM Foundation，简称xDM-F）原子接口的承载体，支持多种具有查询功能的接口。其中：

- 数据实体支持调用find、get、batchget、query、count、list和select接口，可进行单字段排序或多字段排序。
  - 仅支持单字段排序的接口有Get、Batchget和List接口。
  - 支持多字段排序的接口有Find、Query、Count和Select接口。
- 关系实体支持queryRelatedObjects（查询源或目标实体的属性）、batchQueryRelatedObjects（批量查询源或目标实体的属性）、queryRelationship（查询关系实体的数据）、queryTarget（查询关系实例的数据）和deleteTarget（删除关系实例）接口。

功能对比

表 2-1 数据实体查询接口

接口	适用场景	查询效率	多字段排序
Get	适用于通过实体或实例的ID，获取某实体或实例所有信息的场景。	模型越复杂，参考对象和扩展属性越多，查询效率越慢。	不支持。
Batchget	适用于通过实体或实例的ID，获取多个实体或实例所有信息的场景。	模型越复杂，参考对象和扩展属性越多，查询效率越慢。	不支持。
List	适用于只查询数据模型自身信息的场景。 不支持参考模型属性作为查询条件。	查询效率较快。	可通过sorts字段进行多字段排序，通过filter字段进行数据过滤。

接口	适用场景	查询效率	多字段排序
Find	适用于通过指定查询条件，获取符合条件的所有数据模型的所有信息的场景。	模型越复杂，参考对象和扩展属性越多，查询效率越慢。	可通过sorts字段进行多字段排序，通过filter字段进行数据过滤。
Query	适用于只查询数据模型所有的列表属性信息的场景。 只返回符合查询条件的对象及列表属性。	查询效率相较于find更快一些。	可通过sorts字段进行多字段排序，通过filter字段进行数据过滤。
Count	适用于通过指定查询条件，获取符合条件的数据模型总数的场景。 只返回符合查询条件的对象的记录总数。	查询效率较快。	可通过sorts字段进行多字段排序，通过filter字段进行数据过滤。
Select	适用于只查询数据模型指定属性数据的场景。	查询效率较快。	可通过sorts字段进行多字段排序，通过filter字段进行数据过滤。但数据过滤时，还需通过selectedField字段指定属性查询数据。

说明

如果分页偏移量＞50000，受限于数据库分页查询效能，可能存在响应时长增加的情况，为保障查询体验，建议list、find、query和select等接口的分页偏移量≤50000。

表 2-2 关系实体查询接口

接口	描述
queryRelatedObjects	用于查询源或目标实体的属性。
batchQueryRelatedObjects	用于批量查询源或目标实体的属性。
queryRelationship	用于查询关系实例的数据。
queryTarget	用于查询目标实体的数据。
deleteTarget	用于删除关系实例。

使用方法

- 关于数据实体的查询接口的使用和代码示例，请参见**数据实体查询接口**。

- 关于关系实体的查询接口的使用和代码示例，请参见**关系实体查询接口**。

排序规则说明

数据实体、关系实体的接口，涉及排序的，根据数据库类型不同，字段首字母排序遵循如下规则：

- 如果是PostgreSQL数据库，在升降序排序时，先根据字母大小写，然后再根据英文字母顺序进行升降序排序显示。如升序排序时，字段首字母排序显示为B、X、Z、a、b、c。
- 如果是MySQL数据库，在升降序排序时，先根据英文字母顺序，然后再根据字母大小写进行升降序排序显示。如升序排序时，字段首字母排序显示为a、B、b、c、X、Z。

支持的运算符

查询接口支持如下运算符：

- =：支持设置文本、整型、长整型、浮点型、浮点型（自定义精度）、布尔值、枚举、人员、URL和日期类型的属性。
- like：支持设置文本、人员和URL类型的属性。
- startWith：支持设置文本、人员和URL类型的属性。
- endWith：支持设置文本、人员和URL类型的属性。
- in：支持设置文本、整型、长整型、浮点型、浮点型（自定义精度）、布尔值、日期、枚举、人员和URL类型的属性。
- <：支持设置整型、长整型、浮点型、浮点型（自定义精度）和日期类型的属性。
- >：支持设置整型、长整型、浮点型、浮点型（自定义精度）和日期类型的属性。
- <=：支持设置整型、长整型、浮点型、浮点型（自定义精度）和日期类型的属性。
- >=：支持设置整型、长整型、浮点型、浮点型（自定义精度）和日期类型的属性。
- <>：支持设置文本、整型、长整型、浮点型、浮点型（自定义精度）、布尔值、枚举、人员、URL和日期类型的属性。
- ISNULL：无需设置属性。
- NOTNULL：无需设置属性。

2.2 数据实体查询接口

2.2.1 Get

功能介绍

通过填写数据实例/实体的ID，获取该数据实例/实体上的所有信息。

入参

POST http://*{Endpoint}*/*rdm_{appID}_app*/services/dynamic/api/*{entityName}*/get
{

```
  "params": {
    "id": "XXXXXXXXXX"
  }
}
```

其中，{Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。

出参

返回模型所有属性、直接关联的参考对象、扩展属性、分类属性、级联的数据等。

示例场景

有一个实例（People），ID为455304697733976064。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/People/get
{
  "params": {
    "id": "455304697733976064"
  }
}
```

出参示例

```
{
    "result": "SUCCESS",
    "data": [
        {
            "id": "455304697733976064",
            "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
            "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
            "createTime": "2023-01-13T01:38:19.000+00:00",
            "lastUpdateTime": "2023-01-13T01:38:19.000+00:00",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "People",
            "tenant": {
                "id": "-1",
                "creator": "xdmAdmin",
                "modifier": "xdmAdmin",
                "createTime": "2022-08-03T11:27:44.000+00:00",
                "lastUpdateTime": "2022-08-03T11:27:44.000+00:00",
                "rdmVersion": 1,
                "rdmDeleteFlag": 0,
                "rdmExtensionType": "Tenant",
                "tenant": null,
                "className": "Tenant",
                "name": "basicTenant",
                "description": "默认租户",
                "kiaguid": null,
                "securityLevel": "internal",
                "code": "basicTenant",
                "disableFlag": false,
                "dataSource": null
            },
            "className": "People",
            "name": "李兰",
            "description": null,
            "kiaguid": null,
            "securityLevel": "internal",
            "sex": "女",
            "age": 18
```

```
    }
  ],
  "errors": []
}
```

2.2.2 BatchGet

功能介绍

通过填写一个或多个数据实体/实例ID，获取这些实例/实体上的所有信息。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/batchGet
{
  "params": {
    "ids": [
      "XXXXXXXXXX",
      "XXXXXXXXXX"
    ]
  }
}
```

其中，{Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID。

出参

返回模型所有属性、直接关联的参考对象、扩展属性、分类属性、级联的数据等。

示例场景

有两个实例，ID分别为455304697733976064和455304645330341888。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/
dynamic/api/batchGet
  "params": {
    "ids": [
      "455304697733976064",
      "455304645330341888"
    ]
  }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "455304645330341888",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:38:07.000+00:00",
      "lastUpdateTime": "2023-01-13T01:38:07.000+00:00",
      "rdmVersion": 1,
      "rdmDeleteFlag": 0,
      "rdmExtensionType": "People",
      "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "modifier": "xdmAdmin",
```

```
            "createTime": "2022-08-03T11:27:44.000+00:00",
            "lastUpdateTime": "2022-08-03T11:27:44.000+00:00",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "kiaguid": null,
            "securityLevel": "internal",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": null
        },
        "className": "People",
        "name": "李四",
        "description": null,
        "kiaguid": null,
        "securityLevel": "internal",
        "sex": "男",
        "age": 20
    },
    {
        "id": "455304697733976064",
        "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
        "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
        "createTime": "2023-01-13T01:38:19.000+00:00",
        "lastUpdateTime": "2023-01-13T01:38:19.000+00:00",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "People",
        "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "modifier": "xdmAdmin",
            "createTime": "2022-08-03T11:27:44.000+00:00",
            "lastUpdateTime": "2022-08-03T11:27:44.000+00:00",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "kiaguid": null,
            "securityLevel": "internal",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": null
        },
        "className": "People",
        "name": "李兰",
        "description": null,
        "kiaguid": null,
        "securityLevel": "internal",
        "sex": "女",
        "age": 18
    }
  ],
  "errors": []
}
```


2.2.3 List

功能介绍

List接口用于获取自身实例/实体的属性信息。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/list/pageSize/curPage
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "属性名称"
         }

      ],
      "filter": {
         ......
      },
      "isNeedTotal": true
   }
}
```

- {Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。
- 在URL上填写待查询的页码（ curPage ）和每页可显示的数据量（ pageSize ）。
- 在JSON代码中的设置sorts字段和filter字段。
  - sorts：填写需要按哪个字段进行排序，可为空。
  - filter：填写过滤条件，可为空。

出参

仅返回本模型的属性信息。如果属性的类型为参数对象，只返回ID和clazz。

示例场景

有一个实例（ People ），先按名称倒序排序，再按年龄倒序排序。而后根据性别为男性进行过滤。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/People/list/20/1
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "name"
         },
         {
            "sort": "DESC",
            "orderBy": "age"
         }
      ],
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "sex",
               "operator": "=",
               "conditionValues": [
                  "男"
               ]
            }
         ]
      },
```

```
    "isNeedTotal": true
  }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "455304645330341888",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:38:07.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:38:07.000+00:00",
      "rdmVersion": 1,
      "rdmExtensionType": "People",
      "rdmDeleteFlag": 0,
      "tenant": {          //tenant为参考对象
        "id": "-1",
        "clazz": "Tenant"
      },
      "className": "People",
      "name": "李四",
      "description": null,
      "kiaguid": null,
      "securityLevel": "internal",
      "sex": "男",
      "age": 20
    },
    {
      "id": "455304534248394752",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:37:40.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:37:40.000+00:00",
      "rdmVersion": 1,
      "rdmExtensionType": "People",
      "rdmDeleteFlag": 0,
      "tenant": {          //tenant为参考对象
        "id": "-1",
        "clazz": "Tenant"
      },
      "className": "People",
      "name": "张三",
      "description": null,
      "kiaguid": null,
      "securityLevel": "internal",
      "sex": "男",
      "age": 18
    }
  ],
  "errors": [],
  "pageInfo": {
    "curPage": 1,
    "pageSize": 20,
    "totalRows": 2,
    "totalPages": 1
  }
}
```

2.2.4 Find

功能介绍

根据指定条件查询，返回数据模型的所有属性及关联的信息。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/find/pageSize/curPage
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "属性名称"
         }

      ],
      "filter": {
        ......
      },
      "isNeedTotal": true
   }
}
```

- {Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。
- 在URL上填写待查询的页码（curPage）和每页可显示的数据量（pageSize）。
- 在JSON代码中的设置sorts字段和filter字段。
  - sorts：填写需要按哪个字段进行排序，可填写模型自身属性、参考对象的属性、扩展属性及分类属性，也可为空。
  - filter：填写过滤条件，可为空。

出参

返回模型所有属性、直接关联的参考对象、扩展属性、分类属性、级联的数据等。

示例场景

有一个实例（People），先按名称倒序排序，再按年龄倒序排序。而后根据性别为男性进行过滤。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/People/find/20/1
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "name"
         },
         {
            "sort": "DESC",
            "orderBy": "age"
         }
      ],
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "sex",
               "operator": "=",
               "conditionValues": [
                  "男"
               ]
            }
         ]
```

```
    },
    "isNeedTotal": true
  }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "455304645330341888",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:38:07.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:38:07.000+00:00",
      "rdmVersion": 1,
      "rdmExtensionType": "People",
      "rdmDeleteFlag": 0,
      "tenant": {
        "id": "-1",
        "clazz": "Tenant"
      },
      "className": "People",
      "name": "李四",
      "description": null,
      "kiaguid": null,
      "securityLevel": "internal",
      "sex": "男",
      "age": 20
    },
    {
      "id": "455304534248394752",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:37:40.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:37:40.000+00:00",
      "rdmVersion": 1,
      "rdmExtensionType": "People",
      "rdmDeleteFlag": 0,
      "tenant": {
        "id": "-1",
        "clazz": "Tenant"
      },
      "className": "People",
      "name": "张三",
      "description": null,
      "kiaguid": null,
      "securityLevel": "internal",
      "sex": "男",
      "age": 18
    }
  ],
  "errors": [],
  "pageInfo": {
    "curPage": 1,
    "pageSize": 20,
    "totalRows": 2,
    "totalPages": 1
  }
}
```

2.2.5 Query

功能介绍

根据指定条件查询，返回数据模型的所有“列表属性”。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/query/pageSize/curPage
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "属性名称"
         }

      ],
      "filter": {
        ......
      },
      "isNeedTotal": true
   }
}
```

- {Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。
- 在URL上填写待查询的页码（curPage）和每页可显示的数据量（pageSize）。
- 在JSON代码中的设置sorts字段和filter字段。
  - sorts：填写需要按哪个字段进行排序，可填写模型自身属性、参考对象的属性、扩展属性及分类属性，也可为空。
  - filter：填写过滤条件，可为空。

出参

返回模型所有列表属性。

示例场景

有一个实例（People），先按名称倒序排序，再按年龄倒序排序。而后根据性别为男性进行过滤。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/
dynamic/api/People/query/20/1
{
   "params": {
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "name"
         },
         {
            "sort": "DESC",
            "orderBy": "age"
         }
      ],
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "sex",
               "operator": "=",
               "conditionValues": [
                  "男"
               ]
            }
         ]
```

```
    },
    "isNeedTotal": true
  }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "455304645330341888",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:38:07.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:38:07.000+00:00",
      "rdmExtensionType": "People",
      "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "createTime": "2022-08-03T11:27:44.000+00:00",
        "modifier": "xdmAdmin",
        "lastUpdateTime": "2022-08-03T11:27:44.000+00:00",
        "rdmExtensionType": "Tenant",
        "tenant": null,
        "className": "Tenant",
        "name": "basicTenant",
        "description": "默认租户",
        "code": "basicTenant",
        "disableFlag": false,
        "dataSource": null
      },
      "className": "People",
      "name": "李四",
      "description": null,
      "age": 20
    },
    {
      "id": "455304534248394752",
      "creator": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "createTime": "2023-01-13T01:37:40.000+00:00",
      "modifier": "test1 3c03e719256a427eb9277b64fcXXXXXX",
      "lastUpdateTime": "2023-01-13T01:37:40.000+00:00",
      "rdmExtensionType": "People",
      "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "createTime": "2022-08-03T11:27:44.000+00:00",
        "modifier": "xdmAdmin",
        "lastUpdateTime": "2022-08-03T11:27:44.000+00:00",
        "rdmExtensionType": "Tenant",
        "tenant": null,
        "className": "Tenant",
        "name": "basicTenant",
        "description": "默认租户",
        "code": "basicTenant",
        "disableFlag": false,
        "dataSource": null
      },
      "className": "People",
      "name": "张三",
      "description": null,
      "age": 18
    }
  ],
  "errors": [],
  "pageInfo": {
    "curPage": 1,
    "pageSize": 20,
```

```
    "totalRows": 2,
    "totalPages": 1
  }
}
```

2.2.6 Count

功能介绍

根据过滤条件查询，统计数据实体/实例的总数。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/count
{
  "params": {
    "filter": {
      "joiner": "and",
      "conditions": [
        {
          "conditionName": "属性名称",
          "operator": "=",
          "conditionValues": [
            "属性值"
          ]
        }
      ]
    },
    "isNeedTotal": true
  }
}
```

- {Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。
- filter：填写过滤条件，可填写模型自身属性、参考对象的属性、扩展属性及分类属性。

出参

总记录数。

示例场景

有一个实例（People），根据年龄，为性别为男性的数据进行过滤。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/People/count
{
  "params": {
    "filter": {
      "joiner": "and",
      "conditions": [
        {
          "conditionName": "sex",
          "operator": "=",
          "conditionValues": [
            "男"
          ]
        }
      ]
```

```
        },
        "isNeedTotal": true
    }
}
```

出参示例

```
{
"result": "SUCCESS",
    "data": [           //data为返回结果总数
        2
    ],
    "errors": []
}
```

2.2.7 Select

功能介绍

指定需要的属性及查询条件，返回指定属性的结果集。

入参

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/select/pageSize/curPage
{
    "params": {
        "selectedField": [
            {
                "name": "属性名称",
                "nameAs": "属性别名"
            }

        ],
        "sorts": [
            {
                "sort": "DESC",
                "orderBy": "属性名称"
            }

        ],
        "filter": {
            "joiner": "and",
            "conditions": [
                {
                    "conditionName": "属性名称",
                    "operator": "=",
                    "conditionValues": [
                        "属性值"
                    ]
                }
            ]
        }
    }
}
```

- {Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称。
- 在URL上填写待查询的页码（curPage）和每页可显示的数据量（pageSize）。
- 在JSON代码中的设置sorts字段、filter字段和SelectedField字段。
  - sorts：填写需要按哪个字段进行排序，可填写模型自身属性、参考对象的属性、扩展属性及分类属性，也可为空。
  - filter：填写过滤条件，可为空。

- SelectedField：填写需要输出的字段和别名。JSON参数格式为“参数名称.参数值”，如果填写的字段为JSON类型，SelectedField的“name”参数需要填写为“JSON类型字段的英文名称.JSON参数的名称”。例如，需要输出的JSON类型字段为“partJSON”，存在一条数据**[{"city":"shenzhen"}]**，SelectedField填写示例如下：

```
"selectedField": [
    {
      "name": "partJSON.city",
      "nameAs": "partJSON"
    }
  ]
```

出参

指定属性的结果集。

示例 1

示例场景

有一个数据实体（People），先按名称倒序排序，再按年龄倒序排序。而后根据性别为男性进行过滤，且“SelectedField”字段只选择输出名称和年龄，别名为“nameAsName”和“nameAsAge”。

入参示例

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/People/select/20/1
{
   "params": {
      "selectedField": [
         {
            "name": "name",
            "nameAs": "nameAsName"
         },
         {
            "name": "age",
            "nameAs": "nameAsAge"
         }
      ],
      "sorts": [
         {
            "sort": "DESC",
            "orderBy": "name"
         },
         {
            "sort": "DESC",
            "orderBy": "age"
         }
      ],
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "sex",
               "operator": "=",
               "conditionValues": [
                  "男"
               ]
            }
         ]
      }
   }
}
```

出参示例

```
{
   "result": "SUCCESS",
   "data": [
      {
         "nameAsAge": 20,
         "nameAsName": "李四"
      },
      {
         "nameAsAge": 18,
         "nameAsName": "张三"
      }
   ],
   "errors": [],
   "pageInfo": {
      "curPage": 1,
      "pageSize": 20,
      "totalRows": 0,
      "totalPages": 0
   }
}
```

示例 2

示例场景

有一个数据实体（XxDMETest），该实体包含一个参考对象类型的属性（testDme）。而后根据testDme的ID，查询该属性ID的所有信息，并按ID倒序排序。

入参示例

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/XxDMETest/select/20/1
{
   "params": {
      "selectedField": [
         {
            "name" : "testDme"
         }
      ],
      "orderBy": "id",
      "sort": "DESC",
      "isNeedTotal": true,
      "filter": {
         "conditions": [
            {
               "conditionName": "id",
               "operator": "=",
               "conditionValues": [
                  "507612209518485504"
               ]
            }
         ],
         "joiner": "and"
      }
   }
}
```

出参示例

```
{
   "result": "SUCCESS",
   "data": [
      {
         "testDme": {
            "id": "111",
            "creator": "XDM_Developer 93172bbfd0f64437956d4c9de1234567",
```

```
            "modifier": "XDM_Developer 93172bbfd0f64437956d4c9de1234567",
            "createTime": "2023-06-06T07:32:59.629+0000",
            "lastUpdateTime": "2023-06-06T07:32:59.629+0000",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "XxxDME",
            "tenant": {
               "id": "-1",
               "creator": "xdmAdmin",
               "modifier": "xdmAdmin",
               "createTime": "2023-02-08T06:38:48.567+0000",
               "lastUpdateTime": "2023-02-08T06:38:48.567+0000",
               "rdmVersion": 1,
               "rdmDeleteFlag": 0,
               "rdmExtensionType": "Tenant",
               "tenant": null,
               "className": "Tenant",
               "name": "basicTenant",
               "description": "默认租户",
               "kiaguid": null,
               "securityLevel": "internal",
               "code": "basicTenant",
               "disableFlag": false,
               "dataSource": "DefaultDataSource"
            },
            "className": "XxxDME"
         }
      }
   ],
   "errors": [],
   "pageInfo": {
      "curPage": 1,
      "pageSize": 20,
      "totalRows": 1,
      "totalPages": 1
   }
}
```


2.3 关系实体查询接口

2.3.1 queryRelatedObjects

功能介绍

queryRelatedObjects接口可根据入参，查询某源/目标数据模型的信息和属性。

例如，入参为源数据模型，可查询目标数据模型信息和属性。

入参

```
{
    "params":{
          "objectId":XXXX,
          "role":"target",
          "latestOnly":false
    }
}
```

- objectId：对象ID。
- role：角色，源数据模型或目标数据模型。
- latestOnly：目标对象是否仅返回源对象关联的最新版本目标对象，默认为false。（仅对M-V有效，即返回所有版本）

出参

返回查询结果列表。

示例场景

假设有一个源端为RelationLeft，目标端为RelationRight的关系实体（RelationTest）。其中，

- RelationLeft有两个关系实例，唯一编码为454580805678901111和454580805678902222。
- RelationRight有两个关系实例，唯一编码为454580805678903333和454580805678904444。

并创建了如下唯一编码的关系实例：

- 1313：源端为454580805678901111，目标端为454580805678903333。
- 2424：源端为454580805678902222，目标端为454580805678904444。
- 1414：源端为454580805678901111，目标端为454580805678904444。

入参示例

根据目标端RelationRight的ID为454580805678903333的关系实例，查询符合条件的源端关系实例。

```
{
    "params":{
          "objectId":454580805678903333,
          "role":"target",
          "latestOnly":false
    }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "454580805678901111",
      "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
      "createTime": "2023-01-12T11:55:22.797+0000",
      "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
      "lastUpdateTime": "2023-01-12T11:55:22.797+0000",
      "rdmExtensionType": "RelationLeft",
      "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "createTime": "2022-09-22T04:10:48.543+0000",
        "modifier": "xdmAdmin",
        "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
        "rdmExtensionType": "Tenant",
        "tenant": null,
        "className": "Tenant",
        "name": "basicTenant",
        "description": "默认租户",
        "code": "basicTenant",
        "disableFlag": false,
        "dataSource": "DefaultDataSource"
      },
      "className": "RelationLeft",
      "name": null,
```

```
            "description": null
        }
    ],
    "errors": [],
    "pageInfo": {
        "curPage": 1,
        "pageSize": 20,
        "totalRows": 1,
        "totalPages": 1
    }
}
```

2.3.2 batchQueryRelatedObjects

功能介绍

batchQueryRelatedObjects接口可根据入参，批量查询某源/目标数据模型的信息以及对应的属性列表。

例如，入参为源数据模型，可批量查询目标数据模型信息和属性。

入参

```
{
    "params":{
            "objectIds": [ XXXX, XXXX ],
            "role":"source",
            "latestOnly":false
    }
}
```

- objectIds：对象ID列表。
- role：角色，源数据模型或目标数据模型。
- latestOnly：目标对象是否仅返回源对象关联的最新版本目标对象，默认为false。（仅对M-V模型有效，即返回所有版本）

出参

返回查询结果列表。

示例场景

假设有一个源端为RelationLeft，目标端为RelationRight的关系实体（RelationTest）。其中，

- RelationLeft有两个关系实例，唯一编码为454580805678901111和454580805678902222。
- RelationRight有两个关系实例，唯一编码为454580805678903333和454580805678904444。

并创建了如下唯一编码的关系实例：

- 1313：源端为454580805678901111，目标端为454580805678903333。
- 2424：源端为454580805678902222，目标端为454580805678904444。
- 1414：源端为454580805678901111，目标端为454580805678904444。

入参示例

根据源端RelationRight的ID为454580805678901111和454580805678902222的对象实例查询符合条件的目标端对象实例。

```
{
    "params":{
         "objectIds": [ 454580805678901111, 454580805678902222 ],
         "role":"source",
         "latestOnly":false
    }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "objectId": 454580805678902222,
      "relatedList": [
        {
          "id": "454580805678904444",
          "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
          "createTime": "2023-01-12T11:55:47.918+0000",
          "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
          "lastUpdateTime": "2023-01-12T11:55:47.918+0000",
          "rdmExtensionType": "RelationRight",
          "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "modifier": "xdmAdmin",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": "DefaultDataSource"
          },
          "className": "RelationRight",
          "name": null,
          "description": null
        }
      ]
    },
    {
      "objectId": 454580805678901111,
      "relatedList": [
        {
          "id": "454580805678903333",
          "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
          "createTime": "2023-01-12T11:55:43.192+0000",
          "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
          "lastUpdateTime": "2023-01-12T11:55:43.192+0000",
          "rdmExtensionType": "RelationRight",
          "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "modifier": "xdmAdmin",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
```

```
                    "name": "basicTenant",
                    "description": "默认租户",
                    "code": "basicTenant",
                    "disableFlag": false,
                    "dataSource": "DefaultDataSource"
                },
                "className": "RelationRight",
                "name": null,
                "description": null
            },
            {
                "id": "454580805678904444",
                "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
                "createTime": "2023-01-12T11:55:47.918+0000",
                "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
                "lastUpdateTime": "2023-01-12T11:55:47.918+0000",
                "rdmExtensionType": "RelationRight",
                "tenant": {
                    "id": "-1",
                    "creator": "xdmAdmin",
                    "createTime": "2022-09-22T04:10:48.543+0000",
                    "modifier": "xdmAdmin",
                    "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
                    "rdmExtensionType": "Tenant",
                    "tenant": null,
                    "className": "Tenant",
                    "name": "basicTenant",
                    "description": "默认租户",
                    "code": "basicTenant",
                    "disableFlag": false,
                    "dataSource": "DefaultDataSource"
                },
                "className": "RelationRight",
                "name": null,
                "description": null
            }
        ]
    }
  ],
  "errors": [],
  "pageInfo": {
    "curPage": 1,
    "pageSize": 20,
    "totalRows": 3,
    "totalPages": 1
  }
}
```


