图引擎服务

开发指南

文档版本 01

发布日期 2024-04-17

华为技术有限公司

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 简介

图引擎服务概述

图引擎服务（Graph Engine Service，简称GES），是针对以“关系”为基础的“图”结构数据，进行查询、分析的服务。广泛应用于社交关系分析、推荐、精准营销、舆情及社会化聆听、信息传播、防欺诈等具有丰富关系数据的场景。

开发指南概述

图引擎服务软件开发工具包（GES SDK，Graph Engine Service Software Development Kit）是对GES提供的REST API进行的封装，以简化用户的开发工作。用户直接调用GES SDK提供的接口函数即可实现使用GES服务业务能力的目的。

内容导航

GES开发指南将指导您如何安装和配置开发环境、如何通过调用GES SDK提供的接口函数进行二次开发。

章节	内容
简介	简要介绍本服务和开发指南的概念。
准备工作	介绍使用GES SDK前的环境配置和下载SDK。
工程导入	介绍导入工程的方法。
使用业务面SDK	介绍使用业务面GES SDK进行的常用操作。
使用管理面SDK	介绍使用管理面GES SDK进行的常用操作。
SDK和REST API的关系	介绍GES SDK与REST API的关系。

2 准备工作

准备环境

- 已从**Oracle官网**下载并安装JDK1.8或以上版本，配置好JAVA环境变量。
- 已从**Eclipse官网**下载并安装Eclipse IDE for Java Developers最新版本。
- 已在Eclipse中配置好JDK。

下载 SDK

进入图引擎服务管理控制台，在左侧导航栏选择“连接管理”，进入SDK下载页面。具体操作请参见**连接管理操作指导**。

3 工程导入

外部 jar 包打入本地 maven 仓库

1. 解压huaweicloud-ges-sdk-java-xxx.zip，进入解压后该目录下的maven-install目录中，执行ges-sdk-java-maven-install.bat文件或ges-sdk-java-maven-install.sh文件，将sdk-common-xxx.jar、ges-sdk-xxx.jar、graph-sdk-xxx.jar、cypher-jdbc-driver-xxx.jar以及java-sdk-core-xxx.jar安装到本地maven仓库。
2. 创建maven工程，在pom文件中导入依赖即可。

```
<dependency>
    <groupId>com.huawei.ges</groupId>
    <artifactId>ges-sdk</artifactId>
    <version>xxx</version>  //此处需要输入当前管理面sdk的版本号
</dependency>

<dependency>
    <groupId>com.huawei.ges.graph</groupId>
    <artifactId>graph-sdk</artifactId>
    <version>xxx</version> //此处需要输入当前业务面sdk的版本号
</dependency>
<!-- 使用cypher jdbc驱动时引入 -->
<dependency>
    <groupId>com.huawei.ges</groupId>
    <artifactId>cypher-jdbc-driver</artifactId>
    <version>xxx</version>  //此处需要输入cypher-jdbc-driver的版本号
</dependency>
```

本地导入 jar 包

新建工程，解压huaweicloud-ges-sdk-java-xxx.zip，进入huaweicloud-ges-sdk-java-xxx目录，将jars目录下的ges-sdk-xxx-jar-with-dependencies.jar、graph-sdk-xxx-jar-with-dependencies.jar导入工程或者将ges-sdk-xxx.jar、graph-sdk-xxx.jar、cypher-jdbc-driver-xxx.jar以及libs目录下所有包导入工程皆可。

4 使用业务面 SDK

4.1 初始化 GES 业务面客户端

使用GES SDK工具访问GES，需要用户初始化GES客户端。

- 通过内网访问时，endpoint为GES Console界面上的私网IP或者图详情查询API返回结果里面的“privateIp”字段；
- 通过公网访问时，endpoint为GES Console界面上的公网IP或者图详情查询API返回结果里面的“publicIp”字段。

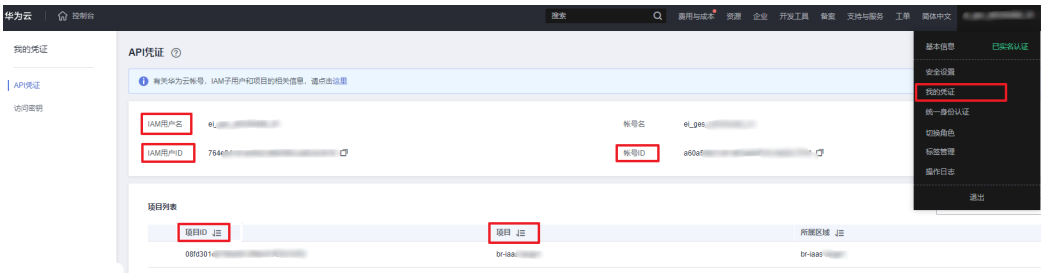
示例代码如下：

```
String endPoint = "endpoint";
String version = "v1.0";
String projectId = "project_id";
String graphName = "graph_name";
//注意：公网访问时，必须使用TOKEN认证方式，且须提供下面的dominName，userName，password
String regionName = "region_name";
String domainName = "domain_name";
String userName = "user_name";
String password = "user_password";
String userId = "userId";

GraphInfo graphInfo = new GraphInfo(endPoint, version, projectId, userId, graphName, regionName,
domainName, userName, password);
//生成graph client并初始化，初始化时默认生成了与图名称同名的对象，供后续gremlin命令直接使用。
graphInfo.setIamEndpoint("iam_endpoint");
GraphClient graphClient = new GraphClient(AuthenticationMode.TOKEN,graphInfo);
```


说明

- graphName、私网IP及公网IP可在GES Console上的“图管理”页面查看，具体请参考**图管理简介**。
- regionName、domainName、userName、userId、projectId的获取方法：在GES Console界面上鼠标移动至右上方的用户名，在下拉列表中选择“我的凭证”。进入个人设置界面，项目即为regionName，domainName为账号名，userName为IAM用户名，userId为IAM用户ID，项目ID即为projectId。