2.3.3 queryRelationship

功能介绍

queryRelationship接口可根据源/目标数据模型的ID和角色，查询关系实体的实例数据列表（实例中包含源和目标的信息）。

入参

```
{
    "params":{
        "objectId": XXXX,
        "role":"source",
        "latestOnly":false
    }
}
```

- objectId：对象ID。

- role：角色，源数据模型或目标数据模型。
- latestOnly：目标对象是否仅返回源对象关联的最新版本目标对象，默认为false。（仅对M-V模型有效，即返回所有版本）

出参

返回关系实例的查询结果列表。

示例场景

假设有一个源端为RelationLeft，目标端为RelationRight的关系实体（RelationTest）。其中，

- RelationLeft有两个关系实例，唯一编码为454580805678901111和454580805678902222。
- RelationRight有两个关系实例，唯一编码为454580805678903333和454580805678904444。

并创建了如下唯一编码的关系实例：

- 1313：源端为454580805678901111，目标端为454580805678903333。
- 2424：源端为454580805678902222，目标端为454580805678904444。
- 1414：源端为454580805678901111，目标端为454580805678904444。

入参示例

根据源端RelationRight的ID为454580805678901111的关系实例，查询符合条件的全部数据。

```
{
    "params":{
        "objectId":454580805678901111,
        "role":"source",
        "latestOnly":false
    }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "454580805678901111",
      "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
      "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
      "createTime": "2023-01-12T11:56:55.551+0000",
      "lastUpdateTime": "2023-01-12T11:56:55.551+0000",
      "rdmVersion": 1,
      "rdmDeleteFlag": 0,
      "rdmExtensionType": "RelationTest",
      "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "modifier": "xdmAdmin",
        "createTime": "2022-09-22T04:10:48.543+0000",
        "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "Tenant",
```

```
        "tenant": null,
        "className": "Tenant",
        "name": "basicTenant",
        "description": "默认租户",
        "kiaguid": null,
        "securityLevel": "internal",
        "code": "basicTenant",
        "disableFlag": false,
        "dataSource": "DefaultDataSource"
    },
    "className": "RelationTest",
    "source": {
        "id": "454580805678901111",
        "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "createTime": "2023-01-12T11:55:22.797+0000",
        "lastUpdateTime": "2023-01-12T11:55:22.797+0000",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "RelationLeft",
        "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "modifier": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "kiaguid": null,
            "securityLevel": "internal",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": "DefaultDataSource"
        },
        "className": "RelationLeft",
        "name": null,
        "description": null,
        "kiaguid": null,
        "securityLevel": "internal",
        "stuId": null,
        "right": {
            "id": "427473106174128128",
            "clazz": "RelationRight"
        },
        "stuName": null
    },
    "target": {
        "id": "454580805678903333",
        "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "createTime": "2023-01-12T11:55:43.192+0000",
        "lastUpdateTime": "2023-01-12T11:55:43.192+0000",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "RelationRight",
        "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "modifier": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "Tenant",
```

```
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "kiaguid": null,
            "securityLevel": "internal",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": "DefaultDataSource"
        },
        "className": "RelationRight",
        "name": null,
        "description": null,
        "kiaguid": null,
        "securityLevel": "internal",
        "stuId": null,
        "courseId": null,
        "relationLeftList": null
    },
    "name": null,
    "description": null
},
{
    "id": "455098158066700288",
    "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
    "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
    "createTime": "2023-01-12T11:57:36.198+0000",
    "lastUpdateTime": "2023-01-12T11:57:36.198+0000",
    "rdmVersion": 1,
    "rdmDeleteFlag": 0,
    "rdmExtensionType": "RelationTest",
    "tenant": {
        "id": "-1",
        "creator": "xdmAdmin",
        "modifier": "xdmAdmin",
        "createTime": "2022-09-22T04:10:48.543+0000",
        "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "Tenant",
        "tenant": null,
        "className": "Tenant",
        "name": "basicTenant",
        "description": "默认租户",
        "kiaguid": null,
        "securityLevel": "internal",
        "code": "basicTenant",
        "disableFlag": false,
        "dataSource": "DefaultDataSource"
    },
    "className": "RelationTest",
    "source": {
        "id": "454580805678901111",
        "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
        "createTime": "2023-01-12T11:55:22.797+0000",
        "lastUpdateTime": "2023-01-12T11:55:22.797+0000",
        "rdmVersion": 1,
        "rdmDeleteFlag": 0,
        "rdmExtensionType": "RelationLeft",
        "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "modifier": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "Tenant",
```

```
                "tenant": null,
                "className": "Tenant",
                "name": "basicTenant",
                "description": "默认租户",
                "kiaguid": null,
                "securityLevel": "internal",
                "code": "basicTenant",
                "disableFlag": false,
                "dataSource": "DefaultDataSource"
            },
            "className": "RelationLeft",
            "name": null,
            "description": null,
            "kiaguid": null,
            "securityLevel": "internal",
            "stuId": null,
            "right": {
                "id": "427473106174128128",
                "clazz": "RelationRight"
            },
            "stuName": null
        },
        "target": {
            "id": "454580805678904444",
            "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
            "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
            "createTime": "2023-01-12T11:55:47.918+0000",
            "lastUpdateTime": "2023-01-12T11:55:47.918+0000",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
            "rdmExtensionType": "RelationRight",
            "tenant": {
                "id": "-1",
                "creator": "xdmAdmin",
                "modifier": "xdmAdmin",
                "createTime": "2022-09-22T04:10:48.543+0000",
                "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
                "rdmVersion": 1,
                "rdmDeleteFlag": 0,
                "rdmExtensionType": "Tenant",
                "tenant": null,
                "className": "Tenant",
                "name": "basicTenant",
                "description": "默认租户",
                "kiaguid": null,
                "securityLevel": "internal",
                "code": "basicTenant",
                "disableFlag": false,
                "dataSource": "DefaultDataSource"
            },
            "className": "RelationRight",
            "name": null,
            "description": null,
            "kiaguid": null,
            "securityLevel": "internal",
            "stuId": null,
            "courseId": null,
            "relationLeftList": null
        },
        "name": null,
        "description": null
    }
],
"errors": [],
"pageInfo": {
    "curPage": 1,
    "pageSize": 20,
    "totalRows": 2,
    "totalPages": 1
```

```
    }
}
```

2.3.4 queryTarget

功能介绍

queryTarget接口可根据源数据模型ID和目标数据模型类型，查询目标实体的数据。

入参

```
{
    "params":{
        "sourceId": XXXX,
        "targetType":"XXXX",
        "latestOnly":false
    }
}
```

- sourceId：源/目标数据模型的ID。
- targetType：目标数据模型的类型。
- latestOnly：目标对象是否仅返回源对象关联的最新版本目标对象，默认为false。（仅对M-V模型有效，即返回所有版本）

出参

返回符合条件的所有目标实体的实例。

示例场景

假设有一个源端为RelationLeft，目标端为RelationRight的关系实体（RelationTest）。其中，

- RelationLeft有两个关系实例，唯一编码为454580805678901111和454580805678902222。
- RelationRight有两个关系实例，唯一编码为454580805678903333和454580805678904444。

并创建了如下唯一编码的关系实例：

- 1313：源端为454580805678901111，目标端为454580805678903333。
- 2424：源端为454580805678902222，目标端为454580805678904444。
- 1414：源端为454580805678901111，目标端为454580805678904444。

入参示例

根据源端RelationRight的ID为454580805678902222的关系实例，查询目标端类型为RelationRight的所有目标实体实例。

```
{
    "params":{
        "sourceId": 454580805678902222,
        "targetType":"RelationRight",
        "latestOnly":false
    }
}
```

出参示例


```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": "454580805678904444",
         "creator": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
         "createTime": "2023-01-12T11:55:47.918+0000",
         "modifier": "xdm_pub_validation f9828b63ee074aa69a2b3fb30aXXXXXX",
         "lastUpdateTime": "2023-01-12T11:55:47.918+0000",
         "rdmExtensionType": "RelationRight",
         "tenant": {
            "id": "-1",
            "creator": "xdmAdmin",
            "createTime": "2022-09-22T04:10:48.543+0000",
            "modifier": "xdmAdmin",
            "lastUpdateTime": "2022-09-22T04:10:48.543+0000",
            "rdmExtensionType": "Tenant",
            "tenant": null,
            "className": "Tenant",
            "name": "basicTenant",
            "description": "默认租户",
            "code": "basicTenant",
            "disableFlag": false,
            "dataSource": "DefaultDataSource"
         },
         "className": "RelationRight",
         "name": null,
         "description": null
      }
   ],
   "errors": [],
   "pageInfo": {
      "curPage": 1,
      "pageSize": 20,
      "totalRows": 1,
      "totalPages": 1
   }
}
```


2.3.5 deleteTarget

功能介绍

deleteTarget接口可根据源数据模型ID和目标数据模型的名称，删除关系实体的实例。

入参

```
{
    "params":{
          "sourceId": XXXX,
          "targetType":"XXXX",
          "latestOnly":false
    }
}
```

- modifier：处理人
- sourceId：源/目标数据模型的ID。
- targetType：目标类型。
- latestOnly：目标对象是否仅返回源对象关联的最新版本目标对象，默认为false。（仅对M-V模型有效，即返回所有版本）

出参

返回删除成功的数量。

示例场景

假设有一个源端为RelationLeft，目标端为RelationRight的关系实体（RelationTest）。其中，

- RelationLeft有两个关系实例，唯一编码为454580805678901111和454580805678902222。
- RelationRight有两个关系实例，唯一编码为454580805678903333和454580805678904444。

并创建了如下唯一编码的关系实例：

- 1313：源端为454580805678901111，目标端为454580805678903333。
- 2424：源端为454580805678902222，目标端为454580805678904444。
- 1414：源端为454580805678901111，目标端为454580805678904444。

入参示例

根据源端RelationRight的ID为454580805678901111的关系实例，删除所有符合条件的关系实例。

```
{
    "params":{
        "sourceId":454580805678901111,
        "targetType":"RelationRight",
        "latestOnly":false
    }
}
```

出参示例

```
{
  "result": "SUCCESS",
  "data": [
    2
  ],
  "errors": []
}
```

2.4 运算符相关示例

嵌套 and、or 和 in

```
{
  "params": {
    "sort": "desc",
    "orderBy": "name",
    "filter": {
      "joiner": "and",
      "conditions": [
        {
          "conditionName": "name",
          "operator": "=",
          "conditionValues": [
            "wyc"
```

```
            ]
         },
         {
            "joiner": "or",
            "conditions": [
               {
                  "conditionName": "creator",
                  "operator": "=",
                  "conditionValues": [
                     "wyc"
                  ]
               },
               {
                  "conditionName": "modifier",
                  "operator": "CONTAINS",
                  "conditionValues": [
                     "wyc",
                     "wxf"
                  ]
               }
            ]
         }
      ]
   }
}
```

如上示例可得出如下SQL查询语句：

```
(name=wyc)  and  (  (creator=wyc) or (modifier in (wyc, wxf) ) ) order by name desc
```

其中，“jioner”用于控制相同层次的“conditions”间的关系。

单个条件

- 示例1：

```
{
   "params": {
      "sort": "desc",
      "orderBy": "name",
      "filter": {
         "conditionName": "name",
         "operator": "=",
         "conditionValues": [
            "wyc"
         ]
      }
   }
}
```

- 示例2：

```
{
   "params": {
      "sort": "desc",
      "orderBy": "name",
      "filter": {
         "joiner": "and", // and或者or
         "conditions": [
            {
               "conditionName": "name",
               "operator": "=",
               "conditionValues": [
                  "wyc"
               ]
            }
         ]
      }
   }
}
```

模型自身属性

支持使用ConditionName和orderBy字段。如果输入的入参不是该模型的属性，会提示错误。其错误示例如下所示：

```
{
   "result": "FAIL",
   "data": [],
   "errors": [
      {
         "code": "rdm.coresdk.unknown.exception",
         "message": "unknown exception",
         "detailMessage": "org.hibernate.QueryException: could not resolve property: string of:
com.example.it.rdm.bean.module.DM0829002 [from com.example.it.rdm.bean.module.DM0829002 p left
join fetch p.master  where p.rdmDeleteFlag = 0  and UPPER(p.string) = UPPER('string') and p.latest = '1'
order by p.id desc]"
      }
   ]
}
```

其中，“could not resolve property: string”为具体错误的属性名称。

ISNULL 和 NOTNULL 传参

```
{
   "params": {
      "sort": "desc",
      "orderBy": "name",
      "filter": {
         "joiner": "and", // 或者or
         "conditions": [
            {
               "conditionName": "name",
               "operator": "ISNULL", // 或者NOTNULL
               "conditionValues": [  // conditionValues可不写
                  "wyc"
               ]
            }
         ]
      }
   }
}
```

参考对象

假设TestQuery的参考对象是SimpleEntity（别名为ref），SimpleEntity的别名为ref，有Long类型的ID字段和String类型的Name字段。用户希望对SimpleEntity过滤ID，其示例代码如下所示：

```
{
   "params": {
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "ref.id",
               "operator": "=",
               "conditionValues": [
                  "406889137980243968"
               ]
            }
         ]
      }
   }
}
```

如需过滤其他属性，基于新增过滤条件即可。

扩展属性

扩展属性的固定前缀为“extAttrs”，格式有“extAttrs.扩展属性名”和“extAttrs.扩展属性名.value”两种。其中，“extAttrs.扩展属性名.value”适用于自定义精度的扩展属性。

```
{
   "params": {
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "extAttrs.extAttrName1",
               "operator": "=",
               "conditionValues": [
                  "406889137980243968"
               ]
            },
            {
               "conditionName": "extAttrs.extAttrName2.value",
               "operator": "=",
               "conditionValues": [
                  "12.133"
               ]
            }
         ]
      }
   }
}
```

分类属性

分类属性的固定前缀为“clsAttrs”，格式有“clsAttrs.分类名.分类内属性名”和“clsAttrs.分类名.分类内属性名.value”两种。其中，“clsAttrs.分类名.分类内属性名.value”适用于自定义精度的分类属性。

```
{
   "params": {
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "clsAttrs.clsAttrName.attrName1",
               "operator": "=",
               "conditionValues": [
                  "123"
               ]
            },
            {
               "conditionName": "clsAttrs.clsAttrName.attrName2.value",
               "operator": "=",
               "conditionValues": [
                  "12.123"
               ]
            }
         ]
      }
   }
}
```

扩展类型+扩展属性/分类属性

扩展类型+扩展属性/分类属性的固定前缀为“clsAttrs”，格式有“clsAttrs.分类名.分类内属性名”和“clsAttrs.分类名.分类内属性名.value”两种。其中，“clsAttrs.分类名.分类内属性名.value”适用于自定义精度的扩展类型+扩展属性/分类属性。如需查

询此类属性，需额外增加一个过滤条件（rdmExtensionType），用于指明具体的扩展类型。

以“services/dynamic/api/Test0928001/find/20/1”的URI为例：

```
{
   "params": {
      "sort": "DESC",
      "orderBy": "lastUpdateTime",
      "filter": {
         "joiner": "and",
         "conditions": [
            {
               "conditionName": "clsAttrs.A0001.Testz",
               "operator": "like",
               "conditionValues": [
                  "WWW"
               ]
            },
            {
               "conditionName": "rdmExtensionType",
               "operator": "=",
               "conditionValues": [
                  "Test0928001"
               ]
            }
          ]
      },
      "isNeedTotal": false
   }
}
```

多字段排序

多字段排序一般使用“sorts”字段表示，其结构如下：

```
{
   "sorts": [
         {
            "sort": "DESC",
            "orderBy": "description"
         },
         {
            "sort": "ASC",
            "orderBy": "name"
         }
     ]
}
```

使用“sorts”字段后，将按照其列表顺序进行排序。如上所示，对象会先按照“description”进行倒序排序，再按照“name”进行正序排序。

以“services/api/SZAPITEST202211280329/find/20/1”的URI为例，使用如下接口进行多字段排序。

- Find接口：

  ```
  {
     "params": {
        "sorts": [
           {
              "sort": "DESC",
              "orderBy": "description"
           },
           {
              "sort": "ASC",
              "orderBy": "name"
           }
  ```

```
    ],
    "filter": {
      "joiner": "and",
      "conditions": [
        {
          "conditionName": "description",
          "operator": "=",
          "conditionValues": [
            "a"
          ]
        },
        {
          "conditionName": "rdmDeleteFlag",
          "operator": "=",
          "conditionValues": [
            "0"
          ]
        }
      ]
    }
  }
}
```

- Query接口：

```
{
  "params": {
    "sorts": [
      {
        "sort": "DESC",
        "orderBy": "description"
      },
      {
        "sort": "ASC",
        "orderBy": "name"
      }
    ],
    "filter": {
      "joiner": "and",
      "conditions": [
        {
          "conditionName": "description",
          "operator": "=",
          "conditionValues": [
            "a"
          ]
        },
        {
          "conditionName": "rdmDeleteFlag",
          "operator": "=",
          "conditionValues": [
            "0"
          ]
        }
      ]
    }
  }
}
```

- Select接口：

```
{
  "params": {
    "selectedField": [
      {
        "name": "name",
        "nameAs": "nameAsName"
      },
      {
        "name": "description",
        "nameAs": "nameAsDescription"
      },
```

```
        {
            "name": "creator"
        },
        {
            "name": "id"
        }
    ],
    "sorts": [
        {
            "sort": "DESC",
            "orderBy": "name"
        },
        {
            "sort": "asc",
            "orderBy": "id"
        }
    ],
    "filter": {
        "joiner": "and",
        "conditions": [
            {
                "conditionName": "description",
                "operator": "=",
                "conditionValues": [
                    "a"
                ]
            }
        ]
    }
  }
}
```

3 使用搜索服务定义搜索数据

搜索服务定义是一个可自定义的将部分模型项或者关系实体的实例数据进行全文检索的搜索服务，属于服务编排的一种类型。相较于全量数据的全文检索，搜索服务定义是一种对某种场景下内关联模型实例数据的一种“小场景”搜索。当用户的某些业务场景数据量巨大，且对搜索的性能和匹配灵活性要求较高时，需要用户自定义搜索服务的配置和搜索的数据范围。搜索服务定义可以更好的明确用户搜索的场景和意图，也为特定场景下大批量的数据搜索和过滤提高搜索效率，避免在大批量数据中召回干扰信息。

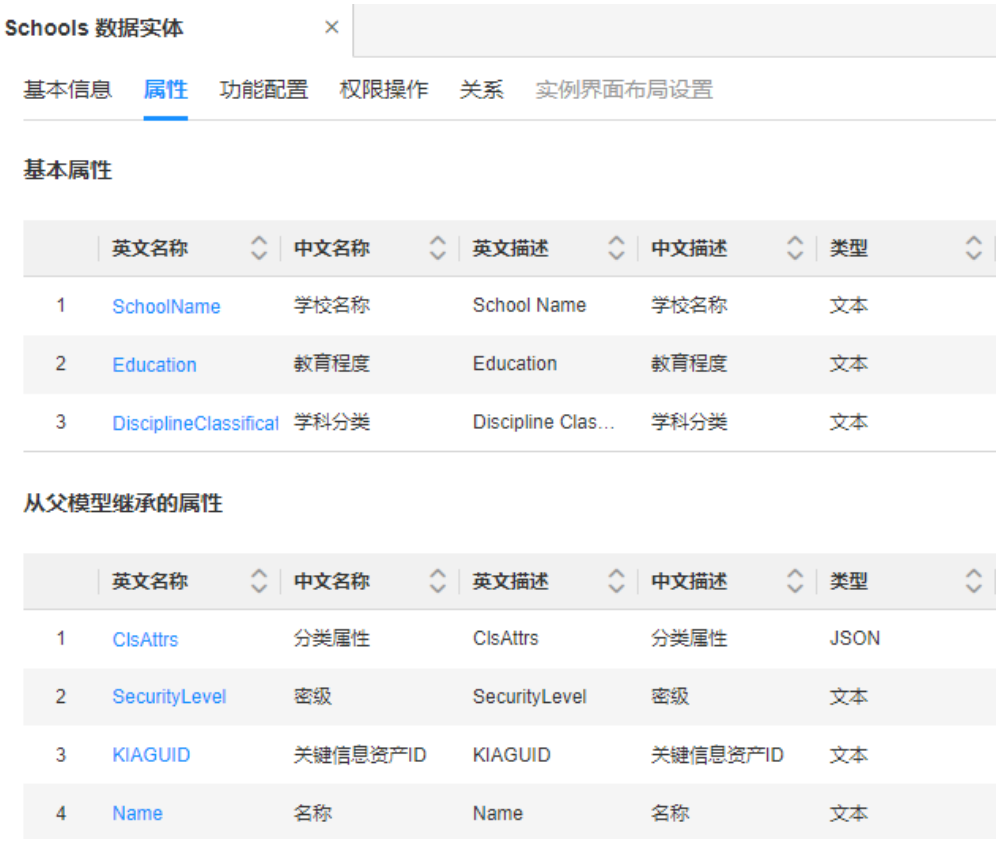
以如下示例场景为例，通过**横向搜索**和**纵向搜索**两个维度指导您如何使用搜索服务定义。

示例场景

假设有如下两个数据实体：

- Schools：表示学校，用于存储学校信息（如学校名称、教育程度、学科分类等）。

图 3-1 Schools 数据实体



Schools 数据实体

基本信息 属性 功能配置 权限操作 关系 实例界面布局设置

基本属性

	英文名称	中文名称	英文描述	中文描述	类型
1	SchoolName	学校名称	School Name	学校名称	文本
2	Education	教育程度	Education	教育程度	文本
3	DisciplineClassifical	学科分类	Discipline Clas...	学科分类	文本

从父模型继承的属性

	英文名称	中文名称	英文描述	中文描述	类型
1	ClsAttrs	分类属性	ClsAttrs	分类属性	JSON
2	SecurityLevel	密级	SecurityLevel	密级	文本
3	KIAGUID	关键信息资产ID	KIAGUID	关键信息资产ID	文本
4	Name	名称	Name	名称	文本

- Students：表示学生，用于存储学生信息（如姓名、性别、年龄、班级、学校等）。其中，学校为“参考对象”类型，参考Schools数据实体。

图 3-2 Students 数据实体


Students 数据实体

基本信息 属性 功能配置 权限操作 关系 实例界面布局设置

基本属性

	英文名称	中文名称	英文描述	中文描述	类型
1	Age	年龄	Age	年龄	整型
2	School	学校	School	学校	参考对象
3	ClassAndGrade	班级	Class and grade	班级	文本
4	Gender	性别	Gender	性别	文本

从父模型继承的属性

	英文名称	中文名称	英文描述	中文描述	类型
1	ClsAttrs	分类属性	ClsAttrs	分类属性	JSON
2	SecurityLevel	密级	SecurityLevel	密级	文本
3	KIAGUID	关键信息资产ID	KIAGUID	关键信息资产ID	文本
4	Name	名称	Name	名称	文本

并通过这两个数据实体分别创建了如表3-1所示的数据实例。

表 3-1 数据实例

数据实体	数据实例	实例信息
Schools	清华大学	学校名称：清华大学
	北京大学	学校名称：北京大学
	深圳大学	学校名称：深圳大学
Students	王小华	• 性别：女 • 年龄：19 • 学校：深圳大学
	吴清	• 性别：女 • 年龄：20 • 学校：北京大学
	李四	• 性别：男 • 年龄：21 • 学校：清华大学

数据实体	数据实例	实例信息
	张三	• 性别：男 • 年龄：20 • 学校：深圳大学

- 关于数据实体的创建和属性的添加，请参见**创建数据实体**和**管理数据实体属性**。
- 关于数据实例的创建，请参见**创建数据实例**。

横向搜索

横向搜索，即在同行业中进行搜索。以查询北京大学的女学生为例，通过搜索服务定义实现该查询，具体操作步骤如下：

步骤1 登录应用运行态。

步骤2 在左侧导航栏中，选择“搜索服务管理 > 搜索服务定义”，单击“创建”。

步骤3 在展开的“服务定义”页面，设置如下主要信息，单击“保存”。

表 3-2 参数信息

参数	说明
API英文名称	输入“Schoolgirl”。
API中文名称	输入“女学生”。
API英文描述	输入“School girl”。
API中文描述	输入“女学生”。

步骤4 选择“索引定义”页签，添加“学生姓名”、“性别”和“学校”三个索引。其主要信息如**表3-3**所示。

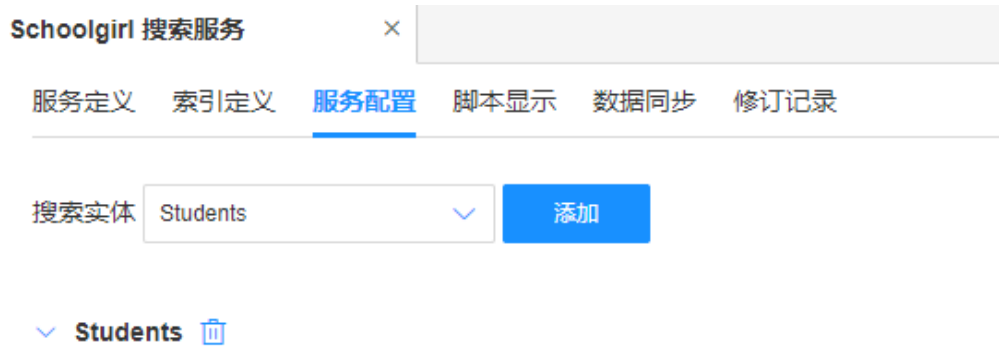
表 3-3 参数信息

索引名称	学生姓名	性别	学校
索引类型	选择“文本”。	选择“文本”。	选择“文本”。
分词方法	选择“普通分词”。	选择“不分词”。	选择“不分词”。
分词选项	选择“不涉及”。	选择“不涉及”。	选择“不涉及”。
作为过滤条件	选择“N”。	选择“Y”。	选择“Y”。
参与关键词搜索	选择“Y”。	选择“Y”。	选择“Y”。
展示	选择“Y”。	选择“Y”。	选择“Y”。