4.2 点过滤查询

GES提供点过滤查询接口，您可以使用该接口查询满足过滤条件的顶点集合。示例代码如下：

```
 public static void vertexQuery(GraphClient graphClient) throws ApiException
{
//点过滤查询请求
VertexFilterQueryReq req = new VertexFilterQueryReq();
 List<String> labels = new ArrayList<>();
 labels.add("movie");
//按label过滤点
req.setLabels(labels);
//设置offset和limit
req.setOffset(0);
 req.setLimit(5);
//发送请求，拿到异步API结果
AsyncAPIResp asyncAPIResp = graphClient.vertexQuery(req);
//点过滤查询为异步API，取出jobId
String jobId = asyncAPIResp.getJobId();
//点查询Job请求
VertexQueryJobReq req1 = new VertexQueryJobReq();
//设置jobId
req1.setJobId(jobId);
//获取job查询结果
JobResp<QueryData<VertexQueryResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```

4.3 边过滤查询

GES提供校验边过滤查询接口，您可以使用该接口查询满足过滤条件边的集合。示例代码如下：

```
  public static void edgeQuery(GraphClient graphClient) throws ApiException
{
//设置边过滤查询
EdgeFilterQueryReq req = new EdgeFilterQueryReq();
//按label过滤边
List<String> lables = new ArrayList<>();
 lables.add("rate");
 req.setLabels(lables);
```



```
//设置offset和limit，限制结果个数
 req.setOffset(0);
 req.setLimit(5);
 AsyncAPIResp resp = graphClient.edgeQuery(req);
 String jobId = resp.getJobId();
 EdgeQueryJobReq req1 = new EdgeQueryJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.4 查询节点详情

您可以使用GES提供的接口查询节点详情，示例代码如下：

```
public static void getVertices(GraphClient graphClient) throws ApiException
{
 VertexDetailReq req = new VertexDetailReq();
 req.setVertexIds("46");//此处需要用户替换为实际的点ID
 VertexDetailResult vertexDetailResult = graphClient.getVertices(req);
 System.out.print(vertexDetailResult);
}
```

4.5 查询边详情

您可以使用GES提供的接口查询边详情，示例代码如下：

```
 public static void getEdge(GraphClient graphClient) throws ApiException
{
 EdgeDetailReq req = new EdgeDetailReq();
 req.setSource("46");//此处需要用户替换为实际的点ID
 req.setTarget("38");//此处需要用户替换为实际的点ID
 req.setIndex(0);
 EdgeDetailResult edgeDetailResult = graphClient.getEdge(req);
 System.out.print(edgeDetailResult);
}
```

4.6 查询图元数据详情

您可以使用GES提供的接口查询元数据详情。示例代码如下：

```
 public static void getSchema(GraphClient graphClient) throws ApiException
{   Map result = graphClient.getGraphSchema();
    //用MapUtils工具类格式化输出
    System.out.print(MapUtils.map2json(result));
}
```

4.7 查询图概要信息

您可以使用GES提供的接口查询图概要信息。示例代码如下：

```
  public static void getSummary(GraphClient graphClient) throws ApiException
{
    Map result = graphClient.getGraphSummary();
    //用MapUtils工具类格式化输出
    System.out.print(MapUtils.map2json(result));
}
```


4.8 执行 Gremlin 查询

您可以使用GES提供的接口执行Gremlin查询。示例代码如下：

说明

在graph client初始化时，已经在系统中生成与graph name同名的对象，用户直接用graph name进行查询即可。

```
 public static void executeGremlinQuery(GraphClient graphClient,String graphName) throws ApiException
{   //gremlin命令，graph client初始化时默认生成了与图名称同名的对象，如下直接使用即可
    String gremlinCommand = graphName + ".V(\"145\")";
    GremlinQueryReq req = new GremlinQueryReq();
    req.setCommand(gremlinCommand);
    Map<String, Object> stringObjectMap = graphClient.gremlinQuery(req);
    System.out.print(MapUtils.map2json(stringObjectMap));
}
```

4.9 执行算法

4.9.1 Page_rank 算法

您可以使用GES提供的接口执行page_rank算法。示例代码如下

```
 public void static executeAlgorithm(GraphClient graphClient) throws ApiException
{
//设置page_rank算法参数
PageRankParameters parameters = new PageRankParameters();
 parameters.setMaxIterations(1000);
//算法请求
AlgorithmReq req = new AlgorithmReq();
req.setAlgorithmName(AlgorithmNames.PAGE_RANK);
 req.setParameters(parameters);
//执行算法
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//获取JobId
String jobId = asyncAPIResp.getJobId();
 PageRankJobReq req1 = new PageRankJobReq();
 req1.setJobId(jobId);
//根据JobId查询算法执行结果
JobResp<QueryData<PageRankResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```

4.9.2 Personal_rank 算法

您可以使用GES提供的接口执行personal_rank算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 PersonalRankParameters parameters = new PersonalRankParameters();
 parameters.setMaxIterations(1000);
 parameters.setSource("46");
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.PERSONAL_RANK);
 req.setParameters(parameters);
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 PersonalRankJobReq req1 = new PersonalRankJobReq();
 req1.setJobId(jobId);
 JobResp<QueryData<PersonalRankResult>> resp = graphClient.queryJobStatus(req1);
```



```
 System.out.println(resp);
}
```

4.9.3 k_hop 算法

您可以使用GES提供的接口执行k_hop算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 KHopParameters parameters = new KHopParameters();
 parameters.setSource("46");
 parameters.setK(2);
 parameters.setMode(EdgeDirection.OUT);
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.K_HOP);
 req.setParameters(parameters);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();;
 KHopJobReq req1 = new KHopJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.4 K_core 算法

您可以使用GES提供的接口执行k_core算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 KCoreParameters parameters = new KCoreParameters();
 parameters.setK(3);

 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.K_CORE);
 req.setParameters(parameters);


 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 KCoreJobReq req1 = new KCoreJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.5 Shortest_path 算法

您可以使用GES提供的接口执行shortest_path算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 ShortestPathParameters parameters = new ShortestPathParameters();
 parameters.setSource("30");
 parameters.setTarget("46");

 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.SHORTEST_PATH);
 req.setParameters(parameters);


 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 ShortestPathJobReq req1 = new ShortestPathJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```