匹配方法	选择“精确匹配”。	选择“精确匹配”。	选择“精确匹配”。

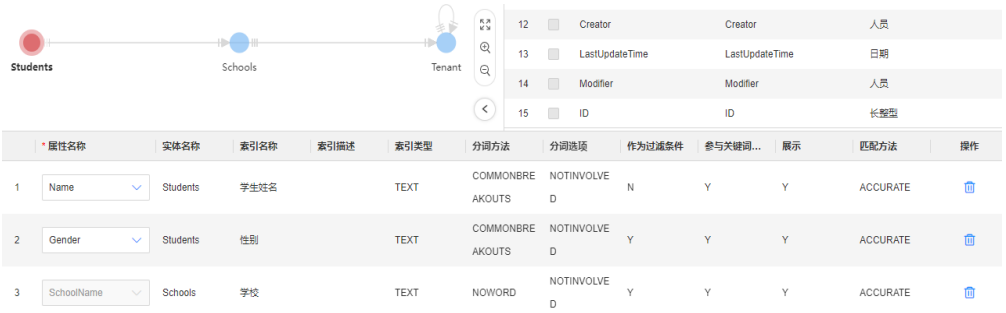
步骤5 选择“服务配置”页签，选择搜索实体（Students），单击“添加”。

图 3-3 服务配置



步骤6 在数据图谱的下方，分别设置如图3-4所示的实体属性，单击“保存”。

图 3-4 选择属性名称

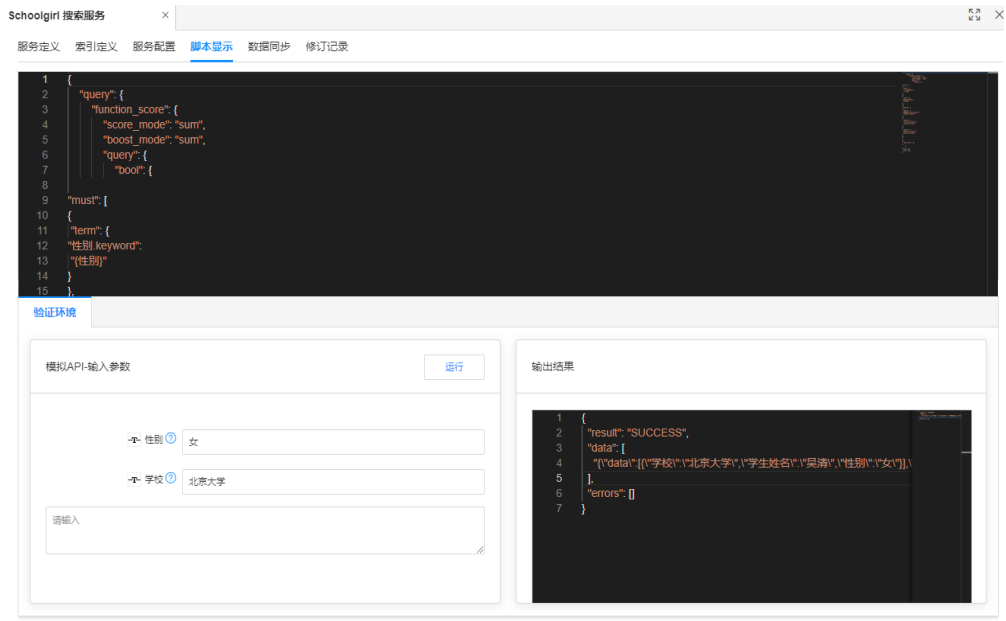


步骤7 单击“发布”。

步骤8 选择“脚本显示”页签，在“验证环境 > 输入参数”的“性别”输入“女”，“学校”输入“北京大学”，单击“运行”。

可在右侧的“输出结果”发现搜索服务定义将北京大学的女学生均搜索出来。

图 3-5 脚本显示



步骤9 选择“数据同步”页签，单击“同步数据”。

步骤10 在弹出的提示框中，单击“确认”。

----结束

纵向搜索

纵向搜索，即在不同行业，相同职位中搜索。以查找名字中带有“华”字的信息为例，通过搜索服务定义实现该查询，具体操作步骤如下：

步骤1 登录运行态。

步骤2 在左侧导航栏中，选择“搜索服务管理 > 搜索服务定义”，单击“创建”。

步骤3 在展开的“服务定义”页面，设置如下主要信息，单击“保存”。

表 3-4 参数信息

参数	说明
API英文名称	输入“Name”。
API中文名称	输入“名字”。
API英文描述	输入“Name”。
API中文描述	输入“名字”。

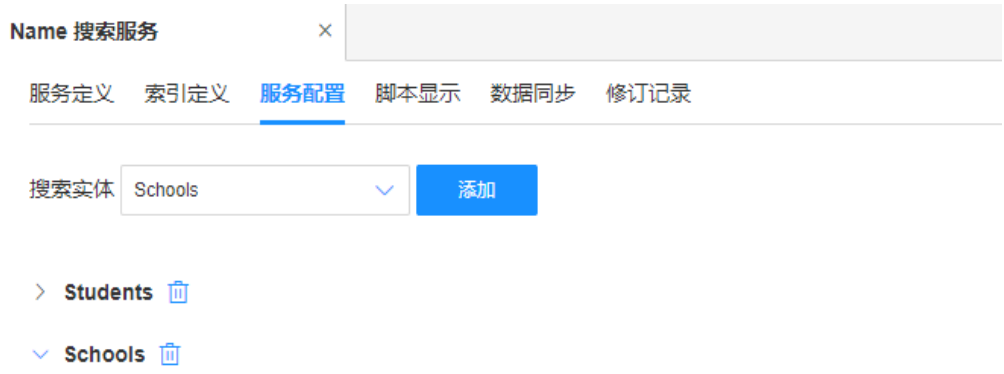
步骤4 选择“索引定义”页签，单击“添加索引”，添加一个“名字”索引。其主要信息如表3-5所示。

表 3-5 参数信息

索引名称	输入“名字”。
索引类型	选择“文本”。
分词方法	选择“单字分词”。
分词选项	选择“全拼”。
作为过滤条件	选择“Y”。
参与关键词搜索	选择“Y”。
展示	选择“Y”。
匹配方法	选择“模糊匹配”。

步骤5 选择“服务配置”页签，添加两个搜索实体（Schools和Students）。

图 3-6 服务配置



步骤6 分别在Schools搜索实体和Students搜索实体的数据图谱下方，将“属性名称”选择为“Name”。

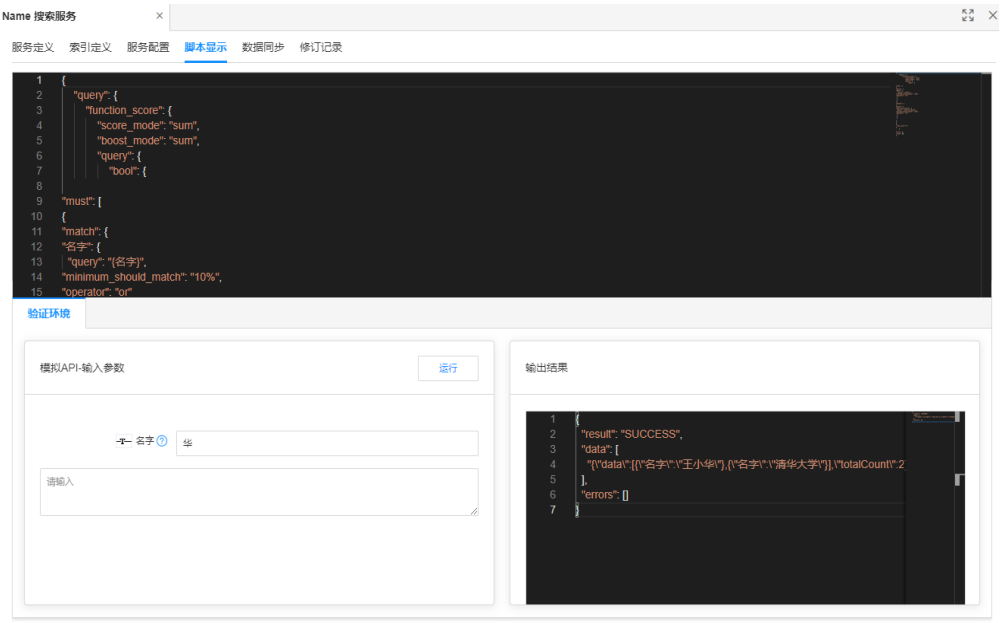
图 3-7 选择属性名称

步骤7 单击“发布”。

步骤8 选择“脚本显示”页签，在“验证环境 > 输入参数”的“名字”输入“华”，单击“运行”。

可在右侧的“输出结果”发现搜索服务将“Name”中带有“华”字的数据均搜索出来。

图 3-8 脚本显示



步骤9 选择“数据同步”页签，单击“同步数据”。

步骤10 在弹出的提示框中，单击“确认”。

----结束

4 使用高代码服务编排自定义 API

工业数字模型驱动引擎-数据建模引擎（xDM Foundation，简称xDM-F）不仅为用户提供了丰富的标准API（如数据实体API、关系实体API、系统管理API），还提供了高代码的服务编排能力，适配Java和JavaScript类型服务编排。当iDME提供的标准API无法满足用户的业务需求时，用户可使用服务编排能力编排原子API形成一个跨实体（表）的组合API，提高应用开发的速度与质量。

说明

当前服务编排管理功能仅支持在广州友好环境使用。

以如下示例场景及JavaScript类型服务编排为例，指导您如何使用高代码服务编排自定义API。

示例场景

假设在应用运行态有如下数据实体和关系实体，并基于这些实体创建了若干实例。希望自定义一个API，该API可根据票价和余票的具体值，查询票价≤票价具体值，余票≥余票具体值的电影院和影片信息，分页显示。例如，查询票价≤19.9，余票≥8的电影院ID、影片ID、票价和余票信息。

表 4-1 实体

实体类型	实体名称	说明
数据实体	英文名称：Cinema 中文名称：电影院	用于存储电影院信息，如部门、收入等。
	英文名称：Film 中文名称：影片	用于存储影片信息，如电影导演、主演、简介、语言、片长、类型等。
关系实体	英文名称：Play 中文名称：上映	为Cinema（源数据实体）和Film（目标数据实体）建立多对多关系。 关系实体属性包含票价（price）和余票（remain），均为数值型属性。

步骤 1：创建高代码编排

步骤1 登录应用运行态。

步骤2 在左侧导航栏中，选择“服务编排管理 > 高代码编排”，单击“创建”。

步骤3 在展开的“服务定义”页面，设置如下主要信息，其他保持默认，单击“保存”。

图 4-1 创建服务编排

表 4-2 参数信息

参数	说明
API英文名称	输入“QueryByPriceRemain”。
API中文名称	输入“按票价和余额查询”。
API英文描述	输入“Query by price and remain”。
API中文描述	输入“按票价和余额查询”。
API类型	选择“JavaScript”。

步骤4 选择“脚本显示”页签，根据系统提供的参考样例，输入如下JavaScript编排脚本，单击“保存”。

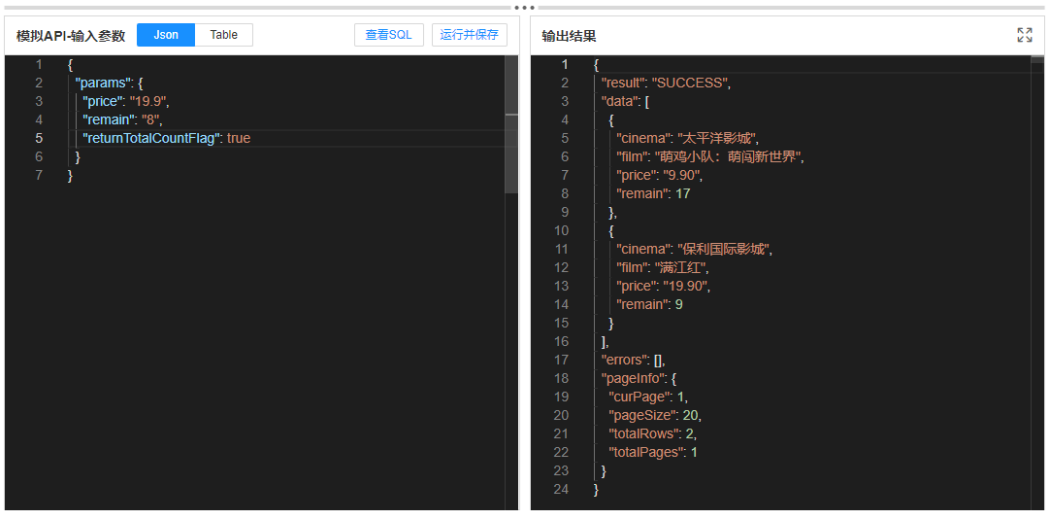
```
var request = "";
var sqlCount  = "select count(*) as count from TestDME_Play_REL p ";
var sql = "select c.name as cinema, f.name as film, p.price, p.remain from TestDME_Play_REL p ";
sql += "left join TestDME_Cinema c on c.id = p._sourceid ";
sql += "left join TestDME_Film f on f.id = p._targetid ";

if (request.price != null && request.remain!=null)
{
   sql += " where price <= {#price} and remain>={#remain} ";
   sqlCount += " where price <= {#price} and remain>= {#remain} ";
}
```

- TestDME_Play_REL/TestDME_Cinema/TestDME_Film：表示Play关系实体/Cinema数据实体/Film数据实体在数据库中的表名称。
- id/name/_sourceid/_targetid/price/remain：表示实体表中的属性名称，即唯一编号（ID属性），名称，源数据实体（Cinema）的唯一编号（ID属性）、目标数据实体（Film）的唯一编号（ID属性）、票价和余票。
- {#price}和{#remain}：数值型属性（票价和余票）的写法。如为文本类型属性，应修改为'{#price}'和'{#remain}'。

步骤5 在“模拟API-输入参数”栏中，输入price和remain的参数值，单击“保存并执行”，验证JavaScript编排脚本。

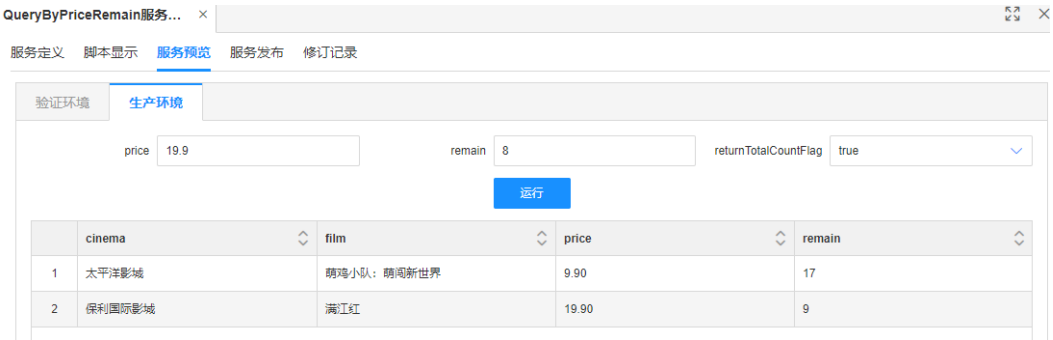
图 4-2 模拟 API



returnTotalCountFlag参数是系统自动添加，如果returnTotalCountFlag为true，输出结果将返回数据实例总数。

步骤6 选择“服务预览”页签，输入price和remain的参数值，单击“运行”，预览生产环境的运行效果。

图 4-3 服务预览



步骤7 选择“服务发布”页签，确认该服务编排的基本信息、请求参数说明、响应参数说明、请求示例、正常响应示例等信息无误后，单击“发布”。

----结束

步骤 2：验证服务编排

步骤1 在应用运行态左侧导航栏中，选择“数据服务管理 > 全量数据服务”，进入全量数据服务页面。

步骤2 在“分类”栏中，展开服务编排，找到并选择刚创建的服务编排（QueryByPriceRemain）。

图 4-4 全量数据服务

步骤3 在API列表栏中，单击API英文名称，查看QueryByPriceRemain API的详细信息。

图 4-5 API 详情

----结束

步骤 3：使用服务编排 API

根据QueryByPriceRemain API的详细信息，在用户应用的管理系统中构造如下请求示例：

```
POST http://{Endpoint}/rdm_{appID}_app/services/rdm/basic/api/customservice/QueryByPriceRemain/execute/20/1
Content-Type: Application/Json
Content-Length: Content Length
X-AUTH-TOKEN: XXXXXXXXXX
{
   "params": {
      "price": 19.9,
      "remain": 8,
      "returnTotalCountFlag": true,
   }
}
```

其中，{Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID。

5 通过 xDM-F 的多租户能力实现应用运行态数据的逻辑隔离

操作场景

假设A公司在工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）开发了一个应用（PDM），希望将该应用授权给B公司和C公司使用，且两个公司之间的数据不互通。此时，用户可以通过iDME应用提供的多租户能力，创建多个租户，为B公司和C公司的数据进行逻辑隔离。

前提条件

已**登录应用运行态**。

步骤 1：创建租户

创建两个租户实例，租户编码分别为tenantB和tenantC。具体操作请参见**租户管理**。

图 5-1 租户实例

步骤 2：逻辑隔离租户数据

- 可视化页面：在应用运行态的右上方，单击租户的下拉列表，切换为tenantB或tenantC。即可在tenantB和tenantC租户下，分别管理B公司和C公司在PDM应用下的模型数据。例如，**创建扩展模型的数据实体**、**创建数据实例**、**创建扩展属性**等。

  图 5-2 切换租户

- API方式：在待调用的API中为B公司和C公司添加“tenantid”必填参数。即，当B公司或者C公司对PDM应用进行增删改查操作时，发送的请求头必须带有tenantid。

  请求头：

```
POST http://{Endpoint}/rdm_{appID}_app/services/dynamic/api/{entityName}/{method}
Content-Type: Application/Json
Content-Length: Content Length
X-AUTH-TOKEN: Token ID
tenantid: 租户编码

{
   "params": {
      "id": "唯一编码",
      ......
   }
}
```

其中，{Endpoint}表示数据建模引擎所在域名或IP地址，{appID}表示应用ID，{entityName}表示实体的英文名称，{method}表示请求方法。

以在tenantB租户下创建单位类型为例，其请求示例如下：

```
POST http://dme.cn-north-4.huaweicloud.com/rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/
services/dynamic/api/UnitType/create
Content-Type: Application/Json
Content-Length: Content Length
X-AUTH-TOKEN: XXXXXXXXXX
tenantid: tenantB

{
   "params": {
      "id": "439166695949471744",
      "nameEn": "UnitTest",
      "name": "UnitTest",
      "descriptionEn": "",
      "descriptionCn": "",
      "unitGbFlag": true
   }
}
```

步骤 3：隔离验证

如下操作以**步骤2：逻辑隔离租户数据**的创建单位类型为示例进行验证。

- 通过可视化页面方式验证。
  a. 在应用运行态的右上方，单击租户的下拉列表，选择“tenantB”。
  b. 在左侧导航栏中，选择“基础数据管理 > 计量单位”，进入“计量单位”页面。
     该页面下存在“UnitTest”的数据。
  c. 在应用运行态的右上方，单击租户的下拉列表，选择“tenantC”。
  d. 在左侧导航栏中，选择“基础数据管理 > 计量单位”，进入“计量单位”页面。
     该页面下无“UnitTest”的数据。
- 通过API方式调用单位类型的查询接口验证。
  – 发送请求头带有“tenantid”为“tenantB”的查询请求。

```
POST http://dme.cn-north-4.huaweicloud.com/
rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/UnitType/find/20/1
Content-Type: Application/Json
Content-Length: Content Length
X-AUTH-TOKEN: XXXXXXXXXX
tenantid: tenantB

{
   "params": {
      "sort": "DESC",
      "orderBy": "lastUpdateTime",
      "filter": {
```

```
            "joiner":"string",
            "conditions": []
        },
        "isNeedTotal": true
    }
}
```

可查询到在“tenantB”租户下创建的单位类型。

- 发送请求头带有“tenantid”为“tenantC”的查询请求。

```
POST http://dme.cn-north-4.huaweicloud.com/
rdm_01a2b2c4764d4e00f123g345fd9baa9f_app/services/dynamic/api/UnitType/find/20/1
Content-Type: Application/Json
Content-Length: Content Length
X-AUTH-TOKEN: XXXXXXXXXX
tenantid: tenantC

{
    "params": {
        "sort": "DESC",
        "orderBy": "lastUpdateTime",
        "filter": {
            "joiner":"string",
            "conditions": []
        },
        "isNeedTotal": true
    }
}
```

查询不到在“tenantB”租户下创建的单位类型。

6 通过反向建模将已有数据库物理表转为iDME 模型

操作场景

反向建模是从数据库物理表到数据模型的映射，相对于将数据模型映射到数据库物理表的正向建模。

当您的本地服务器中已有数据库和物理表，且希望通过工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）统一管理所有模型时，可使用反向建模功能，将已有物理表反向建模至iDME的数据模型中。此功能可帮助您无需再次执行创建模型的操作，即可快速创建数据模型，节省了大量时间成本。

前提条件

- 已**开通iDME设计服务**和**购买iDME运行服务**。本章节以购买体验版数据建模引擎为例。
- 已创建与本地服务器已有数据库类型一致的应用，具体操作请参见**创建应用**。

  即，如本地服务器的数据库类型为MySQL，则创建“数据库类型”为“MySQL”的应用。本章节以MySQL数据库类型为例。
- 已获取如下本地服务器的数据库信息，且该数据库可正常连通。

  表 6-1 MySQL 数据库信息

类型	描述
数据库名称	database_test
用户名	root
密码	123456
数据库地址	192.168.10.10:3306/database_test

- 已确定需要将哪些物理表反向生成模型。

限制和说明

- 反向建模仅支持将物理表反向创建为数据实体和关系实体。
- 物理表的字段在反向建模时自动解析为对应数据实体/关系实体的属性，且不支持编辑。如需编辑，可在建模后前往该数据实体/关系实体详情页进行编辑，具体操作请参见**管理数据实体属性**和**管理关系实体属性**。
- 待反向建模的数据库需允许公网访问。

更多限制和说明请参见**反向建模概述**。

操作步骤

步骤1 登录应用设计态，具体操作请参见**登录应用设计态**。

步骤2 在左侧导航栏中，选择“数据模型管理 > 反向建模”，单击“新增”，弹出“添加数据源”窗口。

步骤3 在弹出的窗口中，根据**前提条件**获取的数据库信息，设置如下信息，单击“确定”。

表 6-2 数据源-基本信息

参数	参数说明
名称	填写数据源名称，用户自定义。 示例：“database_test”。
数据库地址	填写数据库的地址。 示例：“jdbc:mariadb://192.168.10.10:3306/database_test”。
数据库名称	填写数据库的名称。 示例：“database_test”
用户名	填写连接数据库的用户名称。 示例：“root”。
密码	填写连接数据库的密码。
数据库类型	选择与当前应用相同的数据库类型。 示例：选择“MySQL”。
开启SSL	选择是否开启SSL加密。 示例：保持默认设置，选择“否”。

步骤4 选择刚创建的数据源，单击，弹出读取或更新数据的提示框。

步骤5 在弹出的提示框中，单击“确定”。

读取数据需要等待一段时间，您可以单击刷新数据源的状态，当“状态”为“读取成功”即表示成功将本地服务器的数据库物理表读取至应用设计态。

步骤6 单击，进入反向建模页面，确认模型信息。

iDME会根据读取的物理表数据自动生成对应的建模信息和物理表信息，您可以单击“物理表名称”进入物理表详情页进行查看。其中“建模信息”的默认设置如表6-3所示。

表 6-3 物理表的默认建模信息

配置项	默认值
实体类型	数据实体。
沿用表名称	是。
模型表名称	物理表的名称。
模型英文名称	物理表的名称。
模型中文名称	物理表的名称。
模型英文描述	-
模型中文描述	物理表的名称。
模型父模型	BasicObject。
模型分类	业务数据模型。
模型责任人	-

步骤7 生成的建模信息可能会存在偏差，请根据业务需求和前提条件中已确定的建模范围，修改需要创建为数据实体的物理表的建模信息。

表 6-4 修改建模信息

操作类型	操作步骤
批量修改建模信息	1. 勾选需要修改的物理表，单击“批量修改”，展开“批量修改模型信息”页面。 2. 在展开的页面，单击选“择修改字段”下拉框，勾选需要修改的字段。 仅支持批量修改“沿用表名称”、“模型分类”和“模型责任人”。 3. 根据业务需求修改相应配置信息，单击“确定”。 具体操作请参见批量修改建模信息。
修改单个建模信息	1. 找到需要修改的物理表，单击，展开物理表的详情页面。 2. 根据业务需求修改相应配置信息，单击“保存”。 具体操作请参见修改建模信息。

步骤8 勾选待建模的物理表，单击“批量建模”。

完成建模后，您可单击刷新建模状态，您可在反向建模列表查看本次成功创建的数据实体，以及建模失败的详细信息。

表 6-5 建模状态

状态	说明
建模成功	“状态”显示为“已建模”，“模型编码”显示对应数据实体的编码，“操作”显示。 单击“模型编码”或者，可前往物理表对应创建的数据实体详情页，查看或者编辑该数据实体。 具体操作请参见**数据实体**。
建模失败	“状态”仍显示为“未建模”，且带有标识。 将鼠标移动至，显示建模失败的详细信息。

步骤9 重复执行**步骤7**，修改需要创建为关系实体的物理表的建模信息。

步骤10 重复执行**步骤8**，完成反向创建关系实体。

完成建模后，您可单击刷新建模状态，您可在反向建模列表查看本次成功创建的关系实体，以及建模失败的详细信息。

表 6-6 建模状态

状态	说明
建模成功	“状态”显示为“已建模”，“模型编码”显示对应关系实体的编码，“操作”显示。 单击“模型编码”或者，可前往物理表对应创建的关系实体详情页，查看或者编辑该关系实体。 具体操作请参见**关系实体**。
建模失败	“状态”仍显示为“未建模”，且带有标识。 将鼠标移动至，显示建模失败的详细信息。

----结束

下一步操作

反向建模完成后，您可以在iDME执行如下操作。

您可以...	进行...
进入“枚举类型”页面	为数据模型预设枚举值，具体操作请参见**创建枚举**。
进入“接口模型”页面	• 自定义接口模型，对数据实体或关系实体的公共特性进行抽象&接口化，具体操作请参见**创建接口模型**。 • 发布自定义的接口模型，具体操作请参见**发布接口模型**。