4.9.6 All_shortest_paths 算法

您可以使用GES提供的接口执行all_shortest_paths算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
AllShortestPathParameters parameters = new AllShortestPathParameters();
 parameters.setSource("46");
 parameters.setTarget("35");
//算法请求
AlgorithmReq req = new AlgorithmReq();
//设置算法名称
req.setAlgorithmName(AlgorithmNames.ALL_SHORTEST_PATHS);
 req.setParameters(parameters);
//执行算法
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//获取JobId
String jobId = asyncAPIResp.getJobId();
 AllShortestPathJobReq req1 = new AllShortestPathJobReq();
 req1.setJobId(jobId);
//根据JobId查询算法执行结果
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.7 filtered_shortest_path 算法

您可以使用GES提供的接口执行filtered_shortest_path算法。示例代码如下：

```
public void filteredShortestPath(GraphClient graphClient) throws ApiException {
    FilteredShortestPathParameters filteredShortestPathParameters = new FilteredShortestPathParameters();
    filteredShortestPathParameters.setSource("Vivian");
    filteredShortestPathParameters.setTarget("Mercedes");
    filteredShortestPathParameters.setDirected(false);

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.FILTERED_SHORTEST_PATH);  // 算法名
    algorithmReq.setParameters(filteredShortestPathParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.8 SSSP 算法

您可以使用GES提供的接口执行SSSP算法。示例代码如下：

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
 {
 SSSPParameters parameters = new SSSPParameters();
 parameters.setSource("46");
 //算法请求
 AlgorithmReq req = new AlgorithmReq();
 //设置算法名称
 req.setAlgorithmName(AlgorithmNames.SSSP);
 req.setParameters(parameters);
 //执行算法
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 //获取JobId
 String jobId = asyncAPIResp.getJobId();
 SSSPJobReq req1 = new SSSPJobReq();
 req1.setJobId(jobId);
```

```
//根据JobId查询算法执行结果
System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.9 shortest_path_of_vertex_sets 算法

您可以使用GES提供的接口执行shortest_path_of_vertex_sets算法。示例代码如下：

```
public void shortestPathsOfVertexSets(GraphClient graphClient) throws ApiException {
    ShortestPathsOfVertexSetsParameters shortestPathsOfVertexSetsParameters = new
ShortestPathsOfVertexSetsParameters();
    shortestPathsOfVertexSetsParameters.setSources("Vivian,Mercedes");
    shortestPathsOfVertexSetsParameters.setTargets("Katherine");
    shortestPathsOfVertexSetsParameters.setDirected(false);

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.SHORTEST_PATH_OF_VERTEX_SETS);  // 算法名
    algorithmReq.setParameters(shortestPathsOfVertexSetsParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.10 n_paths 算法

您可以使用GES提供的接口执行n_paths算法。示例代码如下：

```
public void nPaths(GraphClient graphClient) throws ApiException {
    NPathsParameters nPathsParameters = new NPathsParameters();
    nPathsParameters.setSource("Vivian");
    nPathsParameters.setTarget("Katherine");
    nPathsParameters.setDirected(false);
    nPathsParameters.setN(10);
    nPathsParameters.setK(5);

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.N_PATHS);  // 算法名
    algorithmReq.setParameters(nPathsParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.11 Closeness 算法

您可以使用GES提供的接口执行closeness算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 ClosenessCentralityParameters parameters = new ClosenessCentralityParameters();
 parameters.setSource("46");
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.CLOSENESS);
 req.setParameters(parameters);
```



```
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 ClosenessJobReq req1 = new ClosenessJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.12 Label_propagation 算法

您可以使用GES提供的接口执行label_propagation算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.LABEL_PROPAGATION);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 LabelPropagationJobReq req1 = new LabelPropagationJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.13 Louvain 算法

您可以使用GES提供的接口执行Louvain算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.LOUVAIN);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 LouvainJobReq req1 = new LouvainJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.14 Link_prediction 算法

您可以使用GES提供的接口执行link_prediction算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 LinkPredictionParameters parameters = new LinkPredictionParameters();
 parameters.setSource("46");
 parameters.setTarget("38");
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.LINK_PREDICTION);
 req.setParameters(parameters);
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 LinkPredictionJobReq req1 = new LinkPredictionJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.15 Node2vec 算法

您可以使用GES提供的接口执行node2vec算法。示例代码如下：

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
 {
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.NODE2VEC);
```



```
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
String jobId = asyncAPIResp.getJobId();
Node2VecReq req1 = new Node2VecReq();
req1.setJobId(jobId);
JobResp<QueryData<Node2VecResult>> resp = graphClient.queryJobStatus(req1);
while (!resp.getStatus().equals("success"))
{
    resp = graphClient.queryJobStatus(req1);
    Thread.sleep(2000);
}
System.out.println(resp)
}
```

4.9.16 Realtime_recommendation 算法

您可以使用GES提供的接口执行realtime_recommendation算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
RealtimeRecParameters parameters = new RealtimeRecParameters();
 parameters.setSources("34,37,34");
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.REALTIME_RECOMMENDATION);
 req.setParameters(parameters);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 RealtimeRecommendationJobReq req1 = new RealtimeRecommendationJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.17 Common_neighbors 算法

您可以使用GES提供的接口执行common_neighbors算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
CommonNeighborsParameters parameters = new CommonNeighborsParameters();
parameters.setSource("46");
parameters.setTarget("38");
AlgorithmReq req = new AlgorithmReq();
req.setAlgorithmName(AlgorithmNames.COMMON_NEIGHBORS);
req.setParameters(parameters);
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
String jobId = asyncAPIResp.getJobId();
CommonNeighborsJobReq req1 = new CommonNeighborsJobReq();
req1.setJobId(jobId);
System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.18 Connect_component 算法

您可以使用GES提供的接口执行connect_component算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.CONNECTED_COMPONENT);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 ConnectComponentJobReq req1 = new ConnectComponentJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.19 Degree_correlation 算法

您可以使用GES提供的接口执行degree_correlation算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 AlgorithmReq req = new AlgorithmReq();
 req.setAlgorithmName(AlgorithmNames.DEGREE_CORRELATION);
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 DegreeCorrelationJobReq req1 = new DegreeCorrelationJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.20 Triangle_count 算法

您可以使用GES提供的接口执行triangle_count算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
//算法请求
AlgorithmReq req = new AlgorithmReq();
req.setAlgorithmName(AlgorithmNames.TRIANGLE_COUNT);
//执行算法
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//获取JobId
String jobId = asyncAPIResp.getJobId();
 TriangleCountJobReq req1 = new TriangleCountJobReq();
 req1.setJobId(jobId);
//根据JobId查询算法执行结果
JobResp<QueryData<TriangleCountResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```

4.9.21 Cluster_coefficient 算法

您可以使用GES提供的接口执行cluster_coefficient算法。示例代码如下

```
public static void executeAlgorithm(GraphClient graphClient) throws ApiException
{
 AlgorithmReq req = new AlgorithmReq();

 req.setAlgorithmName(AlgorithmNames.CLUSTER_COEFFICIENT);

 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
 String jobId = asyncAPIResp.getJobId();
 ClusterCoefficientJobReq req1 = new ClusterCoefficientJobReq();
 req1.setJobId(jobId);
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.22 Filtered_circle_detection

您可以使用GES提供的接口执行filtered_circle_detection算法。示例代码如下：

```
public void filteredCircleDetection(GraphClient graphClient) throws ApiException {
    FilteredCircleDetectionParameters filteredCircleDetectionParameters = new
FilteredCircleDetectionParameters();
    filteredCircleDetectionParameters.setSource("Vivian");
    filteredCircleDetectionParameters.setN(100);

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.FILTERED_CIRCLE_DELECTION);  // 算法名
    algorithmReq.setParameters(filteredCircleDetectionParameters);  // 算法参数
    algorithmReq.addFilter("out", FilterQueryType.EDGE_FILTER, "label_name","labelName", "=", "transfer",
```



```
5);

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.23 subgraph_matching 算法

您可以使用GES提供的接口执行subgraph_matching算法。示例代码如下：

```
public void subgraphMatching(GraphClient graphClient) throws ApiException {
    SubgraphMatchingParameters subgraphMatchingParameters = new SubgraphMatchingParameters();
    subgraphMatchingParameters.setEdges("");
    subgraphMatchingParameters.setVertices("");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.SUBGRAPH_MATCHING);  // 算法名
    algorithmReq.setParameters(subgraphMatchingParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.24 Filtered All Pairs Shortest paths 算法

您可以使用GES提供的接口执行Filtered All Pairs Shortest paths算法。示例代码如下：

```
public void filteredAllPairsShortestPaths(GraphClient graphClient) throws ApiException {
    FilteredAllPairsShortestPathsParameters filteredAllPairsShortestPathsParameters = new
FilteredAllPairsShortestPathsParameters();
    filteredAllPairsShortestPathsParameters.setSources("Vivian");
    filteredAllPairsShortestPathsParameters.setTargets("Lethal Weapon");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.FILTERED_ALL_PAIRS_SHORTEST_PATHS);  // 算法名
    algorithmReq.setParameters(filteredAllPairsShortestPathsParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.25 filtered_all_shortest_paths 算法

您可以使用GES提供的接口执行filtered_all_shortest_paths算法。示例代码如下：

```
public void filteredAllShortestPaths(GraphClient graphClient) throws ApiException {
    FilteredAllShortestPathsParameters filteredAllShortestPathsParameters = new
FilteredAllShortestPathsParameters();
```



```
    filteredAllShortestPathsParameters.setSource("Vivian");
    filteredAllShortestPathsParameters.setTarget("Lethal Weapon");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.FILTERED_ALL_SHORTEST_PATHS); // 算法名
    algorithmReq.setParameters(filteredAllShortestPathsParameters); // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.26 topicrank 算法

您可以使用GES提供的接口执行topicrank算法。示例代码如下：

```
public void topicrank(GraphClient graphClient) throws ApiException {
    TopicrankParameters topicrankParameters = new TopicrankParameters();
    topicrankParameters.setSources("Vivian");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.TOPICRANK); // 算法名
    algorithmReq.setParameters(topicrankParameters); // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.27 filtered_n_paths 算法

您可以使用GES提供的接口执行filtered_n_paths算法。示例代码如下：

```
public void filteredNPaths(GraphClient graphClient) throws ApiException {
    FilteredNPathsParameters filteredNPathsParameters = new FilteredNPathsParameters();
    filteredNPathsParameters.setSource("Vivian");
    filteredNPathsParameters.setTarget("Lethal Weapon");
    filteredNPathsParameters.setK(2);
    filteredNPathsParameters.setN(1);

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.FILTERED_N_PATHS); // 算法名
    algorithmReq.setParameters(filteredNPathsParameters); // 算法参数
    algorithmReq.addFilter("out", FilterQueryType.EDGE_FILTER, "label_name","labelName", "=", "transfer",
5);

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```


4.9.28 common_neighbors_of_vertex_sets 算法

您可以使用GES提供的接口执行common_neighbors_of_vertex_sets算法。示例代码如下：

```
public void commonNeighborsOfVertexSets(GraphClient graphClient) throws ApiException {
    CommonNeighborsOfVertexSetsParameters commonNeighborsOfVertexSetsParameters = new
CommonNeighborsOfVertexSetsParameters();
    commonNeighborsOfVertexSetsParameters.setSources("Vivian");
    commonNeighborsOfVertexSetsParameters.setTargets("Katherine");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.COMMON_NEIGHBORS_OF_VERTEX_SETS);  // 算法名
    algorithmReq.setParameters(commonNeighborsOfVertexSetsParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.29 all_shortest_paths_of_vertex_sets 算法

您可以使用GES提供的接口执行all_shortest_paths_of_vertex_sets算法。示例代码如下：

```
public void allShortestPathsOfVertexSets(GraphClient graphClient) throws ApiException {
    AllShortestPathsOfVertexSetsParameters allShortestPathsOfVertexSetsParameters = new
AllShortestPathsOfVertexSetsParameters();
    allShortestPathsOfVertexSetsParameters.setSources("Vivian");
    allShortestPathsOfVertexSetsParameters.setTargets("Katherine");

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.ALL_SHORTEST_PATH_OF_VERTEX_SETS);  // 算法名
    algorithmReq.setParameters(allShortestPathsOfVertexSetsParameters);  // 算法参数

    // 根据输入参数执行指定算法
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // 根据算法执行任务id，查询任务执行状态
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.10 查询 Job 状态

您可以使用GES提供的接口查询Job状态。示例代码如下：