您可以...	进行...
进入“数据实体”页面	• 查看创建的数据实体，具体操作请参见**查看数据实体**。 • 数据实体属性的添加、修改、删除和上下位置调整，具体操作请参见**管理数据实体属性**。 • 为创建的数据实体添加其他iDME内置的工业数据能力，具体操作请参见**管理功能配置**。 • 发布创建的数据实体，具体操作请参见**发布数据实体**。
进入“关系实体”页面	• 查看创建的关系实体，具体操作请参见**查看关系实体**。 • 关系实体属性的添加、修改、删除和上下位置调整，具体操作请参见**管理关系实体属性**。 • 为创建的关系实体添加其他iDME内置的工业数据能力，具体操作请参见**管理功能配置**。 • 发布创建的关系实体，具体操作请参见**发布关系实体**。
单击右上方的“应用发布”	应用的发布，生成相应代码包，具体操作请参见**发布应用**。
完成发布应用后，返回控制台	应用部署至数据建模引擎，具体操作请参见**部署应用**。
进入应用运行态	相关全量数据服务API的开发操作，具体操作请参见**数据建模引擎使用指南**。

7 文件服务实践

7.1 文件服务概述

业务痛点

近几年来，我国工业的数字化进程加速，从研发、生产、销售、物流等诸多环境都会产生大量的文件，多、杂且分散，给企业管理和协作都带来不少难点。

- **文件多，分散存储，查阅不方便，管理效率低**

  在工业相关企业的经营过程中，会涉及到很多合作单位或者部门，企业员工需要经常查阅各合作单位/部门的相关文件，例如图纸、施工记录、验收报告等纸质文件，各种电子文件、邮件等。传统文件管理方式存在着文件分散、没有集中管理等情况，不仅对文件的查阅带来极大的阻碍，还会耗费大量的人力和时间，效率低下，容易出现错误、遗漏等问题。

- **文件管理安全性不高**

  工业相关企业的文件中往往包含大量的敏感信息，例如设计图纸、客户订单、物料标准、财务报告等，具有很高的安全性和保密性要求，一旦泄露或者丢失会对企业的运营和发展造成很大影响。传统的文件管理方式，企业员工可以随时篡改、盗取或者销毁，极易导致重要数据资产丢失或泄露，给企业带来损失。

- **传输大文件时，传输速度慢、不稳定、不灵活、不兼容等**

  在各个领域中，有很多需要传输大文件的情况，例如国际数据传输、数据协作、数据备份等。这些情况涉及到的大文件包括图纸设计、图像、文档等。使用传统的FTP/HTTP等传输方式时，容易受到网络波动和网络延迟的制约或者其他因素的影响，从而导致传输中断或失败，需要重新开始或者手动恢复，增加工作负担和成本。

解决方案

工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）提供的文件服务功能可大大提升文件管理的效率。

- 通过将文件和数据模型的实例数据建立关联，以对象化方式管理文件，方便文件归类、查找和更新。
- 通过对文件类型属性的“密级”和“加密”设置，实现文件的加密或解密存储，保障企业知识资产。此外，还支持通过API方式随时随地预览文件（图片）。

- 提供BLOB和对象存储两种存储方式，支持简单上传、分块上传、断点续传上传和闪传。

  当文件大于100M或网络环境较差时，使用分块上传可实现并行上传多个分块以加快上传速度。如果分块上传过程中某一分块上传失败，再次上传时会从系统记录的点继续上传，从而达到断点续传的效果。如果系统已存在某个文件，后续再上传该文件时，系统会根据文件的唯一哈希值直接复制文件，无需重新上传文件，即可快速完成上传任务。

操作流程

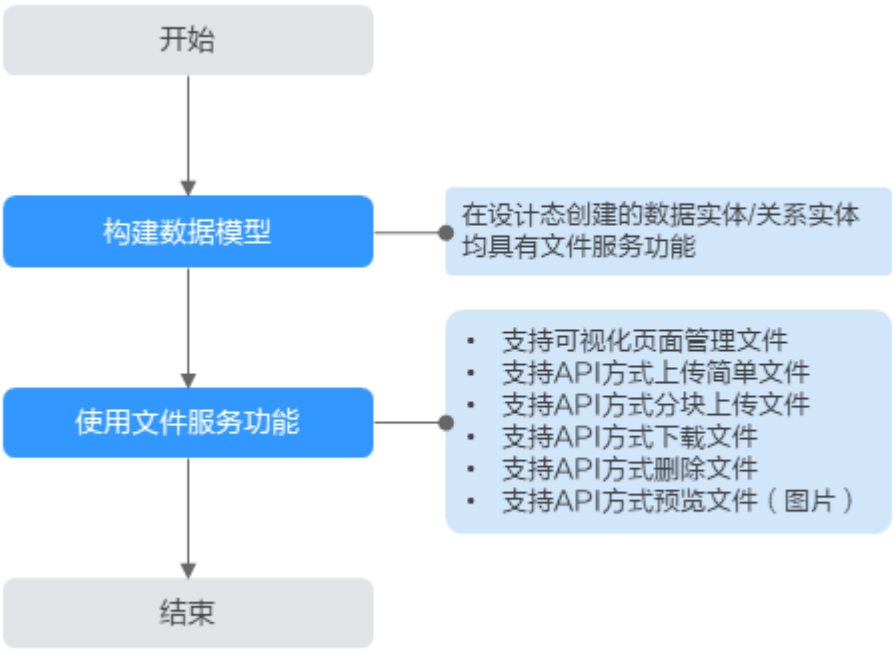


表 7-1 基线管理操作流程

主要操作流程	操作目的
构建数据模型	• 使用iDME的数据模型管理完成对业务数据对象的模型设计，为构建的数据模型添加文件类型属性。 • 发布在应用设计态构建的数据模型，并将应用部署至数据建模引擎。
使用文件服务功能	在数据模型实例化的过程中，使用文件服务功能管理文件，包括上传文件、下载文件、预览文件（图片）等。 • 可视化页面：在应用运行态的数据实例中管理文件。 • API方式：使用iDME的全量数据服务API管理文件。

使用说明

- 接口模型不支持文件服务功能。

- iDME预置多种可上传的文件格式，如果预置的文件格式不能满足您的业务需求，可前往**文件类型白名单**进行自定义配置。

功能差异

下表展示了iDME文件服务功能支持的主要能力，以及可视化页面与API方式支持的功能差异。

说明

“×”代表不支持；“√”代表支持。

功能	可视化页面	API方式
简单文件上传	√	√
分块上传	√	√
断点续传	√	√
外链上传	√	√
下载文件	√	√
删除外链	√	√
预览文件（图片）	×	√
更新文件	×	√

7.2 构建数据模型

操作场景

文件服务是数据实体/关系实体的基础功能之一，在应用设计态创建的数据实体/关系实体均具有该能力。但在使用文件服务功能之前，您需要为数据实体/关系实体添加一个“文件”类型的属性。

- 如果您希望构建的数据模型在实例化时可以直接使用文件服务功能，您可在应用设计态构建数据模型时添加“文件”类型的基础属性/自定义基础属性，具体操作请参见**管理数据实体属性**/**管理关系实体属性**。
- 如果您希望构建的数据模型可以面向多种业务场景，或者不想对已构建的数据模型进行变更，但当前构建的数据模型无法满足您的业务需求时，您可以先在应用设计态构建具有“扩展属性”功能的数据模型，并在数据模型的“功能配置”中配置扩展属性规则（包含配置一个“文件”类型的扩展属性规则）。而后即可在应用运行态对数据模型进行定制化扩展。

本章节以构建一个“文件”类型扩展属性的数据实体为例。

操作步骤

步骤1 **登录应用设计态**。

步骤2 在左侧导航栏中，选择“数据模型管理 > 数据实体”，进入数据实体页面。

步骤3 单击“创建”，根据如下主要信息，创建一个具有“扩展属性”功能的数据实体（例如Industrial_File）。

图 7-1 数据实体信息

表 7-2 主要参数信息

类型	参数	参数说明
基本信息	英文名称	Industrial_File。
	中文名称	工业文件。
	中文描述	用于管理工业文件的上传和下载。
	模型类型	选择“实体模型”。
	父模型	保持默认，此处以BasicObject为例。
功能列表	可选功能	添加“扩展属性”功能。

步骤4 选择“功能配置”页签，根据业务需求，在“设置规则 > 扩展属性”栏设置各类扩展属性的规则，用于约束应用运行态可添加扩展属性的类型和数量。

此处以新增一个“属性类型”为“文件”的扩展属性为例。

图 7-2 设置扩展属性规则

步骤5 依次完成“发布数据实体 > 发布应用 > 部署应用”的操作，具体操作请参见**发布数据实体**、**发布应用**和**部署应用**。

步骤6 待应用完成部署后，**登录应用运行态**。

步骤7 在“数据模型管理 > 属性库”中，创建一个“文件”类型的属性，具体操作请参见**创建属性**。

例如，创建一个文件类型属性“LargeFiles”，“存储方式”选择为“对象存储”，其他参数保持默认。

步骤8 在“数据模型管理 > 数据实体”中，找到“Industrial_File”数据实体，单击。

步骤9 在“基本信息”页签，将“实例界面显示”设置为“是”，单击“保存”。

图 7-3 实例界面显示

步骤10 在“属性”页签的“扩展属性”栏，添加**步骤7**创建的文件类型属性（LargeFiles）。

图 7-4 添加扩展属性

----结束

下一步操作

使用文件服务功能

7.3 使用文件服务功能

7.3.1 通过可视化页面管理文件

操作场景

工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）支持可视化管理iDME应用中的文件。本文为您介绍如何对文件进行上传、下载和删除操作。

前提条件

- 已**构建数据模型**。
- 已**登录应用运行态**。

上传文件

步骤1 在“数据模型管理 > 数据实例”中，选择数据实体“Industrial_File”，单击“创建”。

步骤2 在展开的创建实例数据页面，根据文件类型和文件大小，在“扩展属性”的“ext.LargeFiles”中，选择不同的上传方式将文件上传至实例中。

图 7-5 创建数据实例

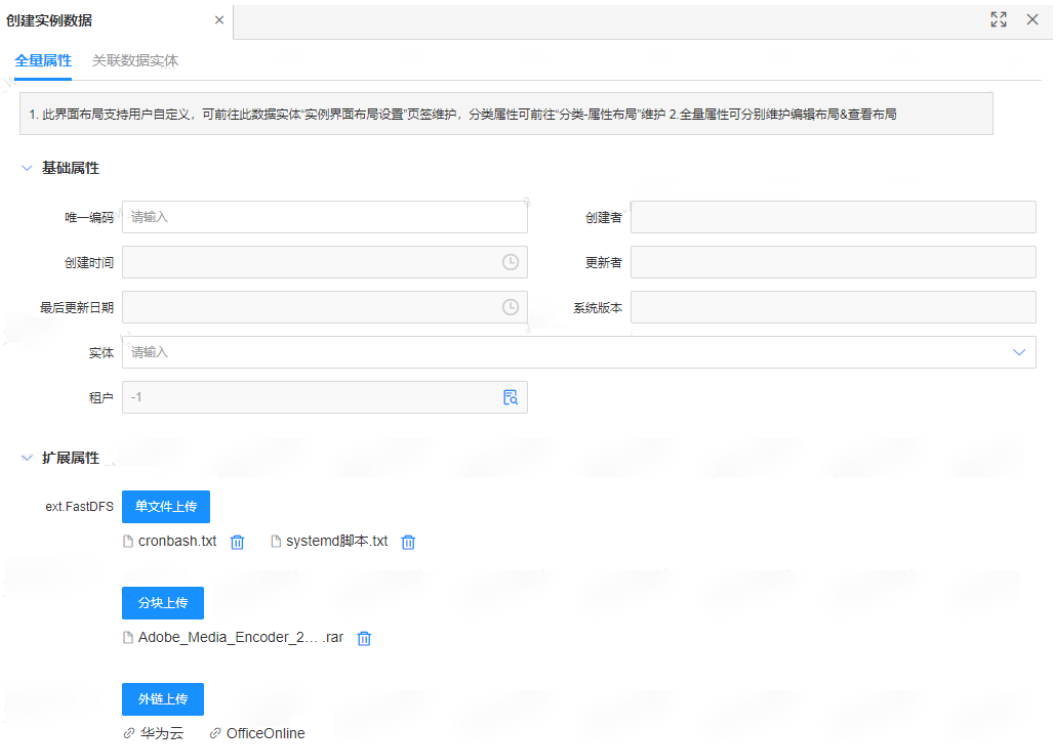


表 7-3 上传文件

上传方式	说明
单文件上传	适用于一次HTTP请求交互即可完成上传的场景。 单文件上传的大小不能超过100MB。
分块上传	适用于如下场景： • 当文件大小超过100MB时，使用分块上传可实现并行上传多个分块以加快上传速度。 • 网络环境较差时，建议使用分块上传。当出现单一分块上传失败的情况，您仅需重新上传文件即可对未完成上传的分块进行断点续传，从而提高整体的上传成功率。
外链上传	适用于存放合作伙伴和友情链接、行业相关论坛、优质博客和文章平台等外链场景。

说明

- 如果数据实例中已上传某个文件，后续再上传该文件时，该文件具有闪传能力。例如在数据实例中上传了A文件，后续再上传A文件时，该A文件具有闪传能力。
- 如果您对上传文件的格式有具体要求，可前往**文件类型白名单**进行配置。

步骤3 完成上传后，单击“确定”。

----结束

下载文件

说明

您可以通过应用运行态对数据实例中已有的文件进行下载操作。

下载文件前，请您确保数据实例中已存在文件。如未上传文件，请先参见**上传文件**进行操作。

步骤1 在“数据模型管理 > 数据实例”中，选择数据实体“Industrial_File”，单击数据实例的唯一编码。

步骤2 在展开的实例数据页面，根据实际需求进行下载操作。

- 下载单个文件：单击需要下载的文件名称，即可下载。
- 批量下载文件：单击“下载全部”，即可下载。

----结束

删除文件

说明

您可以通过应用运行态对数据实例中已有的文件进行删除操作。

此删除操作只是删除文件和数据模型对应数据实例的关联关系，不是真实地删除文件。

步骤1 在“数据模型管理 > 数据实例”中，选择数据实体“Industrial_File”，找到文件所在的数据实例，单击。

步骤2 在展开的实例数据页面，根据实际需求进行删除操作。

- 删除单个文件：找到需要删除的文件名称，单击。
- 删除外链：
  a. 单击“外链上传”。
  b. 在弹出的窗口中，勾选需要删除的外链，单击“删除”。
  c. 在弹出的提示框中，单击“确定”。

步骤3 单击“确定”。

----结束

7.3.2 通过 API 方式上传简单文件

功能介绍

通过“文件管理”接口“upload_uploadFile”可以将本地的文件上传至工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）中。完成文件上传后，可在调用数据实例的创建/更新接口时，将文件与该数据实例进行关联，从而实现对象化管理文件。

说明

“upload_uploadFile”接口最大支持上传100M文件。如需上传大于100M的文件，请使用**分块上传**的接口。

本文仅指导您如何通过API方式上传简单文件。关于如何创建/更新数据实例的接口请参见**全量数据服务**。

URI

- URI格式：

  POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/upload/uploadFile

- 参数说明：

  表 7-4 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
modelNumber	是	String	数据模型的编码。
modelName	是	String	数据模型的英文名称。
dataType	否	String	数据模型的类型。 ● entity：数据实体 ● relation：关系实体
attributeName	是	String	数据模型的属性英文名称。
applicationId	是	String	应用ID。
username	是	String	用户名。
storageType	是	Integer	文件的存储类型。 ● 0：对象存储 ● 1：BLOB
instanceId	是	String	数据实例的唯一编码。
exaAttr	是	String	是否为扩展属性。 ● 0：非扩展属性 ● 1：扩展属性

参数	是否必填	参数类型	描述
encrypted	否	Boolean	是否加密。当“storageType”设置为“0”时，需要设置此参数。 ● true：加密 ● false：不加密

请求参数

表 7-5 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

表 7-6 请求 Body 参数

参数	是否必选	参数类型	描述
files	是	MultipartFile[]	需要上传的文件列表。
file	否	MultipartFile	需要上传的文件。

响应参数

表 7-7 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 ● SUCCESS：成功 ● FAIL：失败
data	List	调用的返回结果。
errors	List	异常信息列表。

请求示例

POST https://**dme.cn-north-4.huaweicloud.com**/rdm_**fce01234567d41828cf3473b07fa7ae2**_app/services/rdm/basic/api/upload/uploadFile?**applicationId**=e22c66fb1d05453fa33162772e3cc9c0**&attributeName**=LargeFiles**&dataType**=entity**&encrypted**=false**&exaAttr**=1**&modelName**=Industrial_File**&modelNumber**=DM00127283**&storageType**=0**&userna**

```
me=XDM_User
X-Auth-Token: ABCDEFJ....

[MultipartFile Form files]
```

响应示例

```
{
    "result": "SUCCESS",
    "data": [
        "563740223079452672"
    ],
    "errors": []
}
```

7.3.3 通过 API 方式分块上传文件

操作场景

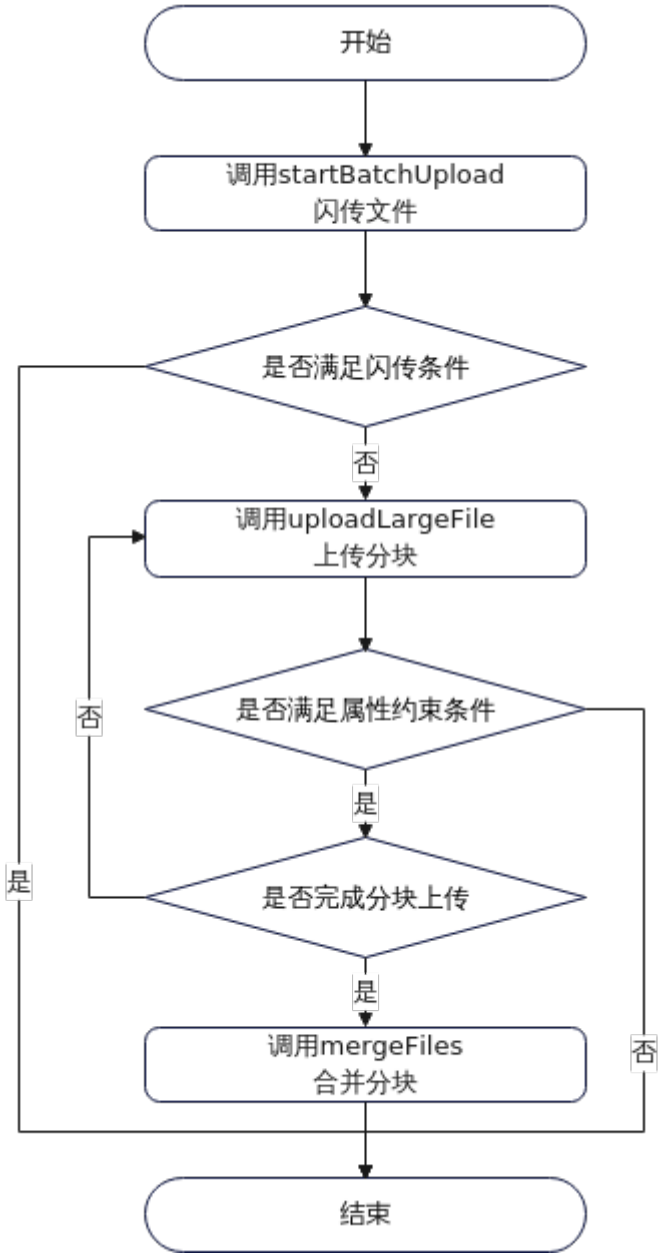
工业数字模型驱动引擎-数据建模引擎（xDM Foundation，简称xDM-F）可以将待上传的文件分成多个分块分别上传，上传完成后再调用“file_mergeFiles”接口将这些分块合并成一个对象存储至某个数据模型中。

完成文件的分块上传后，可在调用数据实例的创建/更新接口时，将文件与该数据实例进行关联，从而实现对象化管理文件。

说明

本文仅指导您如何通过API方式分块上传文件。关于如何创建/更新数据实例的接口请参见**全量数据服务**。

操作流程



1. 使用分块上传文件之前，您必须先调用“文件管理”的“file_startBatchUpload”接口校验待上传的文件之前是否上传至某个数据模型中。如果您之前已经上传某个文件，现在需要重新上传，那么该文件具有闪传能力，会自动闪传至数据模型中，无需重新分块上传。如果您没有上传过该文件，调用“file_startBatchUpload”接口时，初始化分块上传，成功执行此请求后将返回“docId”和“fileId”，用于后续的分块请求。
2. 初始化分块上传后，调用“文件管理”的“upload_uploadLargeFile”接口根据指定的“docId”、“fileId”、“chunk”等参数值上传分块文件。
3. 当使用“upload_uploadLargeFile”完成所有分块上传后，您必须调用“file_mergeFiles”来完成整个文件的分块上传。在使用该接口时，您必须在请求体中给出“docId”、“fileId”和“chunk”等参数值，用来校验每个分块的有效性。当所有的分块验证通过后，系统将把这些分块合并成一个完整的文件。

操作步骤

步骤1 闪传文件。

说明

如果您已上传某个文件，再上传该文件时，您只需执行本步骤即可完成文件上传。

- 接口相关信息

表 7-8 startBatchUpload 接口

接口信息	说明
URI格式	POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/file/startBatchUpload – Endpoint：必填，String类型，承载REST服务端点的服务器域名或IP地址。 – appID：必填，String类型，应用ID。 – applicationId：必填，String类型，应用ID。 – model_name：必填，String类型，数据模型的英文名称。 – model_number：选填，String类型，数据模型的编码。 – attribute_name：必填，String类型，数据模型的属性英文名称。 – file_name：选填，String类型，待上传文件的名称。本参数与fileName参数必须二选一。不能同时为空，且优先级低于fileName。 – file_size：必填，String类型，待上传文件的大小。 – chunks：必填，String类型，待上传文件的分块数量。您可以根据**构建数据模型**时创建属性的约束（分块大小）计算待上传文件的分块数量。 – check_code：必填，String类型，文件唯一校验码，即文件的哈希值。 – instance_id：必填，String类型，数据实例的唯一编码。 – username：必填，String类型，用户名称。 – fileId：选填，String类型，文件ID。 – encrypted：选填，Boolean类型，用户名称。 – exaAttr：选填，String类型，是否为扩展属性。 ▪ 0：非扩展属性 ▪ 1：扩展属性

接口信息	说明
请求参数	– Header参数 X-Auth-Token：必填，String类型，用户的token。 – Body参数 fileName：选填，String类型，待上传文件的名称。本参数与file_name参数必须二选一。不能同时为空，且优先级高于file_name。
响应参数	– result：String类型，调用是否成功。 ▪ SUCCESS：成功 ▪ FAIL：失败 – data：List类型，调用的返回结果。 ▪ isMergedFile：是否合并文件。 false：否 true：是 ▪ docId：文档ID。 ▪ fileId：文件ID。 – errors：List类型，异常信息列表。

- 请求示例

  新上传一个101MB的文件“testFile.zip”，由于在添加“文件”类型属性时“分块大小”为默认值（5MB），文件需分为21个分块。

  ```
  POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/file/startBatchUpload?applicationId=fce01234567d41828cf3473b07fa7ae2&model_name=Craft_File&attribute_name=LargeFiles&file_name=testFile.zip&file_size=103424&chunks=21&check_code=1234567890&instance_id=1&username=XDM_User
  X-Auth-Token: ABCDEFJ....
  ```

- 响应示例

  ```
  {
    "result": "SUCCESS",
    "data": [
      {
        "isMergedFile": false,
        "docId": "0000018BB1E33DC685E9C0045DFC7291",
        "fileId": "564032141298503680"
      }
    ],
    "errors": []
  }
  ```

步骤2 分块上传。

根据**调用startBatchUpload接口**设置的分块数量，依次执行uploadLargeFile接口。

- 接口相关信息

表 7-9 uploadLargeFile 接口

接口信息	说明
URI格式	POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/upload/uploadLargeFile – Endpoint：必填，String类型，承载REST服务端点的服务器域名或IP地址。 – appID：必填，String类型，应用ID。 – applicationId：必填，String类型，应用ID。 – modelNumber：必填，String类型，数据模型的编码。 – modelName：必填，String类型，数据模型的英文名称。 – attributeName：必填，String类型，数据模型的属性英文名称。 – docId：必填，String类型，文档ID，即**调用startBatchUpload接口**返回的docId。 – fileId：必填，String类型，文件ID，即**调用startBatchUpload接口**返回的fileId。 – fileName：必填，String类型，待上传文件的名称。 – checkCode：必填，String类型，文件的唯一校验码，即文件的哈希值。 – chunk：必填，Integer类型，待上传的分块位数。例如您上传第5块分块，填写5。 – storageType：选填，Integer类型，文件的存储类型。 ▪ 0：对象存储 ▪ 1：BLOB – exaAttr：选填，String类型，是否为扩展属性。 ▪ 0：非扩展属性 ▪ 1：扩展属性 – username：选填，String类型，用户名。
请求参数	Header参数 X-Auth-Token：必填，String类型，用户的token。

接口信息	说明
响应参数	– result：String类型，调用是否成功。 ▪ SUCCESS：成功 ▪ FAIL：失败 – data：List类型，调用的返回结果。 – errors：List类型，异常信息列表。

- 请求示例

  根据**请求示例**的文件分块数量，依次执行如下接口。为篇幅起见，这里以上传第1个分块为例。

  ```
  POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/upload/uploadLargeFile?attributeName=LargeFiles&modelName=Craft_File&applicationId=fce01234567d41828cf3473b07fa7ae2&fileId=5640321412985036 80&fileName=testFile.zip&checkCode=1234567890&chunk=1&docId=0000018BB1E33DC685E9C0045DFC7291&modelNumber=DM00127285
  X-Auth-Token: ABCDEFJ....

  [MultipartFile Form files]
  ```

- 响应示例

  ```
  {
     "result": "SUCCESS",
     "data": [
        "564091302493294592"
     ],
     "errors": []
  }
  ```