```
 public static void queryAsyncTask(GraphClient graphClient,String jobId) throws ApiException
{
VertexQueryJobReq req = new VertexQueryJobReq();
//设置分页查询参数，offset默认为0，limit默认为100000
req.setOffset(0);
req.setLimit(100);
req.setJobId(jobId);
JobResp<QueryData<VertexQueryResult>> resp = graphClient.queryJobStatus(req);
System.out.println(resp);
}
```


说明

对过滤查询、边过滤查询、执行算法等异步API，命令下发后，会返回jobId，可以通过jobId查询任务的执行状态。

4.11 取消 Job

您可以使用GES提供的接口取消Job。示例代码如下：

```
public static void stopAsyncJob(GraphClient graphClient,String jobId) throws ApiException
{
    Map result = graphClient.deleteJob(jobId);
    //用MapUtils工具类格式化输出
    System.out.print(MapUtils.map2json(result));
}
```

说明

过滤查询、边过滤查询、执行算法等异步API，命令下发后，会返回jobId，可以通过jobId终止该任务。

4.12 添加点

您可以使用GES提供的接口添加点。示例代码如下：

```
 public static void addVertex(GraphClient graphClient) throws ApiException
{
    Map properties = new HashMap();
    List gender = new ArrayList();
    gender.add("F");
    properties.put("Gender", gender);
    List occupation = new ArrayList();
    occupation.add("artist");
    properties.put("Occupation", occupation);
    List age = new ArrayList();
    age.add("under 18");
    properties.put("Age", age);
    List zipCode = new ArrayList();
    zipCode.add("98133");
    properties.put("Zip-code", zipCode);
    AddVertexReq req = new AddVertexReq();
    //设置vertex Id
    req.setVertexId("Lily");
    //设置label
    req.setLabel("user");
    //设置属性
    req.setProperties(properties);
    graphClient.addVertex(req);

}
```

4.13 删除点

您可以使用GES提供的接口删除点。示例代码如下：

```
public static void deleteVertex(GraphClient graphClient) throws ApiException
{
    String vertexId = "Lily"
    graphClient.deleteVertex(vertexId);
}
```

4.14 添加边

您可以使用GES提供的接口添加边。示例代码如下：

```
public static void addEdge(GraphClient graphClient) throws ApiException
{
      AddEdgeReq req = new AddEdgeReq();
      req.setSource("Lily");
      req.setTarget("Rocky");
      req.setLabel("rate");
      Map properties = new HashMap();
      List score = new ArrayList();
      score.add("5");
      properties.put("Score", score);
      List datetime = new ArrayList();
      datetime.add("2018-01-01 20:30:05");
      properties.put("Datetime", datetime);
      req.setProperties(properties);
      graphClient.addEdge(req);
}
```

4.15 删除边

您可以使用GES提供的接口删除边。示例代码如下：

```
public static void deleteEdge(GraphClient graphClient) throws ApiException
{
  DeleteEdgeReq deleteEdgeReq = new DeleteEdgeReq();
  deleteEdgeReq.setSource("Lily");
  deleteEdgeReq.setTarget("Rocky");
  deleteEdgeReq.setIndex(0);
  graphClient.deleteEdge(deleteEdgeReq);
}
```

4.16 添加索引

您可以使用GES提供的接口添加索引。示例代码如下：

```
public static void excuteCreateIndex(GraphClient graphClient) throws ApiExcepti
 {
     CreateIndexReq req = new CreateIndexReq();
     //表示是否有label，不区分大小写，不填时默认为false
     req.setHasLabel("true");
     //索引名称，只能包含字母或数字不能包含特殊字符，无默认值
     req.setIndexName("ageIndex");
     //元素类型，可选值有“vertex“或”edge“，不区分大小写，无默认值
     req.setElementType("vertex");
     //索引类型，可选”GlobalCompositeVertexIndex”或”GlobalCompositeEdgeIndex”，区分大小写，无默
认值
     // indexType应与elementType一一对应，如elementType为vertex时，indexType应为
GlobalCompositeVertexIndex；elementType为edge时，indexType应为GlobalCompositeEdgeIndex，否则会创
建索引失败
     req.setIndexType("GlobalCompositeVertexIndex");
     //索引的属性列表 (支持的属性类型有：int, float, double, long, enum, char array, string, date)
     //若hasLabel为false或null，则该项为必选
     req.setIndexProperty(req.setIndexProperty(new String[] {"age"});)
     Map<String, Object> result = graphClient.createIndex(req);
     System.out.print(MapUtils.map2json(result));
 }
```


4.17 删除索引

您可以使用GES提供的接口删除索引。示例代码如下：

```
public static void excuteDeleteIndex(GraphClient graphClient) throws ApiException, GraphSdkException
{
    //删除indexName为ageIndex的索引
    Map<String, Object> result = graphClient.deleteIndex("ageIndex");
    System.out.print(MapUtils.map2json(result));
}
```

4.18 查询索引

您可以使用GES提供的接口查询索引。示例代码如下：

```
public static void excuteQueryIndex(GraphClient graphClient) throws ApiException
{
    //查询该图上的所有索引
    QueryIndexResult result = graphClient.queryIndex();
    System.out.print(result);
}
```

4.19 导出图

您可以使用GES提供的接口导出图的数据。示例代码如下：

```
public static void exportGraph(GraphClient graphClient) throws ApiException
{
ExportGraphReq exportGraphReq = new ExportGraphReq();
exportGraphReq.setGraphName(graphInfo.getGraphName());
exportGraphReq.setEdgeSetName("movie-edge.csv");
exportGraphReq.setVertexSetName("movie-vertex.csv");
exportGraphReq.setSchemaName("movie-schema.csv");
exportGraphReq.setGraphExportPath("imagebucket/movie/");
ObsParameters obsParameters = new ObsParameters();
obsParameters.setAccessKey("***");
obsParameters.setSecretKey("***");
obsParameters.setRegion("cn-hk1");
exportGraphReq.setObsParameters(obsParameters);
graphClient.exportGraph(exportGraphReq);
}
```

4.20 清除图

您可以使用GES提供的接口清除图的数据。示例代码如下：

```
public static void clearGraph(GraphClient graphClient) throws ApiException {
    ClearGraphReq clearGraphReq = new ClearGraphReq();
    clearGraphReq.setGraphName(graphInfo.getGraphName());
    graphClient.clearGraph(clearGraphReq);
}
```

4.21 添加 Property

您可以使用GES提供的接口添加Property。示例代码如下：

```
public static void addProperties(GraphClient graphClient) throws ApiException
{
    AddPropertiesReq req = new AddPropertiesReq();
```

```
    req.setLabelName("stop");
    List<Property> properties = new ArrayList<Property>();
    Property property = new Property();
    property.put("name", "a");
    property.put("cardinality", "single");
    property.put("dataType", "int");
    properties.add(property);
    req.setProperties(properties);
    Map<String, Object> result = graphClient.addProperties(req);
    System.out.println(MapUtils.map2json(result));
}
```

4.22 查询路径详情

您可以使用GES提供的接口查询路径详情。示例代码如下：

```
public static void getPathDetail(GraphClient graphClient) throws ApiException
{
    PathDetailReq pathDetailReq = new PathDetailReq();
    pathDetailReq.setDirected(true);
    List<List<String>> paths = new ArrayList();
    List<String> path1 = new ArrayList();
    path1.add("38");
    path1.add("0");
    List<String> path2 = new ArrayList();
    path2.add("35");
    path2.add("40");
    paths.add(path1);
    paths.add(path2);
    pathDetailReq.setPaths(paths);
    Map<String, Object> result = graphClient.getPathDetail(pathDetailReq);
    System.out.println(MapUtils.map2json(result));
}
```

4.23 增量导入图

您可以使用GES提供的接口增量导入图。示例代码如下：

```
public static void incrementImport(GraphClient graphClient) throws ApiException
{
    ImportGraphReq importGraphReq = new ImportGraphReq();
    importGraphReq.setGraphName(graphName);
    importGraphReq.setEdgesetPath("/home/lh/movie/ranking_edge.csv");
    importGraphReq.setEdgesetFormat("csv");
    importGraphReq.setVertexsetPath("/home/lh/movie/movies_vertex.csv");
    importGraphReq.setVertexsetFormat("csv_prop");
    ObsParameters obsParameters = new ObsParameters();
    obsParameters.setAccessKey("XXXXXXXX");
    obsParameters.setSecretKey("XXXXXXXX");
    obsParameters.setRegion("southchina");
    importGraphReq.setObsParameters(obsParameters);
    // 执行导入图
    AsyncAPIResp res = graphClient.incrementImport(importGraphReq);
    // 获取JobId
    String jobId = res.getJobId();
    ImportGraphJobReq req = new ImportGraphJobReq();
    req.setJobId(jobId);
    // 根据JobId查询导图结果
    System.out.println(graphClient.queryJobStatus(req));
}
```

4.24 批量添加点

您可以使用GES提供的接口批量添加点。示例代码如下：


```
public static void addBatchVertice(GraphClient graphClient) throws ApiException
{
    // 构造点属性列表
    Map<String, List<Object>> properties = new HashMap<>();
    properties.put("movieid", Arrays.asList("180"));
    properties.put("title", Arrays.asList("testmoive"));
    properties.put("genres", Arrays.asList("Comedy"));

    // 构造单个点信息
    AddVertexReq vertex = new AddVertexReq();
    vertex.setVertexId("180");
    vertex.setLabel("movie");
    vertex.setProperties(properties);

    // 组成批量点信息
    List<AddVertexReq> vertices = new ArrayList<>();
    vertices.add(vertex);

    // 构造添加批量点请求
    AddBatchVertexReq addBatchVertexReq = new AddBatchVertexReq();
    addBatchVertexReq.setVertices(vertices);

    // 执行添加批量点请求
    Map<String, Object> result = graphClient.addBatchVertex(addBatchVertexReq);
}
```

4.25 批量删除点

您可以使用GES提供的接口批量删除点。示例代码如下：

```
public static void deleteBatchVertice(GraphClient graphClient) throws ApiException
{
    String movieVertex = "2";
    String userVertex = "100";

    // 构造待删除点id列表
    List<String> vertices = new ArrayList<>();
    vertices.add(movieVertex);
    vertices.add(userVertex);

    // 构造批量删除点请求
    DeleteBatchVertexReq deleteBatchVertexReq = new DeleteBatchVertexReq();
    deleteBatchVertexReq.setVertices(vertices);

    // 执行批量删除点
    Map<String, Object> result = graphClient.deleteBatchVertex(deleteBatchVertexReq);
}
```

4.26 批量添加边

您可以使用GES提供的接口批量添加边。示例代码如下：

```
public static void addBatchEdges(GraphClient graphClient) throws ApiException
{
    // 构造边信息
    Edge edge = new Edge();
    edge.setSource("46");
    edge.setTarget("38");
    edge.setLabel("rate");
    Map<String, List<Object>> properties = new HashMap<>();
    properties.put("Rating", Arrays.asList("5"));
    properties.put("Datetime", Arrays.asList("2018-01-0120:30:05"));
    edge.setProperties(properties);

    // 组成边列表
    List<Edge> edges = new ArrayList<>();
    edges.add(edge);
```

```
    // 默认选项为允许重复边
    ParallelEdgeOption parallelEdgeOption = new ParallelEdgeOption();

    // 构造添加批量边请求
    AddBatchEdgeReq addBatchEdgeReq = new AddBatchEdgeReq();
    addBatchEdgeReq.setEdges(edges);
    addBatchEdgeReq.setParallelEdge(parallelEdgeOption);

    // 执行添加批量边请求
    Map<String, Object> result = graphClient.addBatchEdge(addBatchEdgeReq);
}
```

4.27 批量删除边

您可以使用GES提供的接口批量删除边。示例代码如下：