步骤3 分块合并。

- 接口相关信息

表 7-10 mergeFiles 接口

接口信息	说明
URI格式	POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/file/mergeFiles – Endpoint：必填，String类型，承载REST服务端点的服务器域名或IP地址。 – appID：必填，String类型，应用ID。 – applicationId：必填，String类型，应用ID。 – modelName：必填，String类型， – attributeName：必填，String类型， – docId：必填，String类型，文档ID，即**调用startBatchUpload接口**返回的docId。 – fileId：必填，String类型，文件ID，即**调用startBatchUpload接口**返回的fileId。 – fileName：必填，String类型，待上传文件的名称。 – checkCode：必填，String类型，文件的唯一校验码，即文件的哈希值。 – instanceId：选填，String类型，数据实例的唯一编码。 – exaAttr：选填，String类型，是否为扩展属性。 ▪ 0：非扩展属性 ▪ 1：扩展属性
请求参数	Header参数 X-Auth-Token：必填，String类型，用户的token。
响应参数	– result：String类型，调用是否成功。 ▪ SUCCESS：成功 ▪ FAIL：失败 – data：List类型，调用的返回结果。 ▪ fileSize：文件的大小。 ▪ fileId：文件ID。 – errors：List类型，异常信息列表。

- 请求示例

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/file/mergeFiles?applicationId=fce01234567d41828cf3473b07fa7ae2&modelNumber=DM00127285&modelName=Craft_File&attributeName=LargeFiles&fileName=testFile.zip&checkCode=1234567890&docId=0000018BB1E33DC685E9C0045DFC7291&exaAttr=0&fileId=564032141298503680
X-Auth-Token: ABCDEFJ....
```

- 响应示例

```
{
   "result": "SUCCESS",
   "data": [
      {
         "fileSize": "103424",
         "fileId": "564032141298503680"
      }
   ],
   "errors": []
}
```

----结束

7.3.4 通过 API 方式下载文件

功能介绍

通过“文件管理”接口“file_downloadFile”可以下载已上传的文件。

URI

- URI格式：

  GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/file/downloadFile

- 参数说明：

  表 7-11 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
application_id	是	String	应用ID。
attribute_name	是	String	数据模型的属性英文名称。
file_ids	是	String	已上传文件的ID。 如需批量下载文件，多个文件以英文逗号隔开。 例如“file_ids=559100724294721536,559100505549185024”。
instance_id	是	String	数据实例的唯一编码。

参数	是否必填	参数类型	描述
is_master_attr	是	String	是否为扩展属性。 ● 0：非扩展属性 ● 1：扩展属性
model_name	是	String	数据模型的英文名称。

请求参数

表 7-12 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

响应参数

无。

请求示例

GET https://**dme.cn-north-4.huaweicloud.com**/rdm_**fce01234567d41828cf3473b07fa7ae2**_app/services/rdm/basic/api/file/downloadFile?**model_name**=Industrial_File**&model_number**=DM00127281**&instance_id**=559100827910807552**&application_id**=c8198a59e15248b6a1a1269075ef5541**&is_master_attr**=0**&attribute_name**=file**&file_ids**=559100724294721536
X-Auth-Token: ABCDEFJ....

7.3.5 通过 API 方式预览文件（图片）

功能介绍

通过“文件管理”接口“file_images”可以用来查看指定的已上传的文件（图片）。

URI

- URI格式：

  GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/file/images?fileId=**{FileID}**
- 参数说明：

表 7-13 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
fileId	是	String	已上传文件的ID。 该参数会以“?fileId={FileID}”格式拼接在URI后面，其中“FileID”表示需要预览文件（图片）的ID。

请求参数

表 7-14 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

响应参数

无。

请求示例

```
GET https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/file/images?fileId=563781483752329216
X-Auth-Token: ABCDEFJ....
```

8 树形结构实践

概述

树形结构指的是数据元素之间存在着“一对多”的树形关系的数据结构，是一类重要的非线性数据结构。在树形结构中，树根节点没有前驱节点，其余每个节点有且只有一个前驱节点。叶子节点没有后续节点，其余每个节点的后续节点数可以是一个也可以是多个。

为了便于用户维护数据之间的父子关系，工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）提供树形结构功能。用户可在应用设计态创建数据实体时，在功能列表中勾选“树形结构”，即可使用此功能。

具有树形结构功能的数据实体会自动生成五个系统属性“ParentNode”、“LeafFlag”、“RawFullPath”、“RootNode”和“FullPath”，同时对外提供如下7个树形相关接口。

表 8-1 树形结构相关接口

接口名称	描述
GetAllParentList	获取所有父节点（前驱节点）：获取该叶子节点之上的所有父节点，可往上查到顶层父节点。
BatchRemoveChildNode	批量移除叶子节点。
BatchAddChildNode	批量添加叶子节点/批量更新叶子节点的父节点。
GetChildList	获取单层叶子节点。
GetParent	获取单层父节点。
Refresh	刷新所有叶子节点。
GetRoot	获取根节点/顶层父节点。

更多树形结构的接口信息请参见**全量数据服务**。

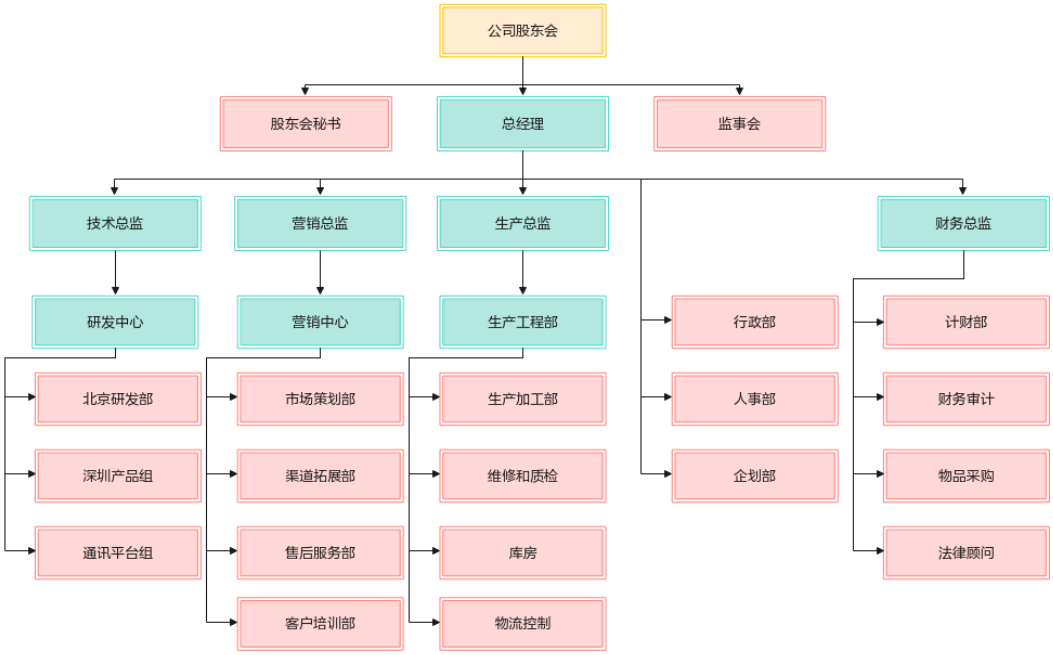
使用说明

- 如果数据实体的父模型存在树形结构，该数据实体自动继承树形结构功能，且不可去勾选。如果后续父模型删除了树形结构功能，该数据实体将不再继承父模型的树形结构功能，但可以自己重新勾选树形结构功能。
- 如果数据实体为Master-Version模型实体（即“父模型”选择为“VersionObject”），则只能在Master模型上选择树形结构功能，Version模型和Branch模型不支持选择树形结构功能。

示例场景

常见的应用系统（如ERP、采购、财务等）中都会包含人员管理，而人员管理往往都是以组织的形式进行管理和展示。这种场景下，树形结构是此功能的基础核心，所有人员的权限操作都是依附于组织。

图 8-1 某企业组织架构图



某企业的组织架构如图8-1所示，根据树形结构进行分析，可得知：

- 黄色方框的部门没有父节点，即为根节点。
- 红色方框的部门没有子节点，即为叶子节点。
- 绿色方框的部门均有子节点。

为此，我们可在构建数据模型时，创建一个具有“树形结构”功能的数据实体用于管理此企业的组织。而后在其数据实体实例化时，通过指定“ParentNode”的属性值来实现组织管理。

如下操作步骤仅指导您如何通过可视化页面使用树形结构功能。关于如何通过API方式使用树形结构功能请参见全量数据服务。

操作步骤

步骤1 登录应用设计态。

步骤2 在左侧导航栏中，选择“数据模型管理 > 数据实体”，进入数据实体页面。

步骤3 单击“创建”，根据如下主要信息，创建一个具有“树形结构”功能的数据实体（例如OrganizationalStructure）。

图 8-2 树形结构

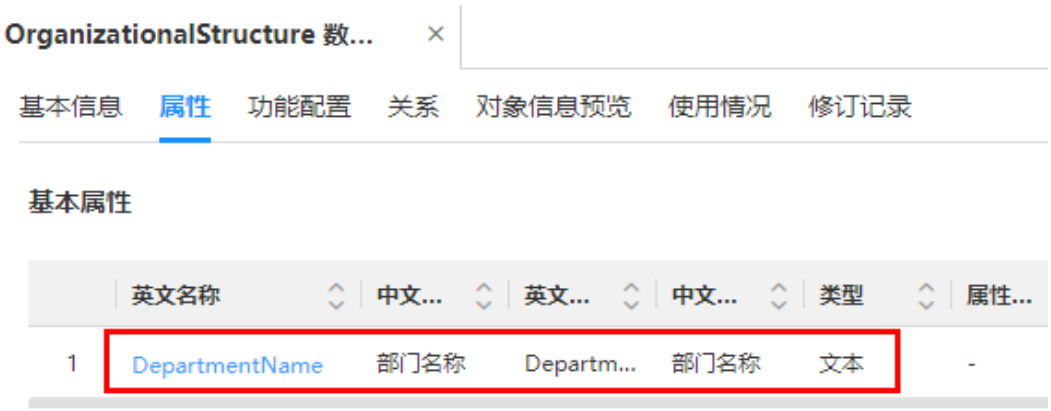
表 8-2 主要参数信息

类型	参数	参数说明
基本信息	英文名称	OrganizationalStructure。
	中文名称	组织结构。
	中文描述	某公司的部门结构。
	模型类型	选择“实体模型”。
	父模型	保持默认，此处以BasicObject为例。

类型	参数	参数说明
功能列表	可选功能	添加“树形结构”功能。

步骤4 选择“属性”页签，新增一个“文本”类型的属性“DepartmentName”，用于记录部门名称。

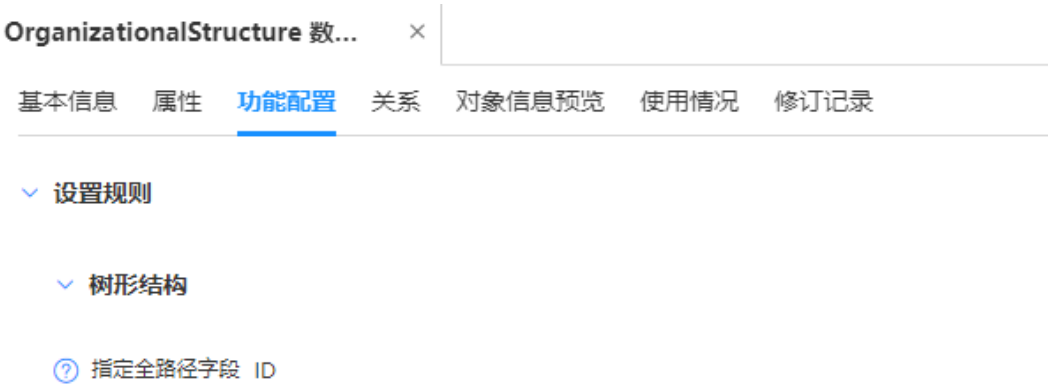
图 8-3 属性



步骤5 选择“功能配置”页签，在“设置规则 > 树形结构”栏设置树形结构的规则。

此处以设置“指定全路径字段”为“ID”为例。

图 8-4 功能配置



步骤6 依次完成“发布数据实体 > 发布应用 > 部署应用”的操作，具体操作请参见**发布数据实体**、**发布应用**和**部署应用**。

步骤7 待应用完成部署后，**登录应用运行态**。

步骤8 在“数据模型管理 > 数据实体”中，找到并编辑“OrganizationalStructure”数据实体，将“实例界面显示”设置为“是”。

步骤9 在“数据模型管理 > 数据实例”中，选择数据实体“OrganizationalStructure”，根据**示例场景**的企业组织架构，依次创建如下数据实例。

图 8-5 创建数据实例

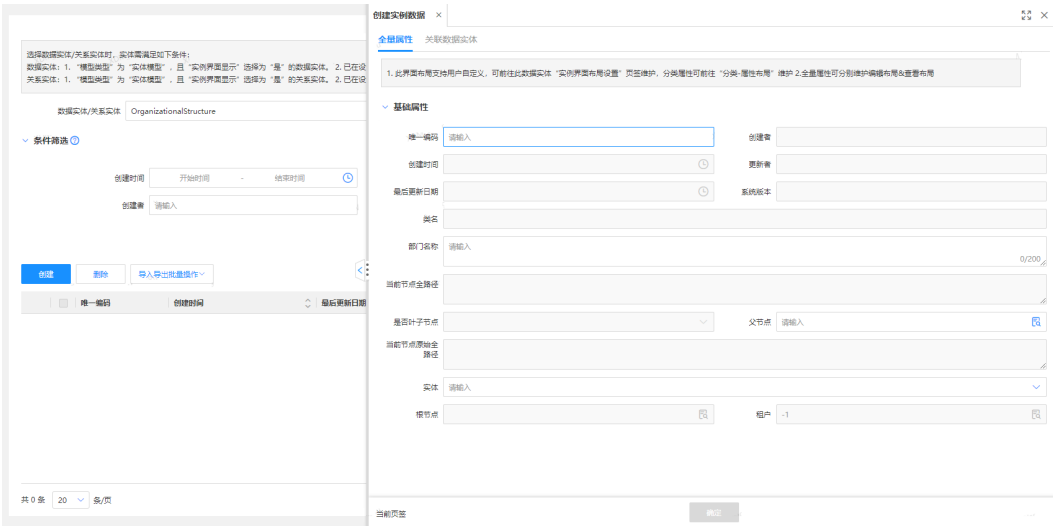


表 8-3 数据实例信息

唯一编码（ID）	部门名称（DepartmentName）	父节点（ParentNode）
10001	公司股东会	-
20001	股东会秘书	10001
20002	总经理	10001
20003	监事会	10001
30001	技术总监	20002
30002	营销总监	20002
30003	生产总监	20002
30004	财务总监	20002
.....	.....	.....

为篇幅起见，**表8-3**仅列出部分数据实例信息。

----结束

9 通过基线功能管理产品历史状态

9.1 方案概述

基线（BaseLine）是软件、文档、源码、产品或其他产出物的一个稳定版本，是进一步演进的基础。在一个产品开发流程中，会先后经历计划阶段、开发阶段、验证阶段、发布阶段和维护阶段。而进入每个阶段的标志一般称为DCP（Decision Checkpoint，决策评审点）或TR（Technical Review，技术评审）点。您可以在这个DCP/TR点创建基线，为产品开发流程中的不同阶段提供一个定点或快照，用于在产品开发中的某个时刻需要重新生成开发环境，更新不稳定或不可信时需要取消变更，追溯产品开发的历史状态记录等场景。

在使用工业数字模型驱动引擎（Industrial Digital Model Engine，简称iDME）时，可通过工业数字模型驱动引擎-数据建模引擎（xDM Foundation，简称xDM-F）提供的基线管理功能将已构建的数据模型添加到基线中，用以记录基线下的数据模型实例数据的变更、锁定/解锁基线。

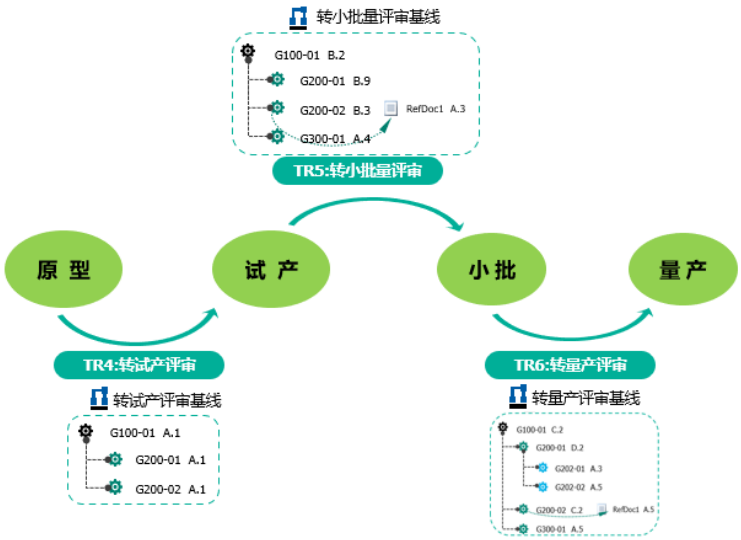
示例场景

某工业产品的开发流程如图9-1所示，先后经历了原型、试产、小批和量产四个阶段。

- 原型：对产品结构（如外貌、功能、用户体验等）进行规划和设计，期间会频繁修改。
- 试产：对产品设计的验证，期间材料结构不会变化，但会存在一些细节的变更。
- 小批：即小批量生产，将所有治具、夹具、机器、仪器、测试工具等按照量产的标准配置生产，用来验证产品是否能够量产。
- 量产：通过测试验证、规格审定后，大批量生产。

随着产品的迭代更新，会在原版本的产品上进行改良升级。此时，可分别在转试产的时间点、转小批量的时间点和转量产的时间点创建基线，每个基线分别记录当前基线成员的变化，便于对产品历史的追踪管理。

图 9-1 基线



操作流程

本文通过iDME的基线管理功能，为您演示开发某工业产品场景下的基线管理流程。

表 9-1 基线管理操作流程

主要操作流程	操作目的
创建数据模型及其实例数据	• 使用iDME的数据模型管理完成对业务数据对象的模型设计，并发布数据模型。 • iDME会将应用设计态创建的数据模型部署至应用运行态，完成数据模型的实例化、API调用等操作。
创建基线对象	使用iDME的全量数据服务API完成基线对象的创建。
为基线对象添加基线成员	使用iDME的全量数据服务API，将数据模型的实例化数据与基线对象进行关联。
锁定基线对象	使用iDME的全量数据服务API完成基线对象的锁定，不允许修改该基线对象。

使用说明

- 本方案以部分API为操作示例，如需了解更多基线管理API，请参见全量数据服务（“XDM基线对象”和“基线对象与被基线对象的关系”）。
- 基线对象锁定后，不能对该基线对象进行添加和删除基线成员、更新和删除基线对象的操作。
- 基线对象锁定后，不能对该基线对象下的基线成员进行撤销检出和删除的操作。
- 基线对象锁定后，支持对该基线对象下的基线成员进行修改、修订、检入和检出的操作。如果基线成员为其他基线对象且也被锁定，则不允许修改。
- 如需解锁已锁定的基线对象，可使用“XDM基线对象”的接口“BaseLine_enable”进行解锁。

9.2 创建数据模型及其实例数据

操作场景

本章节以可视化页面的方式创建M-V模型数据实体及其数据实例为操作示例。

前提条件

- 已**创建应用**。
- 已**购买数据建模引擎**。

操作步骤

步骤1 **登录应用设计态**。

步骤2 在左侧导航栏中，选择“数据模型管理 > 数据实体”，进入“数据实体”页面。

步骤3 创建两个M-V模型数据实体，具体操作请参见**创建数据实体**。

例如，创建如**图9-2**和**图9-3**所示的两个M-V模型数据实体（数据实体的名称分别为“治具”和“夹具”）。

图 9-2 M-V 模型数据实体（治具）

图 9-3 M-V 模型数据实体（夹具）

步骤4 勾选刚创建的两个数据实体，单击“发布”，发布数据实体。

图 9-4 发布数据实体

步骤5 在右上方单击“应用发布”，确认应用发布的信息，并根据界面提示完成应用发布。

步骤6 在左上方单击“控制台”，切换至iDME控制台。

步骤7 在左侧导航栏中，单击“运行服务 > 数据建模引擎”，进入数据建模引擎页面。

步骤8 找到待部署的应用，并根据页面提示**部署应用**。

步骤9 待应用部署完成后，**登录应用运行态**。

步骤10 在左侧导航栏选择“数据模型管理 > 数据实体”，修改**步骤3**创建的两个数据实体，将基本信息中的“实例界面显示”修改为“是”。

图 9-5 实例界面显示

步骤11 在左侧导航栏选择“数据模型管理 > 数据实例”，进入“数据实例”页面。

步骤12 分别选择**步骤3**创建的数据实体，创建相应的数据实例，具体操作请参见**创建数据实例**。

例如，为数据实体“夹具”创建“装配夹具”、“检验夹具”、“机床夹具”和“焊接夹具”四个数据实例。

图 9-6 创建数据实例

----结束

9.3 创建基线对象

操作场景

使用iDME提供的“XDM基线对象”接口“BaseLine_create”指导用户创建基线对象。

URI

- URI格式：

  POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/common/api/BaseLine/create

- 参数说明：

  表 9-2 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。

请求参数

表 9-3 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

表 9-4 请求 Body 参数

参数	是否必选	参数类型	描述
id	否	Int	唯一编码。
name	否	String	基线对象的名称。
nameEn	否	String	基线对象的英文名称。
description	否	String	基线对象的描述。
descriptionEn	否	String	基线对象的英文描述。

为篇幅起见，这里只展示部分内容。更多参数信息，您可以在**全量数据服务**进行查看。

响应参数

表 9-5 响应 Body 参数

参数	参数类型	描述
id	Int	唯一编码。
rdmExtensionType	String	实体类型。
rdmVersion	Integer	系统版本。
className	String	实体名称。
name	String	基线对象的名称。
nameEn	String	基线对象的英文名称。
description	String	基线对象的描述。
descriptionEn	String	基线对象的英文描述。
disableFlag	Boolean	是否锁定。 ● true：锁定。 ● false：解锁，默认为false。新建的基线对象默认为解锁状态，如需锁定基线对象，请参见**锁定基线对象**。

为篇幅起见，这里只展示部分内容。更多参数信息，您可以在**全量数据服务**进行查看。

请求示例

根据**方案概述**的示例场景，以创建转试产评审基线为例。

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/common/api/BaseLine/create
X-AUTH-TOKEN: ABCDEFJ....
{
    "params": {
        "name": "转试产",
        "nameEn": "TurnToTrialProduce",
        "description": "转试产评审基线",
        "descriptionEn": "Turn to trial-produce review baseline"
    }
}
```

响应示例

```
{
    "result": "SUCCESS",
    "data": [
        {
            ......
            "id": "553881194513567744",
            "rdmVersion": 1,
            "rdmDeleteFlag": 0,
```

```
            "rdmExtensionType": "BaseLine",
            "tenant": {
                ......
            },
            "className": "BaseLine",
            "descriptionEn": "Turn to trial-produce review baseline",
            "name": "转试产",
            "description": "转试产评审基线",
            "nameEn": "TurnToTrialProduce",
            "disableFlag": false
        }
    ],
    "errors": []
}
```


9.4 为基线对象添加基线成员

操作场景

使用iDME提供的“基线对象与被基线对象的关系”接口“BaseLineLink_create”或“BaseLineLink_batchCreate”，将已创建的数据模型实例添加至基线对象中。本章节以“BaseLineLink_create”为例。

URI

- URI格式：

  POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/common/api/BaseLineLink/create

- 参数说明：

  表 9-6 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。

请求参数

表 9-7 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

表 9-8 请求 Body 参数

参数	是否必选	参数类型	描述
source	是	Object	基线对象信息，需输入基线对象的ID和实体名称。 • id：基线对象ID，即**创建基线对象**返回的ID。 • clazz：基线对象的实体名称“BaseLine”。
target	是	Object	基线成员信息，需输入基线成员的ID和实体名称。 • id：数据实例ID，即**创建数据模型及其实例数据**的数据实例唯一编码。 • clazz：数据实例的实体名称。例如“Fixture”。

为篇幅起见，这里只展示部分内容。更多参数信息，您可以在**全量数据服务**进行查看。

响应参数

表 9-9 响应 Body 参数

参数	参数类型	描述
id	Int	唯一编码。
rdmExtensionType	String	实体类型。
rdmVersion	Integer	系统版本。
className	String	实体名称。
name	String	基线对象的名称。
nameEn	String	基线对象的英文名称。
description	String	基线对象的描述。
descriptionEn	String	基线对象的英文描述。
disableFlag	Boolean	是否锁定。 • true：锁定。 • false：解锁，默认为false。

为篇幅起见，这里只展示部分内容。更多参数信息，您可以在**全量数据服务**进行查看。

请求示例

使用“BaseLineLink_create”接口将数据实例“焊接夹具”添加至基线对象“转试产”。

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/common/api/BaseLineLink/create
X-AUTH-TOKEN: ABCDEFJ....
{
   "params": {
      "source":{
         "id":553881194513567744,
         "clazz":"BaseLine"
      },
      "target":{
         "id":553607469406887936,
         "clazz":"Fixture"
      }
   }
}
```

响应示例

为篇幅起见，这里只展示部分内容。

```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": "553956041486766080",
         "rdmExtensionType": "BaseLineLink",
         "tenant": {
            ......
         },
         "className": "BaseLineLink",
         "source": {
            "id": "553881194513567744",
            "rdmExtensionType": "BaseLine",
            "tenant": {
               ......
            },
            "className": "BaseLine",
            "descriptionEn": "Turn to trial-produce review baseline",
            "name": "转试产",
            "description": "转试产评审基线",
            "nameEn": "TurnToTrialProduce",
            "state": null,
            "disableFlag": false
         },
         "target": {
            "id": "553953656563568640",
            "rdmExtensionType": "Fixture",
            "className": "Fixture",
            "name": "焊接夹具",
            "master": {
               "id": "553953656563568641",
               "rdmExtensionType": "FixtureMaster",
               "tenant": {
                  ......
               },
               "className": "FixtureMaster"
            },
            "branch": {
               "id": "553953656563568642",
```