```
public static void deleteBatchEdges(GraphClient graphClient) throws ApiException
{
    // 构造单条边信息
    DeleteEdgeReq edge = new DeleteEdgeReq();
    edge.setSource("46");
    edge.setTarget("39");

    DeleteEdgeReq edgeWithIndex = new DeleteEdgeReq();
    edgeWithIndex.setSource("46");
    edgeWithIndex.setTarget("38");
    edgeWithIndex.setIndex("8");

    // 组成待删除边列表
    List<DeleteEdgeReq> edges = new ArrayList<>();
    edges.add(edge);
    edges.add(edgeWithIndex);

    // 构造删除批量边请求
    DeleteBatchEdgeReq deleteBatchEdgeReq = new DeleteBatchEdgeReq();
    deleteBatchEdgeReq.setEdges(edges);

    // 执行删除批量边请求
    Map<String, Object> result = graphClient.deleteBatchEdge(deleteBatchEdgeReq);
}
```

4.28 批量更新点属性

您可以使用GES提供的接口批量更新点属性。示例代码如下：

```
public static void updateBatchVertice(GraphClient graphClient) throws ApiException
{
    // 构造单个点属性信息
    Map<String, List<Object>> properties = new HashMap<>();
    properties.put("年龄", Arrays.asList(20));
    AddVertexReq vertex = new AddVertexReq();
    vertex.setVertexId("张三1");
    vertex.setProperties(properties);

    Map<String, List<Object>> listProperties = new HashMap<>();
    listProperties.put("名称", Arrays.asList("测试","数学"));
    AddVertexReq vertexWithListProperty = new AddVertexReq();
    vertexWithListProperty.setVertexId("张三0");
    vertexWithListProperty.setProperties(listProperties);

    Map<String, List<Object>> setProperties = new HashMap<>();
    setProperties.put("name", Arrays.asList("a","d"));
    AddVertexReq vertexWithSetProperty = new AddVertexReq();
    vertexWithSetProperty.setVertexId("张三140");
    vertexWithSetProperty.setProperties(setProperties);
```

```
    // 组成批量点属性信息
    List<AddVertexReq> vertices = new ArrayList<>();
    vertices.add(vertex);
    vertices.add(vertexWithListProperty);
    vertices.add(vertexWithSetProperty);

    // 构造更新批量点属性请求
    AddBatchVertexReq updateBatchVertexReq = new AddBatchVertexReq();
    updateBatchVertexReq.setVertices(vertices);

    // 执行更新批量点属性请求
    Map<String, Object> result = graphClient.updateBatchVertex("batch-update", updateBatchVertexReq);
}
```

4.29 批量更新边属性

您可以使用GES提供的接口批量更新边属性。示例代码如下：

```
public static void updateBatchEdges(GraphClient graphClient) throws ApiException
{
    // 构造边信息，不指定index
    EdgeWithIndex edgeWithoutIndex = new EdgeWithIndex();
    edgeWithoutIndex.setSource("46");
    edgeWithoutIndex.setTarget("37");
    edgeWithoutIndex.setLabel("rate");
    Map<String, List<Object>> properties = new HashMap<>();
    properties.put("Rating", Arrays.asList("5"));
    properties.put("Datetime", Arrays.asList("2020-01-0120:30:05"));
    edgeWithoutIndex.setProperties(properties);

    // 构造边信息，指定index
    EdgeWithIndex edgeWithIndex = new EdgeWithIndex();
    edgeWithIndex.setSource("46");
    edgeWithIndex.setTarget("38");
    edgeWithIndex.setIndex("0");
    edgeWithIndex.setProperties(properties);

    // 组成边列表
    List<EdgeWithIndex> edges = new ArrayList<>();
    edges.add(edgeWithoutIndex);
    edges.add(edgeWithIndex);

    // 构造更新批量边属性请求
    UpdateBatchEdgePropertyReq updateBatchEdgeReq = new UpdateBatchEdgePropertyReq();
    updateBatchEdgeReq.setEdges(edges);

    // 执行更新批量边属性请求
    Map<String, Object> result = graphClient.updateBatchEdge("batch-update", updateBatchEdgeReq);
}
```

5 使用管理面 SDK

5.1 初始化 GES 管理面客户端

使用GES SDK工具访问GES管理面，需要用户初始化GES客户端。用户可以使用AK/SK(Access Key ID/Secret Access Key)或Token两种认证方式初始化客户端，示例代码如下：

- AK/SK认证方式样例代码

```
String ak = "ak";
String sk = "sk";
String regionName = "regionname";
String projectId = "project_id";
GesInfo gesInfo = new GesInfo(regionName, ak, sk, projectId);
gesInfo.setIamEndpoint("iam_endpoint");
gesInfo.setGesEndpoint("ges_endpoint");
GesClient client = new GesClient(AuthenticationMode.AKSK, gesInfo);
```

- Token认证方式样例代码

```
String domainName = "domainname";
String userName = "username";
String password = "password";
String regionName = "regionname";
String projectId = "project_id";
GesInfo gesInfo = new GesInfo(regionName, domainName, userName, password, projectId);
gesInfo.setIamEndpoint("iam_endpoint");
gesInfo.setGesEndpoint("ges_endpoint");
GesClient client = new GesClient(AuthenticationMode.TOKEN, gesInfo);
```

5.2 查询配额

GES提供查询配额接口，您可以使用该接口查询当前租户的配额，包括图配额和图备份配额。示例代码如下：

```
private static void getQuotas(GesClient client) throws GesSdkException{
    Quotas quotas = client.getQuotas();
    System.out.println(quotas);
}
```


5.3 校验元数据文件

GES提供校验元数据文件接口，您可以使用该接口校验选择的数据集和元数据文件是否匹配。示例代码如下：

```
private static void checkSchema(GesClient client) throws GesSdkException {
    Schema schema = new Schema();
    schema.setSchemaPath("gesdata/movie/schema.xml");
    schema.setEdgesetPath("gesdata/movie/edge.csv");
    schema.setVertexsetPath("gesdata/movie/vertex.csv");
    boolean checkResult = client.checkSchema(schema);
    System.out.println(checkResult);