```
                "rdmExtensionType": "FixtureBranch",
                "className": "FixtureBranch",
                "version": "A"
            },
            "versionCode": 1,
            "iteration": 1,
            "version": "A",
            "latestVersion": true,
            "workingCopy": false,
            "workingState": {
                "code": "CHECKED_IN",
                "cnName": "已检入",
                "enName": "checked in",
                "alias": "CHECKED_IN"
            }
        }
    }
  ],
  "errors": []
}
```


9.5 锁定基线对象

操作场景

使用iDME提供的“XDM基线对象”接口“BaseLine_disable”将基线对象进行锁定，不允许修改该基线对象。

URI

- URI格式：

  POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/common/api/BaseLine/disable

- 参数说明：

  表 9-10 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。

请求参数

表 9-11 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

表 9-12 请求 Body 参数

参数	是否必选	参数类型	描述
ids	是	List	基线对象的ID列表。
modifier	否	String	处理人。
applicationId	否	String	应用ID。

响应参数

表 9-13 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 ● SUCCESS：成功 ● FAIL：失败
data	List	调用的返回结果。
errors	List	异常信息列表。

请求示例

锁定基线对象“转试产”，其ID为553881194513567744。

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/common/api/BaseLine/disable
X-AUTH-TOKEN: ABCDEFJ....
{
    "params":{
        "ids":[
            553881194513567744
        ]
    }
}
```

响应示例

```
{
    "result": "SUCCESS",
    "data": [
        1
    ],
    "errors": []
}
```

10 通过事务型任务 API 实现事务一致性

10.1 方案概述

背景信息

在应用开发过程中，可能会存在一个核心业务逻辑的执行需要同时调用多个下游业务进行处理的场景。因此，如何保证核心业务和多个下游业务的执行结果完全一致，是事务需要解决的主要问题。

以某工厂整体业务流程为例，生产管理部向生产车间发放生产工单这一核心操作的同时会涉及到生产部的领料加工、采购部的原材料采购等多个子系统的变更。当前业务的处理分支包括：

- 主分支生产工单系统状态更新：由待处理变更为生产中。
- 加工系统状态变更：新增产品加工记录，新增用料记录。
- 原材料采购状态变更：变更原材料数量，更新原材料库存表。

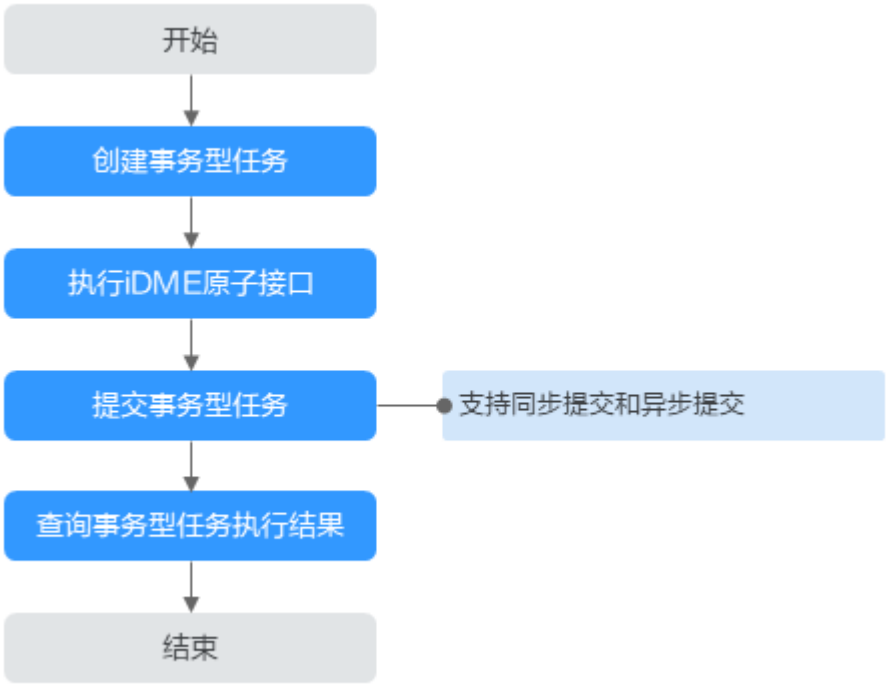
基于普通消息方案，将工单系统变更作为本地事务，剩下的系统变更作为普通消息的下游来执行。此时，消息下游分支和工单系统变更的主分支很容易出现不一致的现象。例如：

- 消息发送成功，工单没有执行成功，需要回滚整个事务。
- 工单执行成功，消息没有发送成功，需要额外补偿才能发现不一致。
- 消息发送超时未知，此时无法判断需要回滚工单还是提交工单变更。

为了保证上述分支的执行结果一致，具备提交、回滚和统一协调的能力，工业数字模型驱动引擎-数据建模引擎（ xDM Foundation，简称xDM-F ）提供事务型任务功能，编排多个原子接口，并与本地事务绑定，实现多接口提交结果的一致性，并提升接口调用速率和性能。

操作流程

图 10-1 事务型任务操作流程



1. **创建事务型任务**

   此接口需要设置执行iDME原子接口的数量，完成事务型任务的初始化。

2. **执行iDME原子接口**

   根据创建事务型任务时设置的原子接口数量，依次执行。执行的原子接口会被单独存储到iDME事务存储系统中，等待提交事务型任务返回执行结果后再统一提交。

3. **提交事务型任务**

   - 同步提交：原子接口按照指定顺序执行，需等待上一个原子接口执行完成后再执行下一个原子接口，适用于需要等待任务完成后才能继续执行的场景。例如，在读取文件或者网络请求时，应用需要等待文件读取完成后才能继续执行下一步操作。
   - 异步提交：各原子接口间串行执行，适用于需要同时执行多个任务的场景。例如，在多线程或者多进程的编程中，应用可以同时执行多个任务，提高应用的执行效率。

   所有原子接口都提交成功，则原子接口数据提交成功；否则，所有原子接口都进行数据回滚操作。

4. **查询事务型任务执行结果**

   您可以通过此接口快速搜索您想要的事务任务执行结果。

使用说明

- v1版本事务型任务接口最多支持执行5个原子接口。v2版本事务型任务接口，在异步提交事务型任务的原子接口取消数量限制。

- 不支持跨租户或跨数据源使用事务型任务功能。
- 不支持非数据库操作（如文件上传）。

10.2 步骤 1：创建事务型任务

操作场景

本章节指导用户通过调用API创建事务型任务。

URI

- URI格式：
  - v1版本：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/transaction-apis/transactions**?api-count=N**
  - v2版本：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v2/transaction-apis/transactions
- 参数说明：

表 10-1 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
api-count	是	Integer	**说明** 仅v1版本接口需要配置此参数。 需要执行原子接口的数量，取值范围：1-5。 该参数会以“?api-count=N”格式拼接在URI后面，其中“N”表示需要执行原子接口的数量。

请求参数

表 10-2 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

响应参数

表 10-3 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 ● SUCCESS：成功 ● FAIL：失败
data	List of **data**	事务型任务数据。
errors	List	异常信息列表。

表 10-4 data

参数	参数类型	描述
data.transactionId	Long	事务型任务ID。
data.taskNo	Integer	原子接口的执行序号，默认为0。
data.taskCount	Integer	原子接口的数量。 **说明** 仅v1版本接口会返回此参数信息。

请求示例

创建一个执行原子接口数量为2的事务型任务。

- v1版本请求示例：

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/transaction-apis/transactions?api-count=2
X-Auth-Token: ABCDEFJ....
```

- v2版本请求示例：

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v2/transaction-apis/transactions
X-Auth-Token: ABCDEFJ....
```

响应示例

- v1版本响应示例：

```
{
    "result": "SUCCESS",
    "data": [
        {
            "transactionId": 538322343718555649,
            "taskNo": 0,
            "taskCount": 2
        }
    ],
    "errors": []
}
```

- v2版本响应示例：

```
{
    "result": "SUCCESS",
    "data": [
        {
            "transactionId": 538322343718555650,
            "taskNo": 0
        }
    ],
    "errors": []
}
```

10.3 步骤 2：执行 iDME 原子接口

操作场景

创建事务型任务后，系统将根据原子接口执行序号依次执行原子接口。

本章节以创建“ExtDataModel”数据实体的数据实例为例，指导用户执行原子接口。

URI

- URI格式：

  POST http://**{Endpoint}**/rdm_**{appID}**_app/services/dynamic/api/**{entityName}**/**{atomicAPI}**

- 参数说明：

  表 10-5 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
entityName	是	String	实体的英文名称。
atomicAPI	是	String	实体的原子接口，例如create。

请求参数

表 10-6 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

参数	是否必选	参数类型	描述
X-Dme-Transaction-Task-Id	是	Long	事务型任务ID，即**步骤1：创建事务型任务**返回的transactionId。
X-Dme-Transaction-Task-No	是	Integer	当前原子接口的执行序号。

表 10-7 请求 Body 参数

参数	是否必选	参数类型	描述
id	是	Int	唯一编码。
rdmVersion	是	Int	系统版本。
rdmExtensionType	否	String	实体类型。
createTime	否	Date	创建时间。
creator	否	String	创建者。
lastUpdateTime	否	Date	最后更新时间。
modifier	否	String	更新者。
tenant	否	Object	租户。

根据实体类型、功能的不同，请求Body参数不同，您可以在**全量数据服务**查看实体对应API的具体参数。

响应参数

表 10-8 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 ● SUCCESS：成功 ● FAIL：失败
data	List	调用的返回结果，默认为空。
errors	List	异常信息列表。

请求示例

根据设置的原子接口执行序号，依次执行如下原子接口。

- 原子接口1：
  POST https://**dme.cn-north-4.huaweicloud.com**/rdm_**fce01234567d41828cf3473b07fa7ae2**_app/services/dynamic/api/**ExtDataModel/create**
  X-Auth-Token: ABCDEFJ....
  X-Dme-Transaction-Task-Id: 538322343718555649
  X-Dme-Transaction-Task-No: 1
- 原子接口2：
  POST https://**dme.cn-north-4.huaweicloud.com**/rdm_**fce01234567d41828cf3473b07fa7ae2**_app/services/dynamic/api/**ExtDataModel/create**
  X-Auth-Token: ABCDEFJ....
  X-Dme-Transaction-Task-Id: 538322343718555649
  X-Dme-Transaction-Task-No: 2

响应示例

原子接口1和原子接口2均返回如下结果：

```
{
   "result": "SUCCESS",
   "data": [],
   "errors": []
}
```

10.4 步骤 3：提交事务型任务

操作场景

工业数字模型驱动引擎-数据建模引擎（ xDM Foundation，简称xDM-F ）提供同步提交和异步提交两种提交方式。

URI

- **同步提交**
  - URI格式：
    PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/transaction-apis/transactions/**{transaction-id}**
  - 参数说明：

    表 10-9 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。

参数	是否必填	参数类型	描述
transaction-id	是	Long	事务型任务ID，即**步骤1：创建事务型任务**返回的transactionId。

- **异步提交**
  - URI格式：
    - v1版本：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/transaction-apis/transactions/async/**{transaction-id}**
    - v2版本：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v2/transaction-apis/transactions/async/**{transaction_id}**?**api_count=N**
  - 参数说明：

表 10-10 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
transaction-id	是	Long	**说明** 仅v1版本接口需要配置此参数。 事务型任务ID，即**步骤1：创建事务型任务**返回的transactionId。
transaction_id	是	Long	**说明** 仅v2版本接口需要配置此参数。 事务型任务ID，即**步骤1：创建事务型任务**返回的transactionId。

参数	是否必填	参数类型	描述
api_count	是	Integer	**说明** 仅v2版本接口需要配置此参数。 需要提交执行原子接口的数量。 该参数会以“?api_count=N”格式拼接在URI后面，其中“N”表示需要提交执行原子接口的数量。

请求参数

“同步提交”的请求参数和“异步提交”的请求参数相同。

表 10-11 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

响应参数

- 同步提交

表 10-12 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 • SUCCESS：成功 • FAIL：失败
data	List of **data**	调用的返回结果。
errors	List	异常信息列表。

表 10-13 data

参数	参数类型	描述
data.id	Long	事务型任务ID。

参数	参数类型	描述
data.status	String	事务型任务是否提交成功。**步骤2：执行iDME原子接口**执行的任一原子接口若执行失败，均表示事务型任务提交失败。 • success：成功 • fail：失败
data.tasks	List of **task**	事务型任务下的原子接口列表。

表 10-14 tasks

参数	参数类型	描述
data.tasks.result	String	原子接口的执行详情。
data.tasks.task_no	Integer	原子接口的执行序号。
data.tasks.task_status	String	原子接口的执行状态。 • success：成功 • fail：失败

- 异步提交

表 10-15 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 • SUCCESS：成功 • FAIL：失败
data	List	调用的返回结果，默认为空。
errors	List	异常信息列表。

请求示例

- 同步提交

```
PUT https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/transaction-apis/transactions/538322343718555649
X-Auth-Token: ABCDEFJ....
```

- 异步提交
  - v1版本请求示例：

```
PUT https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/transaction-apis/transactions/async/538322343718555649
X-Auth-Token: ABCDEFJ....
```

- v2版本请求示例：

```
PUT https://dme.cn-north-4.huaweicloud.com/
rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v2/transaction-apis/
transactions/async/538322343718555650?api_count=2
X-Auth-Token: ABCDEFJ....
```

响应示例

- 同步提交

```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": 538322343718555649,
         "status": "success",
         "tasks": [
            {
               "result": "{\"id\": 538327077254860800, \"tenant\": {\"id\": -1, \"code\": \"basicTenant\",
\"name\": \"basicTenant\", \"creator\": \"xdmAdmin\", \"modifier\": \"xdmAdmin\", \"className\":
\"Tenant\", \"createTime\": 1688108575000, \"dataSource\": \"DefaultDataSource\", \"rdmVersion\": 1,
\"description\": \"默认租户\", \"disableFlag\": false, \"rdmDeleteFlag\": 0, \"securityLevel\": \"internal
\", \"lastUpdateTime\": 1688108575000, \"rdmExtensionType\": \"Tenant\"}, \"creator\":
\"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"extAttrs\": [{\"name\": \"ExtString\",
\"type\": \"STRING\", \"value\": \"0123\"}, {\"name\": \"ExtDouble\", \"type\": \"DECIMAL\"}],
\"modifier\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"className\":
\"ExtDataModel\", \"createTime\": 1693367976851, \"extAttrMap\": {}, \"rdmVersion\": 1,
\"rdmDeleteFlag\": 0, \"lastUpdateTime\": 1693367976851, \"rdmExtensionType\": \"ExtDataModel\"}",
               "task_no": 1,
               "task_status": "success"
            },
            {
               "result": "{\"id\": 535837830857887744, \"tenant\": {\"id\": -1, \"code\": \"basicTenant\",
\"name\": \"basicTenant\", \"creator\": \"xdmAdmin\", \"modifier\": \"xdmAdmin\", \"className\":
\"Tenant\", \"createTime\": 1688108575000, \"dataSource\": \"DefaultDataSource\", \"rdmVersion\": 1,
\"description\": \"默认租户\", \"disableFlag\": false, \"rdmDeleteFlag\": 0, \"securityLevel\": \"internal
\", \"lastUpdateTime\": 1688108575000, \"rdmExtensionType\": \"Tenant\"}, \"creator\":
\"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"extAttrs\": [{\"name\": \"ExtString\",
\"type\": \"STRING\", \"value\": \"1234056789\"}, {\"name\": \"ExtDouble\", \"type\": \"DECIMAL\"}],
\"modifier\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"className\":
\"ExtDataModel\", \"createTime\": 1692774494000, \"extAttrMap\": {}, \"rdmVersion\": 38,
\"rdmDeleteFlag\": 0, \"lastUpdateTime\": 1693367978040, \"rdmExtensionType\": \"ExtDataModel\"}",
               "task_no": 2,
               "task_status": "success"
            }
         ],
         "task_count": 2
      }
   ],
   "errors": []
}
```

- 异步提交

```
{
    "result": "SUCCESS",
    "data": [],
    "errors": []
}
```

10.5 步骤 4：查询事务执行结果

操作场景

当您提交事务型任务后，可通过本接口快速查询事务型任务的执行结果。

URI

- URI格式：
  GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/transaction-apis/transactions/**{transaction-id}**
- 参数说明：

  表 10-16 URI 参数说明

参数	是否必填	参数类型	描述
Endpoint	是	String	承载REST服务端点的服务器域名或IP地址。
appID	是	String	应用ID。
transaction-id	是	Long	事务型任务ID，即**步骤1：创建事务型任务**返回的transactionId。

请求参数

表 10-17 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	用户的token。

响应参数

表 10-18 响应 Body 参数

参数	参数类型	描述
result	String	调用是否成功。 ● SUCCESS：成功 ● FAIL：失败
data	List of **data**	调用的返回结果。
errors	List	异常信息列表。

表 10-19 data

参数	参数类型	描述
data.id	Long	事务型任务ID。
data.status	String	事务型任务的提交状态。 • not start：未提交 • executing：执行中 • success：提交成功 • fail：提交失败，**步骤2：执行iDME原子接口**执行的任一原子接口若执行失败，均表示事务型任务提交失败。
data.tasks	List of **task**	事务型任务下的原子接口列表。

表 10-20 tasks

参数	参数类型	描述
data.tasks.result	String	原子接口的执行详情。
data.tasks.task_no	Integer	原子接口的执行序号。
data.tasks.task_status	String	原子接口的执行状态。 • success：成功 • fail：失败 • not start：未执行

请求示例

根据事务型任务ID，查询该事务型任务的执行结果。

```
GET https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/transaction-apis/transactions/538322343718555649
X-Auth-Token: ABCDEFJ....
```

响应示例

```
{
    "result": "SUCCESS",
    "data": [
        {
            "id": 538322343718555649,
            "status": "success",
            "tasks": [
                {
                    "result": "{\"id\": 538327077254860800, \"tenant\": {\"id\": -1, \"code\": \"basicTenant\", \"name\": \"basicTenant\", \"creator\": \"xdmAdmin\", \"modifier\": \"xdmAdmin\", \"className\": \"Tenant\", \"createTime\": 1688108575000, \"dataSource\": \"DefaultDataSource\", \"rdmVersion\": 1, \"description\": \"默认租户\", \"disableFlag\": false, \"rdmDeleteFlag\": 0, \"securityLevel\": \"internal\", \"lastUpdateTime
```

\": 1688108575000, \"rdmExtensionType\": \"Tenant\"}, \"creator\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"extAttrs\": [{\"name\": \"ExtString\", \"type\": \"STRING\", \"value\": \"0123\"}, {\"name\": \"ExtDouble\", \"type\": \"DECIMAL\"}], \"modifier\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"className\": \"ExtDataModel\", \"createTime\": 1693367976851, \"extAttrMap\": {}, \"rdmVersion\": 1, \"rdmDeleteFlag\": 0, \"lastUpdateTime\": 1693367976851, \"rdmExtensionType\": \"ExtDataModel\"}",
                    "task_no": 1,
                    "task_status": "success"
                },
                {
                    "result": "{\"id\": 535837830857887744, \"tenant\": {\"id\": -1, \"code\": \"basicTenant\", \"name\": \"basicTenant\", \"creator\": \"xdmAdmin\", \"modifier\": \"xdmAdmin\", \"className\": \"Tenant\", \"createTime\": 1688108575000, \"dataSource\": \"DefaultDataSource\", \"rdmVersion\": 1, \"description\": \"默认租户\", \"disableFlag\": false, \"rdmDeleteFlag\": 0, \"securityLevel\": \"internal\", \"lastUpdateTime\": 1688108575000, \"rdmExtensionType\": \"Tenant\"}, \"creator\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"extAttrs\": [{\"name\": \"ExtString\", \"type\": \"STRING\", \"value\": \"1234056789\"}, {\"name\": \"ExtDouble\", \"type\": \"DECIMAL\"}], \"modifier\": \"XDM_Developer 93172bbfd0f64437956d4c9de9345386\", \"className\": \"ExtDataModel\", \"createTime\": 1692774494000, \"extAttrMap\": {}, \"rdmVersion\": 38, \"rdmDeleteFlag\": 0, \"lastUpdateTime\": 1693367978040, \"rdmExtensionType\": \"ExtDataModel\"}",
                    "task_no": 2,
                    "task_status": "success"
                }
            ],
            "task_count": 2
        }
    ],
    "errors": []
}

11 基于图形化方式编排 API

11.1 方案概述

背景信息

追溯是维护工作的基本能力，维护工作中经常需要通过追溯排查问题、评估风险。

例如追溯出现问题的内存，需要先根据内存批次信息手工转换成采购商SN，然后手工分段查询当前状态，不仅效率低，而且查不到故障件的历史装配关系，出现追溯遗漏。一般情况下，在服务器存储硬件维护中，每周都有2~3次追溯需求。如果采用手工排查方式，计算维护人员、存储维护人员及MQE需要在CMES、悍马等多个系统手工导出数据，涉及十亿级数据量，使用Excel分析，简单的追溯每次至少1天时间；如果追溯的条件复杂（ 如18nm内存发货，批次、型号、工艺等组合查询），有时甚至需要1周，且无法保证质量，追溯的需求非常迫切。

解决方案

使用数字主线引擎（LinkX Foundation，简称LinkX-F）提供的图谱技术，构建出海量业务数据全关联图谱，可以联接内存的批次、容量、工艺、型号等信息，包括内存所在的父项条码、阶段去向、来料批次号、厂家名称、容量、型号等相关数据。通过多系统超大数据量处理，即可实现快速支撑连续性器件质量追溯一键式分析器件问题影响，有效提升器件质量问题分析处理效率。

在数字主线引擎中，聚合服务是数据服务的基本单位，也是实现业务功能的最小单元。利用聚合服务编排，通过拖拉拽图形化的方式，即可快速生成一个API。将数字主线引擎中开发的API集成到对应的数据查询应用，即可提供按厂家SN、21条码、整机SN等组合内存数据查询、数据导出、导入模板查询统计的能力，产品线硬件维护人员也可通过器件追溯应用对内存数据过滤查询、导出导入数据。利用数字主线引擎，轻松实现百亿级数据分钟级追溯，快速获得如下价值和收益。

- 计算&数据存储与机器视觉产品维护人员每周2-3次的追溯内存的效率，由原来的N天甚至1周提升到现在的1-10分钟。
- 支持1000条以内厂家SN、21条码、征集SN的文本输入查询，有效提升易用性。
- 支持10W+厂家SN、21条码数据查询，10分钟以内返回查询结果，快速高效。

示例场景

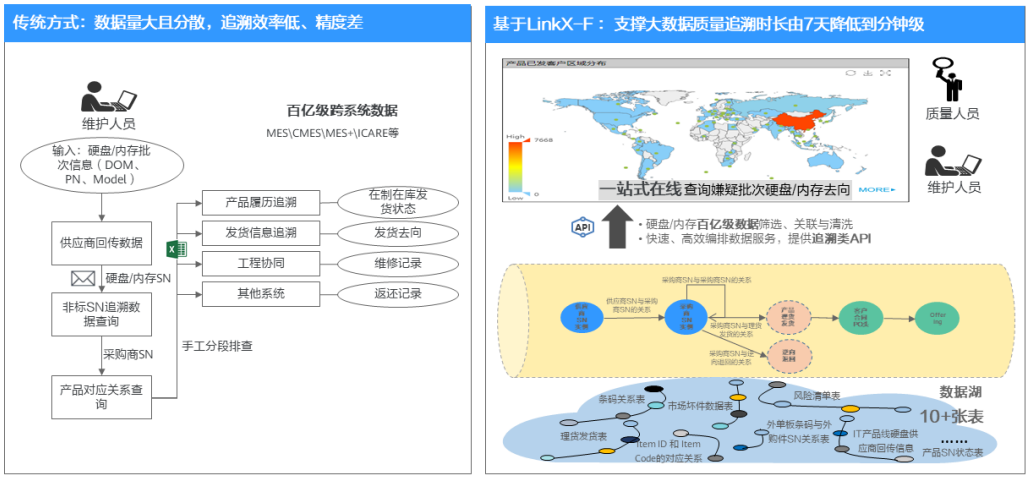
假设存在如下场景：

1. 维护工程师收到问题单板时，采集其采购商条码和相关器件编码，追溯到涉及器件的DateCode、LotCode，确认问题器件批次等信息。
2. 根据问题器件批次追溯到来料供应商，器件坏件影响的单板、模块以及影响客户、合同，制定应对措施并指导实施。
3. 同时识别同批次器件来料和在库状态，及时做好质量隔离措施。
4. 根据问题单板条码逆向追溯相关硬盘、内存等各种介质对应的供应商SN、批次等，支撑定位出嫌疑介质信息。根据嫌疑介质供应商SN等正向追溯相关硬盘阶段去向（在制在库、理货、发货、逆向退回、报废），及时采取召回、整改等措施将业务影响降到最低。

优势说明

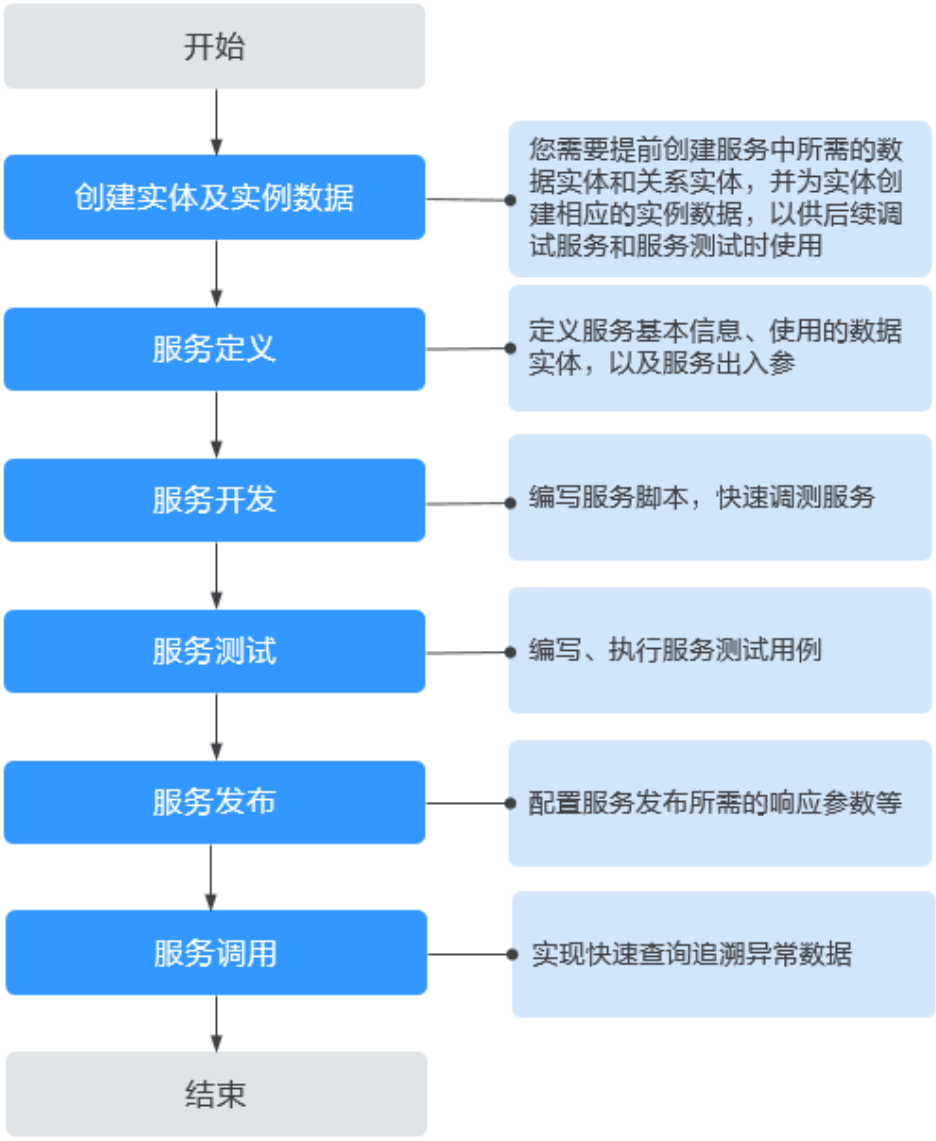
对比传统多个系统中分段追溯的痛点，数字主线引擎可以实现10W级条码数据查询10min内，100W级的条码数据查询1h内，从而快速、准确一键式追溯到问题器件DateCode、LotCode等批次信息和硬盘、内存厂家SN，以及装配的发货父项编码以及对应的合同、客户等相关数据。

图 11-1 对比优势



服务开发与发布流程

图 11-2 服务开发与发布流程



11.2 实施步骤

11.2.1 准备工作

在进行API编排前，已创建如下数据实体和关系实体，并基于这些实体创建了若干实例。

表 11-1 服务使用的数据实体

英文名称	中文名称	中文描述	英文描述
SupplierSN	供应商SN实例	供应商SN的实例信息。	SupplierSN
Part	Part主信息	称为Item或部件，指企业经营活动相关的原材料、外协件、在制品、半成品、备件、成品、费用、服务等，包括编码、描述、计量单位、项目模板、状态、厂家信息等属性。	Part
PurchaserSN	采购商SN实例	采购商SN的实例信息，含21条码、09条码等。	PurchaserSN
ReverseBackItems	单板维修及返还记录	单板维修及返还记录，含逆向退回和报废数据。	ReverseBackItems
ProductTallyShip	产品理货发货信息	产品理货发货信息，含条码理货、发货信息。	ProductTallyShip
Offering	Offering整合信息	Offering。	Offering
PartVersion	Part版本	Part版本。	PartVersion

表 11-2 服务使用的关系实体中源端、目标端数据实体对应的关联属性

数据实体	属性英文名称	属性中文名称	属性英文描述	属性中文描述	类型	是否唯一键	密级
PartVersion	PartNumber	Part编码	PartNumber	Part的唯一标识	文本	否	内部公开
PartVersion	OfferingId	Offering ID	OfferingId	Offering的ID，IT系统自动生成流水码	文本	否	内部公开
SupplierSN	ID	ID	ID	唯一编码	文本	是	内部公开
ProductTallyShip	BarCode	BarCode	BarCode	BarCode	文本	否	内部公开

数据实体	属性英文名称	属性中文名称	属性英文描述	属性中文描述	类型	是否唯一键	密级
PurchaserSN	PartNumber	Part编码	PartNumber	Part的唯一标识	文本	否	内部公开
PurchaserSN	ID	ID	ID	唯一编码	文本	是	内部公开
ReverseBackItems	Barcode	barcode	barcode	barcode	文本	否	内部公开
Part	BpartNumber	Part编码	BpartNumber	Part的唯一标识	文本	是	内部公开
Offering	ID	ID	ID	唯一编码	文本	是	内部公开

表 11-3 服务使用的关系实体

英文名称	中文名称	中文描述	英文描述	是否实体关系	源端数据实体	源端关联属性	目标端数据实体	目标端关联属性
PartVersionTOPartANDContain	Part版本与Part的关系	Part版本与Part的关系。	PartVersion And Part Relationship	否	Part Version	Part Number	Part	BpartNumber

英文名称	中文名称	中文描述	英文描述	是否实体关系	源端数据实体	源端关联属性	目标端数据实体	目标端关联属性
PartVersionTOOfferingANDContain	Part版本与Offering的关系	Part版本与Offering的关系。	PartVersion And Offering Relationship	否	PartVersion	OfferingId	Offering	ID
SupplierSNAndPurchaserSNRelationship	供应商SN实例与采购商SN实例的关系	供应商SN实例与采购商SN实例的关系。	SupplierSN And PurchaserSN Relationship	是	SupplierSN	ID	PurchaserSN	ID
PurchaserSNAndProductTallyShipRelationship	产品理货发货信息与采购商SN实例的关系	产品理货发货信息与采购商SN实例的关系。	ProductTallyShip And PurchaserSN Relationship	否	ProductTallyShip	BarCode	PurchaserSN	ID
PurchaserSNAndPartRelationship	采购商SN实例与Part的关系	采购商SN实例与Part的关系。	PurchaserSN And Part Relationship	否	PurchaserSN	PartNumber	Part	BpartNumber
ReverseBackItemstoPurchaserSN	单板维修及返还记录与采购商SN实例的关系	单板维修及返还记录与采购商SN实例的关系。	ReverseBackItems And PurchaserSN Relationship	否	ReverseBackItems	Barcode	PurchaserSN	ID

11.2.2 步骤 1：服务定义

根据方案概述中的示例场景，此处以编排一个“硬盘质量追溯_获取明细列表”的非纯脚本服务为例，指导您如何通过拖拉拽图形化的方式快速编排生成API。

操作步骤

步骤1 登录LinkX-F系统首页。

步骤2 选择进入“数据服务 > 聚合服务编排”页面。

步骤3 单击“创建”，弹出“创建聚合服务”界面。默认进入“服务定义”页的“基本信息”阶段。

步骤4 定义HarddiskTracing_GetDetailList聚合服务。

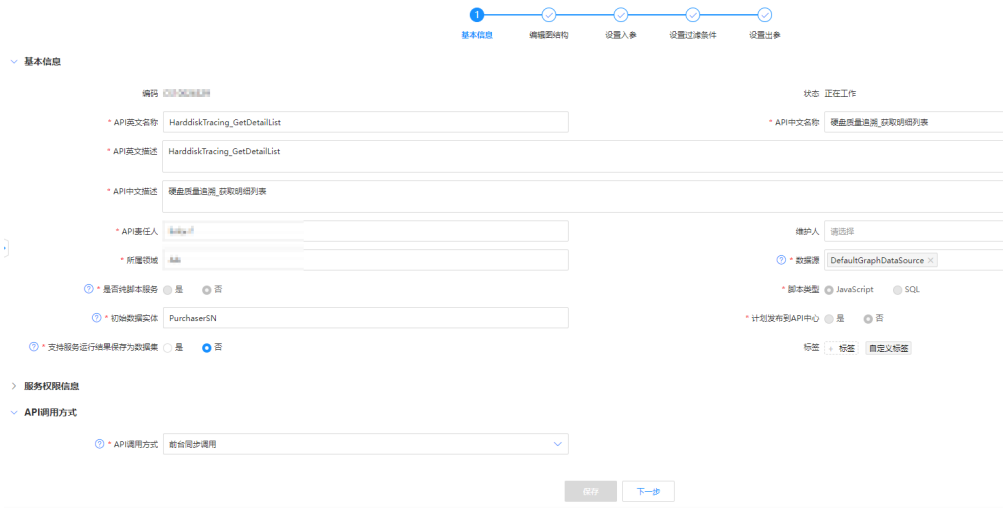
1. 填写服务定义基本信息，如表11-4所示：

表 11-4 HarddiskTracing_GetDetailList 服务定义基本信息

参数	说明
基本信息	
API英文名称	HarddiskTracing_GetDetailList
API中文名称	硬盘质量追溯_获取明细列表
API英文描述	HarddiskTracing_GetDetailList
API中文描述	硬盘质量追溯_获取明细列表。
API责任人	user001。 必填，输入服务的责任人。
维护人	user001。 非必填，输入服务的维护人。
所属领域	DefaultDomain。 来源于“系统管理 > 领域”。
数据源	Default Graph DataSource。 必填，该聚合服务实例数据所读取的图数据库，我们选择系统提供的默认图数据库。
是否纯脚本服务	选“否”：用户可参考全量数据实体构建聚合服务图结构。
脚本类型	当“是否纯脚本服务”选“否”时，自动写入且不支持修改。
初始数据实体	选取最合适的初始数据实体，选PurchaserSN。
计划发布到API中心	默认“否”不可修改。
支持服务运行结果保存为数据集	选“否”。
服务权限信息	
服务密级	“内部公开”。

参数	说明
API调用方式	
API调用方式	选前台同步调用：需考核接口响应时间。

图 11-3 服务基本信息



2. 填写完后单击“保存并下一步”，进入“编辑图结构”阶段。

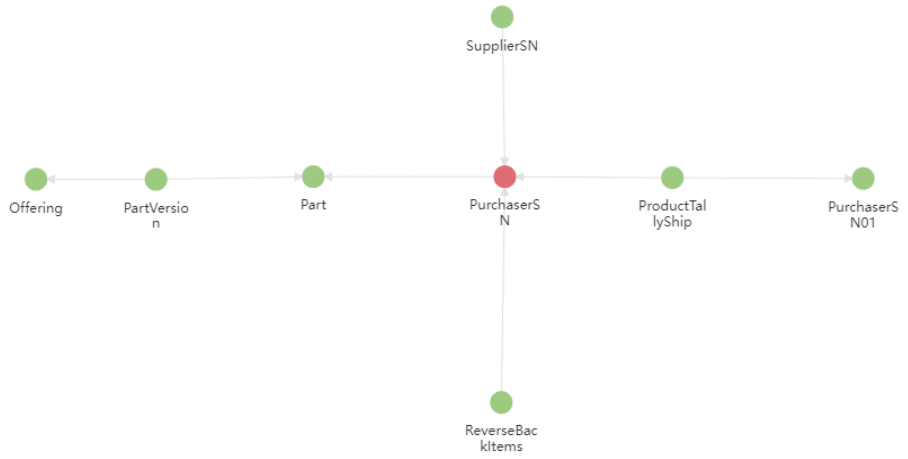
步骤5 编辑HarddiskTracing_GetDetailList服务图结构。

这一步可实现图形化编排数据模型的交互路径，系统会根据编排的图结构生成高效的查询脚本。

1. 在“服务定义”页签选中“编辑图结构”阶段。
2. 选中PurchaserSN模型节点后右键单击，在弹出的快捷菜单中选择“添加数据实体与关系”。
3. “添加关联数据实体与关系”窗口默认展示全部与PurchaserSN实体所关联的数据实体及关系。

   勾选SupplierSN（供应商SN实例）、ReverseBackItems（单板维修及返还记录）、Part（Part主信息）等数据实体。
4. 单击“确定”。重复以上操作添加更多数据实体与关系，最终画布展示所有已勾选的实体及关系组成的图谱。

图 11-4 图结构



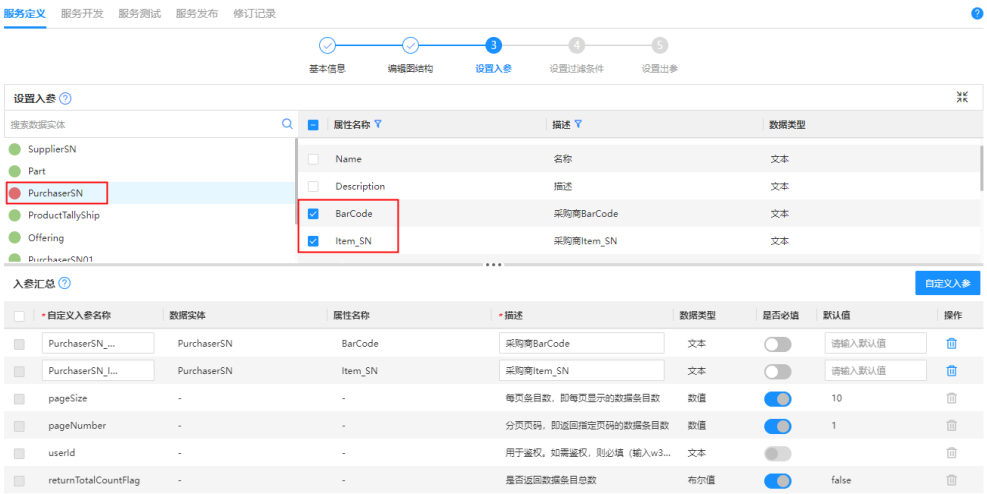
5. 单击“下一步”，进入“设置入参”阶段。

步骤6 设置HarddiskTracing_GetDetailList入参。

本示例是以获取明细列表为目的，以PurchaserSN实体作为入参实体比较合适。

1. 在 设置入参 处单击PurchaserSN实体，在右侧PurchaserSN实体属性列表中勾选入参属性BarCode、Item_SN。
   已选中的入参属性将在下方“入参汇总”处展示。

图 11-5 设置入参



2. 单击“下一步”，进入“设置过滤条件”阶段。

步骤7 设置入参过滤条件。

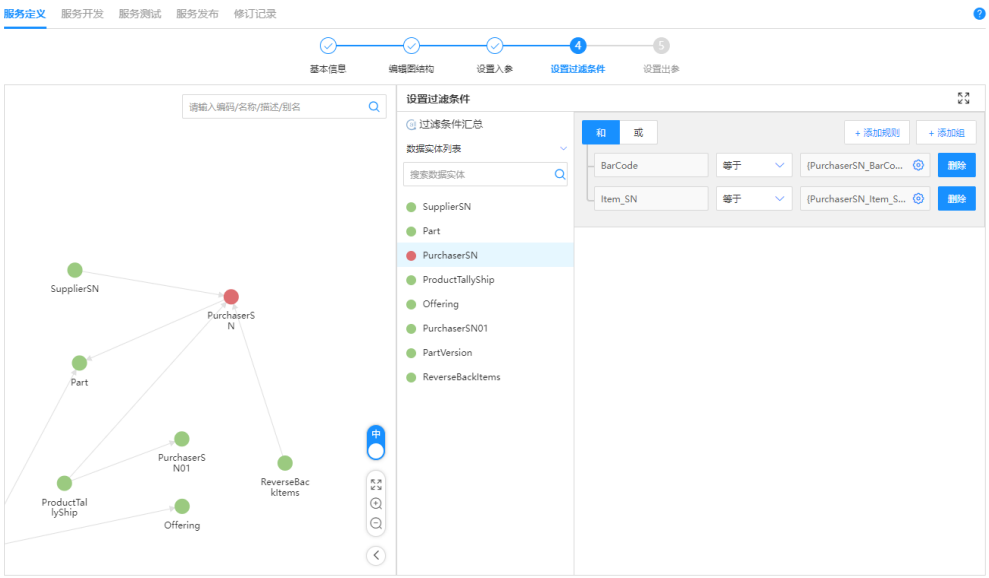
“过滤条件汇总”列表处展示的是“数据实体列表”中所有实体已设置的入参集合。

1. 展开“数据实体列表”，选择PurchaserSN实体，右侧显示PurchaserSN实体已设置的BarCode、Item_SN属性。
2. 为BarCode、Item_SN属性设置过滤条件，如表11-5所示：

表 11-5 PurchaserSN 实体属性过滤条件设置

参数	说明
各过滤条件之间的关系	选“和”。
过滤条件的属性	BarCode、Item_SN。
过滤条件匹配方式	等于。
过滤条件	PurchaserSN_BarCode、PurchaserSN_Item_SN。

图 11-6 设置过滤条件

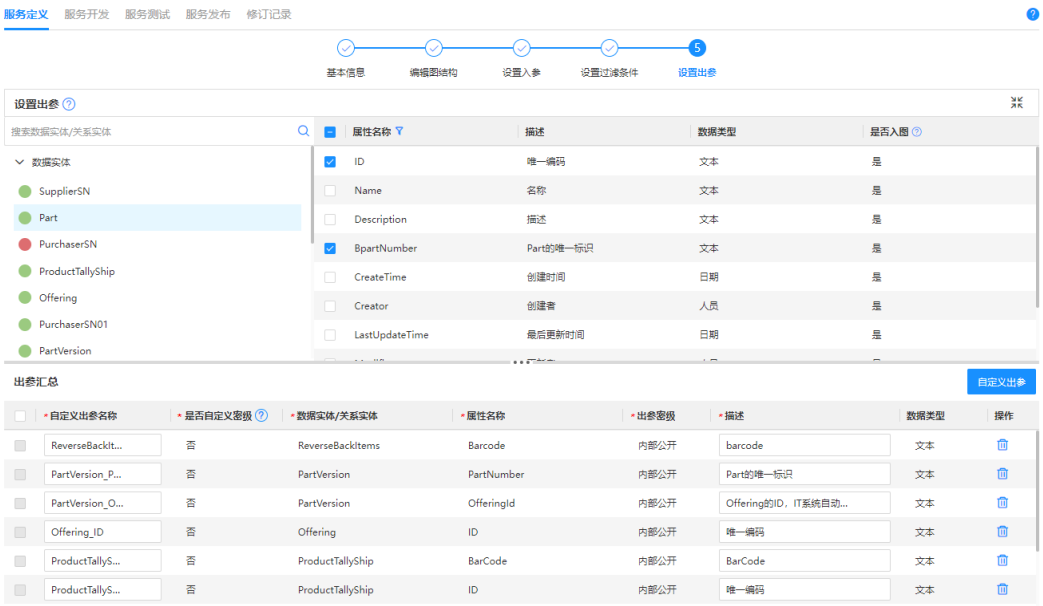


3. 单击“下一步”，进入“设置出参”阶段。

步骤8 设置出参。

在 设置出参 处依次单击出参数据实体，在右侧对应数据实体的属性列表中勾选出参的属性，出参属性在下侧表格汇总显示。

图 11-7 设置出参



步骤9 生成API。

在“服务定义”页签完成“设置入参”、“设置出参”等后，单击“生成API”同时生成相似服务。

----结束

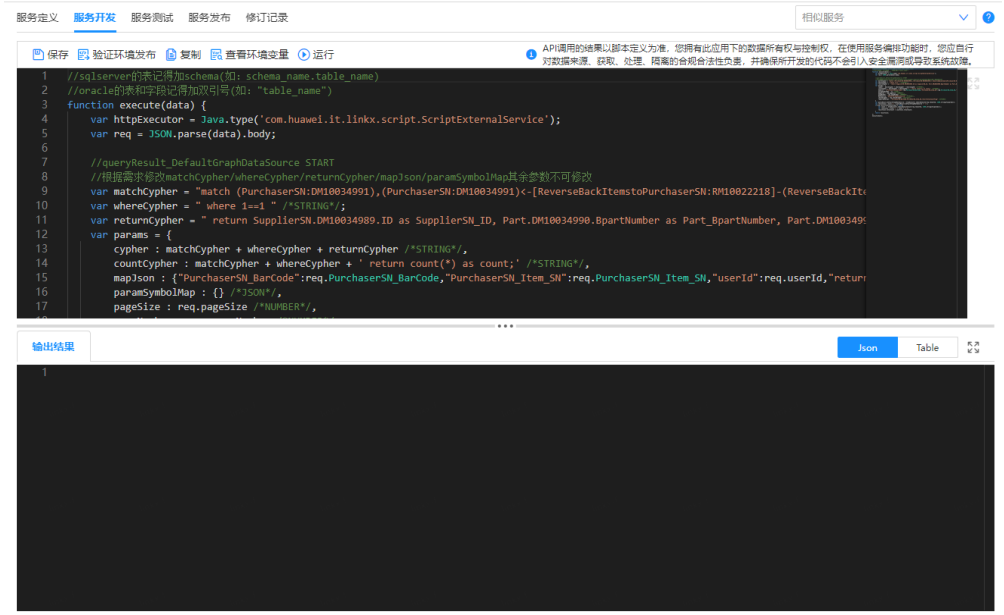
11.2.3 步骤 2：服务开发

以下过程是为了调试HarddiskTracing_GetDetailList服务。在进行服务调试前，请确保对应数据实体、关系实体的实例数据已入图。

操作步骤

步骤1 进入HarddiskTracing_GetDetailList服务详情页切换至“服务开发”页签，可以看到系统已根据**步骤1：服务定义**中编排的自动生成高效的服务查询脚本。

图 11-8 服务开发



步骤2 单击“运行”，弹出输入请求参数窗口。

步骤3 在“PurchaserSN_BarCode”、“PurchaserSN_Item_SN”输入框中输入值，单击“验证环境运行”或“生产环境运行”。

步骤4 （可选）修改出参名称。

如果对出参展示名称不满意，可在脚本编辑框中进行修改后再次运行。

----结束

11.2.4 步骤 3：服务测试

以下过程是为了给HarddiskTracing_GetDetailList服务新增测试用例并验证通过测试用例。

操作步骤

步骤1 进入HarddiskTracing_GetDetailList服务详情页切换至“服务测试”页签。

步骤2 在用例列表上方单击“新增用例”，弹出“新增测试用例”窗口。

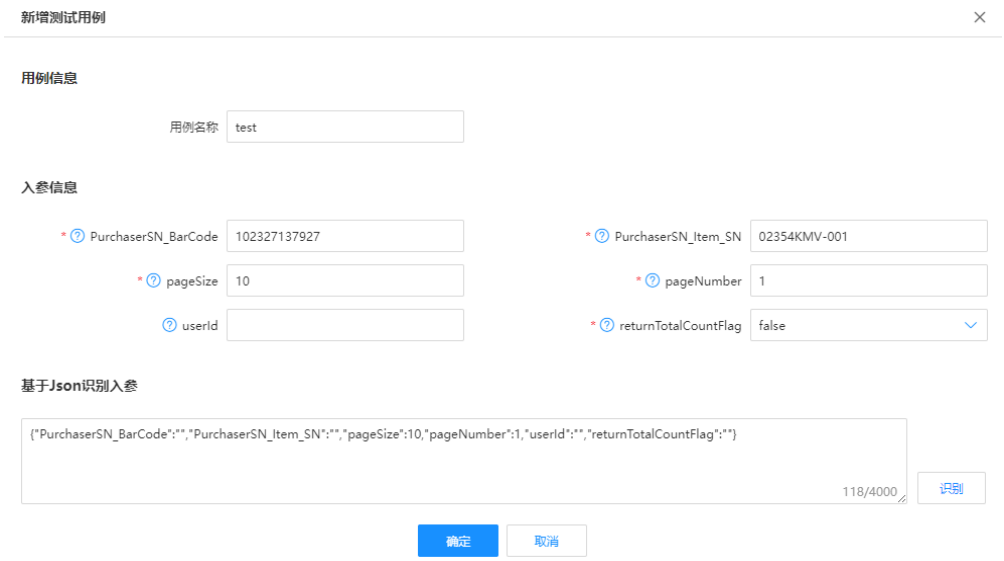
步骤3 填写测试用例信息，如表11-6所示：

表 11-6 测试用例信息

参数	说明
用例信息	
用例名称	test。
标签	支持用户自定义标签，可不填。
入参信息	

参数	说明
PurchaserSN_BarCode	示例：102327137927。
Item_SN	示例：02354KMV-001。
pageSize	10。 每页条目数，即每页显示的数据条目数。
pageNumber	1。 分页页码，即返回指定页码的数据条目数。
returnTotalCountFlag	false。 是否返回数据条目总数。
基于JSON识别入参	用户输入JSON脚本后，系统自动解析JSON脚本中的测试用例，并自动填充测试用例各字段，JSON参数名称区分大小写。

图 11-9 新增测试用例



步骤4 填写完后单击“确定”。

步骤5 在用例列表中，勾选想要执行的test测试用例，单击列表上方的“执行用例”。

用例执行完成之后，您可查看下方“执行结果”，查看该条用例的执行结果详情。

----结束

11.2.5 步骤 4：服务发布

以下过程是为了定义HarddiskTracing_GetDetailList服务的规范，如请求示例、正常响应示例、异常响应示例和错误示例码等。

操作步骤

步骤1 进入HarddiskTracing_GetDetailList服务详情页切换至“服务发布”页签。

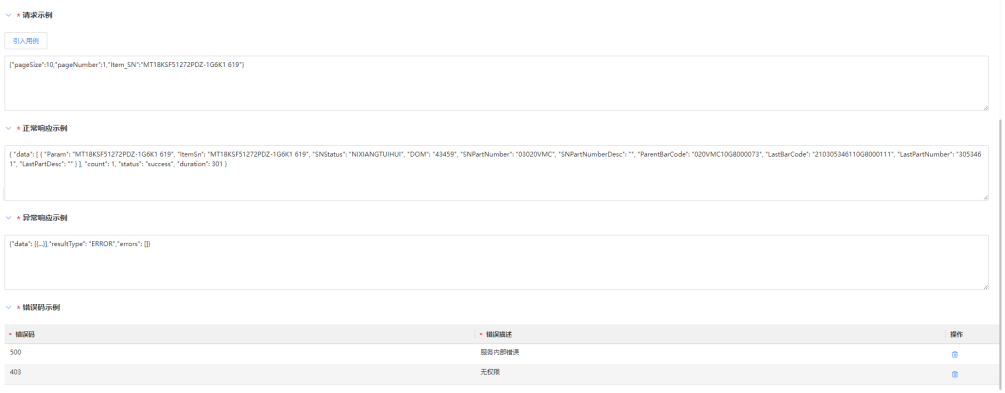
步骤2 配置服务发布信息。

1. 检查“基本信息”、入参、出参等信息无误。
   若发现缺少入参、出参，您可单击“编辑入参”、“编辑出参”快速进入“服务定义 > 入参设置”、“服务定义 > 出参设置”页面进行配置。
2. 输入响应参数说明，如表11-7所示：

表 11-7 响应参数说明

响应参数	示例内容
请求示例	{"pageSize":10,"pageNumber":1,"Item_SN":"MT18KSF51272PDZ-1G6K1 619"}
正常响应示例	{ "data": [{ "Param": "MT18KSF51272PDZ-1G6K1 619", "ItemSn": "MT18KSF51272PDZ-1G6K1 619", "SNStatus": "NIXIANGTUIHUI", "DOM": "43459", "SNPartNumber": "03020VMC", "SNPartNumberDesc": "", "ParentBarCode": "020VMC10G8000073", "LastBarCode": "2103053461l0G8000111", "LastPartNumber": "3053461", "LastPartDesc": "" }], "count": 1, "status": "success", "duration": 301 }
异常响应示例	{"data": [{...}],"resultType": "ERROR","errors": []}
错误码示例	– 403 {"message":"无权限"} – 500 {"message":"服务器内部异常"}

图 11-10 输入响应参数

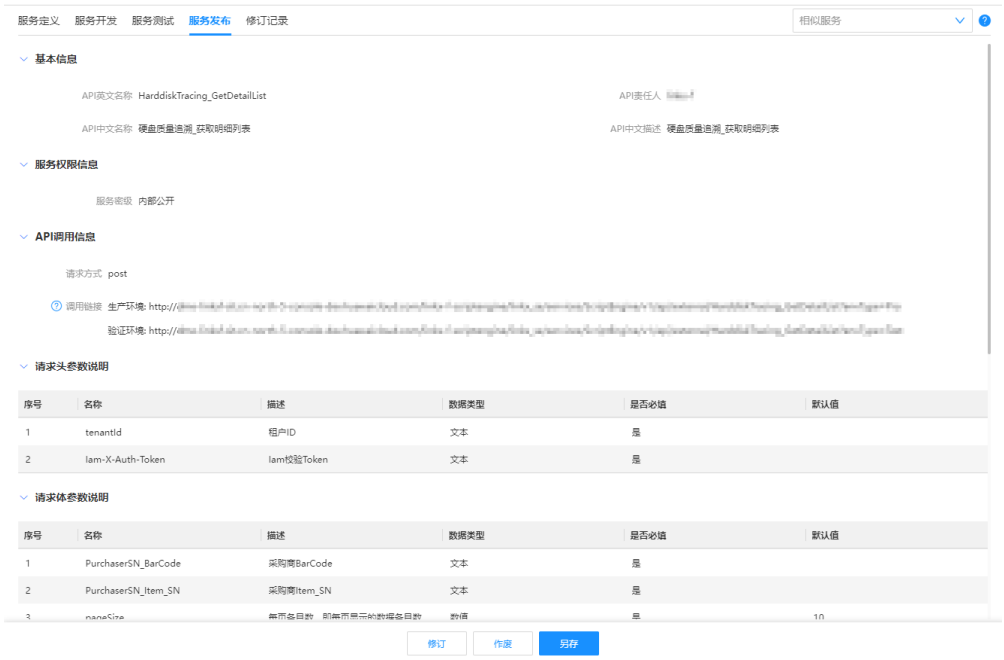


3. 单击“保存”。

步骤3 发布服务。

在“聚合服务”列表中单击HarddiskTracing_GetDetailList服务后的，弹出API调用方式提示框，单击“发布”。

图 11-11 服务发布信息



----结束

11.2.6 步骤 5：服务调用

在服务调用前，请确保已完成服务发布，具体操作请参见**步骤4：服务发布**。

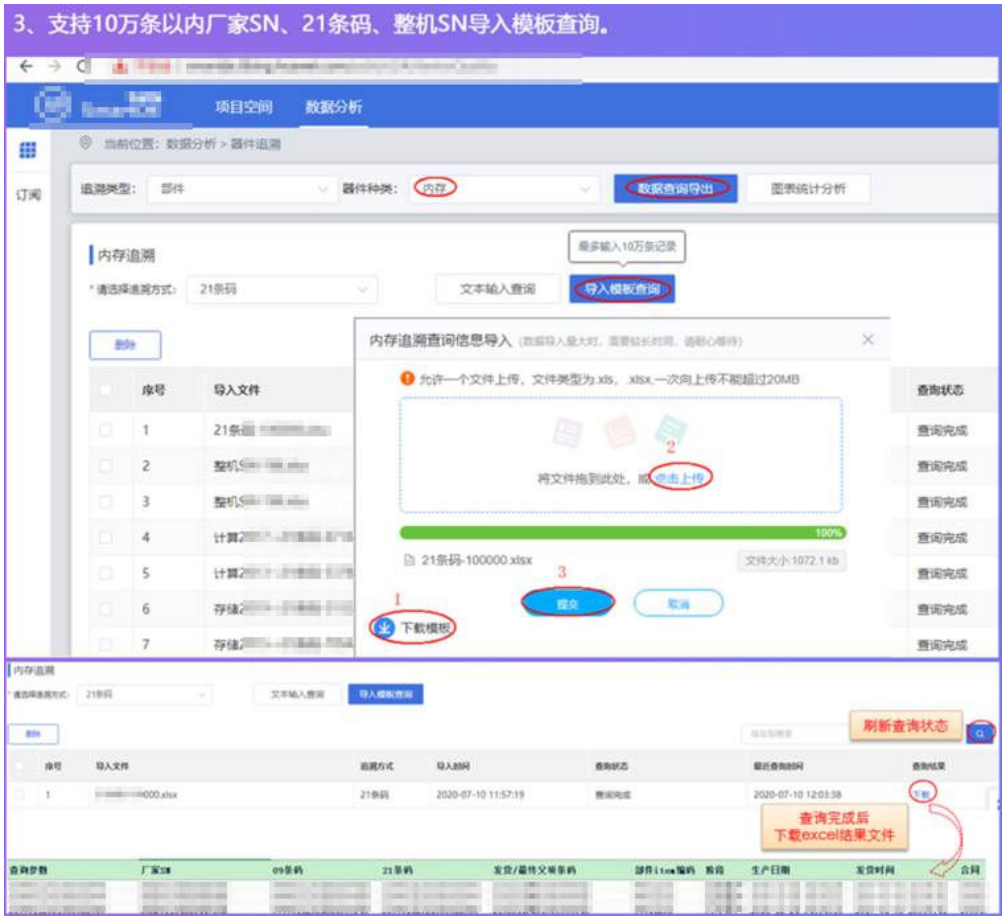
您可在“服务发布”页签中获取相应的API调用信息，包括生产环境API调用链接和验证环境API调用链接。其中验证环境API需在验证环境发布后才支持调用。

通过调用API，您可享受快速实现生产质量管理场景下跨系统异构数据中异常数据的排查和风险评估，并将结果集成到上层应用中使用，如BI系统、数据大屏应用。

图 11-12 器件追溯应用示例 1

图 11-13 器件追溯应用示例 2

图 11-14 器件追溯应用示例 3



12 重置应用运行态的租户数据

12.1 场景 1：清理某租户下的实例数据

操作场景

一般情况下，应用开发完成后需要测试人员对应用进行功能检查。而在测试期间，测试人员会在应用中编造出很多测试数据。如果您希望完成测试后可以一键清理这些无效数据，您可以通过调用“数据清理与重置”的“tenant-clean-task”接口，将“type”参数设置为“TENANT_RESET”，并指定“tenant_id”和“name”参数值，清理指定租户下的所有实例数据。

本章节以清理租户“TestUser”下的实例数据为例，如需了解更多数据清理与重置的API，请参见**全量数据服务**的“系统管理API > 数据清理与重置”。

⚠ 注意

- 清理某租户下的实例数据，将导致数据不可恢复，请谨慎操作。
- 如果您已将数据模型及其实例数据同步至LinkX-F系统，进行此操作后，将无法在LinkX-F系统进行数据模型及其实例数据的更新。
- 在清理指定租户下的实例数据期间，如果指定的租户是“-1”租户，该租户可以正常使用。如果指定的租户不是“-1”租户，该租户不能使用。

前提条件

已获取租户的唯一编码。

涉及接口

- 创建并执行清理数据任务

  URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task
- 查询任务执行结果

  URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**

- 重试清理数据任务

  URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**

其中，{endpoint}表示承载REST服务端点的服务器域名或IP地址，{appID}表示应用ID，{id}表示清理数据任务ID。

操作步骤

步骤1 创建并执行清理数据任务。

说明

- 如果存在响应参数“taskStatus”为“await_start”、“ongoing”或“fail_awai_retry”的任务，则不能对该任务中的租户创建新的清理数据任务。
- 创建并执行清理数据任务时，除了该租户在应用运行态中自定义的扩展模型、服务编排、搜索服务、联合索引数据和内置模型数据以外，其他数据均被删除。

- URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task
X-AUTH-TOKEN: ABCDEFJ....
{
   "params": {
      "tenant_id": "123456789",
      "name": "TestUser",
      "type": "TENANT_RESET"
   }
}
```

- 响应示例：

```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": "576798761137737728",
         "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
         "createTime": "2023-12-14T15:52:21.058+0800",
         "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
         "lastUpdateTime": "2023-12-14T15:52:21.058+0800",
         "rdmVersion": 1,
         "rdmDeleteFlag": 0,
         "className": null,
         "rdmExtensionType": "TenantCleanTask",
         "hibernateVersion": 1,
         "tenant": {
            "id": "-1",
            "clazz": "Tenant"
         },
         "name": "TestUser",
         "applicationTaskId": null,
         "cleanTaskTenantId": 12,
         "type": "TENANT_RESET",
         "taskStatus": "ONGOING",
         "retryTimes": 0,
         "fileStatus": "AWAIT_START",
         "ddsStatus": "AWAIT_START",
         "cssStatus": "AWAIT_START",
         "rdsStatus": "AWAIT_START",
         "metadataStatus": "AWAIT_START",
         "failMessage": null
```

```
    }
  ],
  "errors": []
}
```

步骤2 记录**响应**中“data”的“id”，例如“576798761137737728”。

步骤3 查询任务执行结果。

- URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
GET https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task/576798761137737728
X-AUTH-TOKEN: ABCDEFJ....
```

- 响应示例：

```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "576798761137737728",
      "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "createTime": "2023-12-14T15:53:30.420+0800",
      "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "lastUpdateTime": "2023-12-14T15:53:30.420+0800",
      "rdmVersion": 5,
      "rdmDeleteFlag": 0,
      "className": null,
      "rdmExtensionType": "TenantCleanTask",
      "hibernateVersion": 8,
      "tenant": {
        "id": "-1",
        "clazz": "Tenant"
      },
      "name": "TestUser",
      "applicationTaskId": null,
      "cleanTaskTenantId": 12,
      "type": "TENANT_RESET",
      "taskStatus": "SUCCESS",
      "retryTimes": 0,
      "fileStatus": "SUCCESS",
      "ddsStatus": "SUCCESS",
      "cssStatus": "SUCCESS",
      "rdsStatus": "SUCCESS",
      "metadataStatus": "SUCCESS",
      "failMessage": null
    }
  ],
  "errors": []
}
```

步骤4 （可选）手动重试清理数据任务。

说明

- 如果响应的“taskStatus”为“success”或“fail”，不允许调用重试接口。
- 如果响应的“taskStatus”为“fail_awai_retry”，则表示清理失败，正在等待重试。等待重试的任务仅支持重试3次，如果重试3次后仍清理失败，则任务结束。

  即，当响应的“retryTimes”＜3时，您可以直接调用重试清理数据任务接口进行手动重试，也可以等待系统触发重试机制自动重试（系统会在凌晨3点自动重试）。

- URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**

- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
PUT https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task/576798761137737728
X-AUTH-TOKEN: ABCDEFJ....
```

- 响应示例：

```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": "576798761137737728",
         "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
         "createTime": "2023-12-14T15:53:30.420+0800",
         "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
         "lastUpdateTime": "2023-12-14T15:53:30.420+0800",
         "rdmVersion": 5,
         "rdmDeleteFlag": 0,
         "className": null,
         "rdmExtensionType": "TenantCleanTask",
         "hibernateVersion": 8,
         "tenant": {
            "id": "-1",
            "clazz": "Tenant"
         },
         "name": "TestUser",
         "applicationTaskId": null,
         "cleanTaskTenantId": 12,
         "type": "TENANT_RESET",
         "taskStatus": "ONGOING",
         "retryTimes": 1,
         "fileStatus": "SUCCESS",
         "ddsStatus": "SUCCESS",
         "cssStatus": "SUCCESS",
         "rdsStatus": "FAIL",
         "metadataStatus": "AWAIT_START",
         "failMessage": "生命周期状态[Start]已存在"
      }
   ],
   "errors": []
}
```

----结束

12.2 场景 2：删除某租户及其所有数据

操作场景

在使用应用运行态时，如果您创建的某个逻辑租户后续不再使用，您可以通过调用“数据清理与重置”的“tenant-clean-task”接口，将“type”参数设置为“TENANT_DELETE”，并指定“tenant_id”和“name”参数值，删除指定租户及该租户下的所有实例数据。

⚠ 注意

- 删除租户时，会删除该租户下的所有数据，请谨慎操作。
- 如果您已将数据模型及其实例数据同步至LinkX-F系统，进行此操作后，将无法在LinkX-F系统进行数据模型及其实例数据的更新。
- 不支持删除唯一编码为“-1”的租户。
- 执行删除非“-1”的租户操作后，该租户不能再使用。

前提条件

已获取租户的唯一编码。

涉及接口

- 创建并执行清理数据任务

  URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task
- 查询任务执行结果

  URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 重试清理数据任务

  URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**

其中，{endpoint}表示承载REST服务端点的服务器域名或IP地址，{appID}表示应用ID，{id}表示清理数据任务ID。

操作步骤

步骤1 创建并执行清理数据任务。

说明

如果存在响应参数“taskStatus”为“await_start”、“ongoing”或“fail_awai_retry”的任务，则不能对该任务中的租户创建新的清理数据任务。

- URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task
X-AUTH-TOKEN: ABCDEFJ....
{
    "params": {
        "tenant_id": "123456789",
        "name": "TestUser",
        "type": "TENANT_DELETE"
    }
}
```

- 响应示例：

```
{
    "result": "SUCCESS",
```

```
  "data": [
    {
      "id": "576804608546967552",
      "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "createTime": "2023-12-14T16:15:35.189+0800",
      "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "lastUpdateTime": "2023-12-14T16:15:35.189+0800",
      "rdmVersion": 1,
      "rdmDeleteFlag": 0,
      "className": null,
      "rdmExtensionType": "TenantCleanTask",
      "hibernateVersion": 1,
      "tenant": {
        "id": "-1",
        "clazz": "Tenant"
      },
      "name": "TestUser",
      "applicationTaskId": null,
      "cleanTaskTenantId": 12,
      "type": "TENANT_DELETE",
      "taskStatus": "ONGOING",
      "retryTimes": 0,
      "fileStatus": "AWAIT_START",
      "ddsStatus": "AWAIT_START",
      "cssStatus": "AWAIT_START",
      "rdsStatus": "AWAIT_START",
      "metadataStatus": "AWAIT_START",
      "failMessage": null
    }
  ],
  "errors": []
}
```


步骤2 记录**响应**中“data”的“id”，例如“576804608546967552”。

步骤3 查询任务执行结果。

- URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

  ```
  GET https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task/576804608546967552
  X-AUTH-TOKEN: ABCDEFJ....
  ```

- 响应示例：


```
{
  "result": "SUCCESS",
  "data": [
    {
      "id": "576804608546967552",
      "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "createTime": "2023-12-14T16:15:35.189+0800",
      "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
      "lastUpdateTime": "2023-12-14T16:15:35.189+0800",
      "rdmVersion": 5,
      "rdmDeleteFlag": 0,
      "className": null,
      "rdmExtensionType": "TenantCleanTask",
      "hibernateVersion": 8,
      "tenant": {
        "id": "-1",
        "clazz": "Tenant"
      },
      "name": "TestUser",
      "applicationTaskId": null,
      "cleanTaskTenantId": 12,
      "type": "TENANT_DELETE",
```



```
            "taskStatus": "SUCCESS",
            "retryTimes": 0,
            "fileStatus": "SUCCESS",
            "ddsStatus": "SUCCESS",
            "cssStatus": "SUCCESS",
            "rdsStatus": "SUCCESS",
            "metadataStatus": "SUCCESS",
            "failMessage": null
        }
    ],
    "errors": []
}
```

步骤4 （可选）手动重试清理数据任务。

说明

- 如果响应的“taskStatus”为“success”或“fail”，不允许调用重试接口。
- 如果响应的“taskStatus”为“fail_awai_retry”，则表示清理失败，正在等待重试。系统会连续3天在凌晨3点自动重试，如果重试3次后仍清理失败，则任务结束。您也可以通过此步骤手动重试。
- 一个“taskStatus”为“fail_awai_retry”的任务仅支持重试3次。

- URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
PUT https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task/576804608546967552
X-AUTH-TOKEN: ABCDEFJ....
```

- 响应示例：

```
{
    "result": "SUCCESS",
    "data": [
        {
            "id": "576804608546967552",
            "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
            "createTime": "2023-12-14T16:15:35.189+0800",
            "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
            "lastUpdateTime": "2023-12-14T16:15:35.189+0800",
            "rdmVersion": 5,
            "rdmDeleteFlag": 0,
            "className": null,
            "rdmExtensionType": "TenantCleanTask",
            "hibernateVersion": 8,
            "tenant": {
                "id": "-1",
                "clazz": "Tenant"
            },
            "name": "TestUser",
            "applicationTaskId": null,
            "cleanTaskTenantId": 12,
            "type": "TENANT_DELETE",
            "taskStatus": "ONGOING",
            "retryTimes": 1,
            "fileStatus": "SUCCESS",
            "ddsStatus": "SUCCESS",
            "cssStatus": "SUCCESS",
            "rdsStatus": "SUCCESS",
            "metadataStatus": "SUCCESS",
            "failMessage": null
        }
    ],
```

```
    "errors": []
}
```

----结束

12.3 场景 3：重置某应用运行态所有数据

操作场景

当您应用中的多个逻辑租户均产生很多脏数据，您希望可以一键恢复至应用的初始状态，您可以通过调用“数据清理与重置”的“application-clean-task”接口，自定义“name”参数值删除该应用运行态下的所有数据，重置应用，释放空间。

⚠ 注意

- 重置应用数据会清除应用运行态上的所有数据，将导致数据不可恢复，请谨慎操作。
- 如果您已将数据模型及其实例数据同步至LinkX-F系统，进行此操作后，将无法在LinkX-F系统进行数据模型及其实例数据的更新。

涉及接口

- 创建并执行重置应用数据任务

  URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/application-clean-task
- 查询任务执行结果

  URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/application-clean-task/**{id}**
- 重试重置应用数据任务

  URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/application-clean-task/**{id}**

其中，{endpoint}表示承载REST服务端点的服务器域名或IP地址，{appID}表示应用ID，{id}表示重置应用数据任务ID。

操作步骤

步骤1 创建并执行重置应用数据任务。

- URI格式：POST http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/application-clean-task
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
POST https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/application-clean-task
X-AUTH-TOKEN: ABCDEFJ....
{
    "name": "ResetTestApp"
}
```

- 响应示例：

为篇幅起见，这里只展示部分内容。请求成功时，响应参数如下：


```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": 576800426431614976,
         "name": "ResetTestApp",
         "task_status": "AWAIT_START",
         "tenant_clean_tasks": [
            {
               "id": "576800426490335232",
               "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
               "createTime": "2023-12-14T15:58:58.109+0800",
               "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
               "lastUpdateTime": "2023-12-14T15:58:58.109+0800",
               "rdmVersion": 1,
               "rdmDeleteFlag": 0,
               "className": null,
               "rdmExtensionType": "TenantCleanTask",
               "hibernateVersion": 1,
               "tenant": {
                  "id": "-1",
                  "clazz": "Tenant"
               },
               "name": "ResetTestApp",
               "applicationTaskId": 576800426431614976,
               "cleanTaskTenantId": -1,
               "type": "TENANT_RESET",
               "taskStatus": "ONGOING",
               "retryTimes": 0,
               "fileStatus": "AWAIT_START",
               "ddsStatus": "AWAIT_START",
               "cssStatus": "AWAIT_START",
               "rdsStatus": "AWAIT_START",
               "metadataStatus": "AWAIT_START",
               "failMessage": null
            },
            {
               ......
            },
            ......
         ]
      }
   ],
   "errors": []
}
```


步骤2 记录**响应**中“data”的“id”，例如“576800426431614976”。

步骤3 查询任务执行结果。

- URI格式：GET http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

  GET https://**dme.cn-north-4.huaweicloud.com**/rdm_**fce01234567d41828cf3473b07fa7ae2**_app/services/rdm/basic/api/v1/tenant-clean-task/**576800426431614976**
  X-AUTH-TOKEN: ABCDEFJ....
- 响应示例：


```
{
   "result": "SUCCESS",
   "data": [
      {
         "id": 576800426431614976,
         "name": "ResetTestApp",
         "task_status": "SUCCESS",
```

```
        "tenant_clean_tasks": [
          {
            "id": "576800426490335232",
            "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
            "createTime": "2023-12-14T15:58:58.109+0800",
            "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
            "lastUpdateTime": "2023-12-14T15:58:58.109+0800",
            "rdmVersion": 5,
            "rdmDeleteFlag": 0,
            "className": null,
            "rdmExtensionType": "TenantCleanTask",
            "hibernateVersion": 8,
            "tenant": {
              "id": "-1",
              "clazz": "Tenant"
            },
            "name": "ResetTestApp",
            "applicationTaskId": 576800426431614976,
            "cleanTaskTenantId": -1,
            "type": "TENANT_RESET",
            "taskStatus": "SUCCESS",
            "retryTimes": 0,
            "fileStatus": "SUCCESS",
            "ddsStatus": "SUCCESS",
            "cssStatus": "SUCCESS",
            "rdsStatus": "SUCCESS",
            "metadataStatus": "SUCCESS",
            "failMessage": null
          },
          {
            ......
          },
          ......
        ]
      }
    ],
    "errors": []
}
```


步骤4 （可选）手动重试重置应用数据任务。

说明

- 如果响应的“taskStatus”为“success”或“fail”，不允许调用重试接口。
- 如果响应的“taskStatus”为“fail_awai_retry”，则表示重置失败，正在等待重试。系统会连续3天在凌晨3点自动重试，如果重试3次后仍重置失败，则任务结束。您也可以通过此步骤手动重试。
- 一个“taskStatus”为“fail_awai_retry”的任务仅支持重试3次。

- URI格式：PUT http://**{Endpoint}**/rdm_**{appID}**_app/services/rdm/basic/api/v1/tenant-clean-task/**{id}**
- 参数说明请参见**全量数据服务**的“系统管理API > 数据清理与重置”。
- 请求示例：

```
PUT https://dme.cn-north-4.huaweicloud.com/rdm_fce01234567d41828cf3473b07fa7ae2_app/services/rdm/basic/api/v1/tenant-clean-task/576800426431614976
X-AUTH-TOKEN: ABCDEFJ....
```

- 响应示例：


```
{
    "result": "SUCCESS",
    "data": [
        {
            "id": 576800426431614976,
            "name": "ResetTestApp",
            "task_status": "SUCCESS",
            "tenant_clean_tasks": [
                {
```



```
                "id": "576800426490335232",
                "creator": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
                "createTime": "2023-12-14T15:58:58.109+0800",
                "modifier": "Tester_HQ 362613cab50d4f42be5749d111111d4f",
                "lastUpdateTime": "2023-12-14T15:58:58.109+0800",
                "rdmVersion": 5,
                "rdmDeleteFlag": 0,
                "className": null,
                "rdmExtensionType": "TenantCleanTask",
                "hibernateVersion": 8,
                "tenant": {
                    "id": "-1",
                    "clazz": "Tenant"
                },
                "name": "ResetTestApp",
                "applicationTaskId": 576800426431614976,
                "cleanTaskTenantId": -1,
                "type": "TENANT_RESET",
                "taskStatus": "ONGOING",
                "retryTimes": 1,
                "fileStatus": "SUCCESS",
                "ddsStatus": "SUCCESS",
                "cssStatus": "SUCCESS",
                "rdsStatus": "SUCCESS",
                "metadataStatus": "SUCCESS",
                "failMessage": null
            },
            {
                ......
            },
            ......
        ]
    }
  ],
  "errors": []
}
```


----结束

13 修订记录

发布日期	修改说明
2024-06-24	第十三次正式发布。 修改**查询接口概述**
2024-02-23	第十二次正式发布。 修改**通过API方式分块上传文件**
2023-12-28	第十一次正式发布。 新增**基于图形化方式编排API**
2023-12-25	第十次正式发布。 新增 • **重置应用运行态的租户数据** • **树形结构实践** 修改**通过事务型任务API实现事务一致性**
2023-11-27	第九次正式发布。 修改 • **文件服务实践** • **通过反向建模将已有数据库物理表转为iDME模型**
2023-10-29	第八次正式发布。 新增**锁定基线对象**
2023-09-16	第七次正式发布。 新增**通过事务型任务API实现事务一致性**

发布日期	修改说明
2023-06-17	第六次正式发布。 修改 ● **实施步骤** ● **查询接口概述** ● **数据实体查询接口** ● **使用高代码服务编排自定义API** ● **通过xDM-F的多租户能力实现应用运行态数据的逻辑隔离** ● 通过文件服务功能管理工业文件
2023-05-20	第五次正式发布。 新增 ● 通过文件服务功能管理工业文件 ● **通过基线功能实现历史阶段产品状态的回溯**
2023-03-18	第四次正式发布。 新增**使用高代码服务编排自定义API**
2023-02-18	第三次正式发布。 新增**使用搜索服务定义搜索数据**
2023-01-15	第二次正式发布。 新增**xDM-F的查询相关接口**
2023-01-06	第一次正式发布。