}
```

5.4 查询图列表

您可以使用GES提供的接口查询图列表，示例代码如下：

```
private static void listGraphs(GesClient client) throws GesSdkException {
    GraphList graphList = client.listGraphs();
    for (Graph graph : graphList.getGraphs())
    {
        System.out.println(graph);
    }
}
```

5.5 查询图详情

您可以使用GES提供的接口查询图详情，示例代码如下：

```
private static void getGraphDetail(GesClient client, String graphId) throws GesSdkException {
    GraphDetail graphDetail = client.queryGraphById(graphId);
    System.out.println(graphDetail);
}
```

5.6 创建图

您可以使用GES提供的接口创建一个图。示例代码如下：

```
private static void createGraph(GesClient client) throws GesSdkException {
    GraphReq graphReq = new GraphReq();
    graphReq.setName("ges0211");
    graphReq.setRegionCode("cn-north-1");
    graphReq.setAzCode("cn-north-1a");
    graphReq.setGraphSizeTypeIndex(1);
    graphReq.setSchemaPath("gesdata/movie/schema.xml");
    graphReq.setVertexsetPath("gesdata/movie/vertex.csv");
    graphReq.setEdgesetPath("gesdata/movie/edge.csv");
    graphReq.setEdgesetFormat("csv");
    graphReq.setVpcId("98a8900c-bd4c-4a29-a488-93f4b71378fb");
    graphReq.setSubnetId("b491203a-bcb7-4e0f-88e2-cdab3a636eef");
    graphReq.setSecurityGroupId("cefb75f5-fc97-4c82-a613-42d55299bd12");
    CreateGraphReq createGraphReq = new CreateGraphReq();
    createGraphReq.setGraph(graphReq);
    Graph graph = client.createGraph(createGraphReq);
    System.out.println(graph);
}
```

5.7 关闭图

您可以使用GES提供的接口关闭一个图。示例代码如下：

```
private static void stopGraph(GesClient client, String graphId) throws GesSdkException
  {
     Job job = client.stopGraph(graphId);
     System.out.println(job);
  }
```

5.8 启动图

您可以使用GES提供的接口启动一个图，可以从原图启动或者从某个备份ID启动。示例代码如下：

```
//从原图启动
private static void startGraph(GesClient client, String graphId) throws GesSdkException
  {
     Job job = client.startGraph(graphId);
     System.out.println(job);
  }
//从备份启动
 private static void startGraphFromBackup(GesClient client, String graphId, String backUpId) throws
GesSdkException
  {
     Job job = client.startGraph(graphId,backUpId);
     System.out.println(job);
  }
```

5.9 删除图

您可以使用GES提供的接口删除一个图。示例代码如下：

```
private static void deleteGraph(GesClient client, String graphId) throws GesSdkException
  {
     Job job = client.deleteGraph(graphId);
     System.out.println(job);
  }
```

5.10 增量导入图

您可以使用GES提供的接口增量导入图。示例代码如下：

```
public static void incrementImport(GraphClient graphClient) throws ApiException
{
    ImportGraphReq importGraphReq = new ImportGraphReq();
    importGraphReq.setGraphName(graphName);
    importGraphReq.setEdgesetPath("/home/lh/movie/ranking_edge.csv");
    importGraphReq.setEdgesetFormat("csv");
    importGraphReq.setVertexsetPath("/home/lh/movie/movies_vertex.csv");
    importGraphReq.setVertexsetFormat("csv_prop");
    ObsParameters obsParameters = new ObsParameters();
    obsParameters.setAccessKey("EW39NCDEXJ4E1JTN2PCP");
    obsParameters.setSecretKey("rhsS0TP89IdNnDe6dug1iraEbQUeNZlbJ3QGgW5D");
    obsParameters.setRegion("southchina");
    importGraphReq.setObsParameters(obsParameters);
    // 执行导入图
    AsyncAPIResp res = graphClient.incrementImport(importGraphReq);
    // 获取JobId
    String jobId = res.getJobId();
```

```
    ImportGraphJobReq req = new ImportGraphJobReq();
    req.setJobId(jobId);
    // 根据JobId查询导图结果
    System.out.println(graphClient.queryJobStatus(req));
}
```

5.11 导出图

您可以使用GES提供的接口导出图的数据。示例代码如下：

```
public static void exportGraph(GraphClient graphClient) throws ApiException
{
ExportGraphReq exportGraphReq = new ExportGraphReq();
exportGraphReq.setGraphName(graphInfo.getGraphName());
exportGraphReq.setEdgeSetName("movie-edge.csv");
exportGraphReq.setVertexSetName("movie-vertex.csv");
exportGraphReq.setSchemaName("movie-schema.csv");
exportGraphReq.setGraphExportPath("imagebucket/movie/");
ObsParameters obsParameters = new ObsParameters();
obsParameters.setAccessKey("***");
obsParameters.setSecretKey("***");
obsParameters.setRegion("cn-hk1");
exportGraphReq.setObsParameters(obsParameters);
graphClient.exportGraph(exportGraphReq);
}
```

5.12 清空图

您可以使用GES提供的接口清除图的数据。示例代码如下：

```
public static void clearGraph(GraphClient graphClient) throws ApiException {
    ClearGraphReq clearGraphReq = new ClearGraphReq();
    clearGraphReq.setGraphName(graphInfo.getGraphName());
    graphClient.clearGraph(clearGraphReq);
}
```

5.13 升级图

您可以使用GES提供的接口升级一个图。示例代码如下：

```
public void upgradeGraph() throws Exception {
      UpgradeGraphReq upgradeGraphReq = new UpgradeGraphReq();
       Status status = client.upgradeGraph(graphId);
       System.out.println(status);
   }
```

5.14 绑定 EIP

您可以使用GES提供的接口将EIP绑定到一个图上。示例代码如下：

```
public void testBindEip() throws Exception {
       BindEipReq bindEipReq = new BindEipReq();
       Status status = client.bindEip(graphId, bindEipReq);
       System.out.println(status);
   }
```

5.15 解绑 EIP

您可以使用GES提供的接口将一个图绑定的EIP解绑。示例代码如下：


```
public void testUnbindEip() throws Exception {
        UnBindEipReq unBindEipReq = new UnBindEipReq();
        Status status = client.unbindEip(graphId, unBindEipReq);
        System.out.println(status);
    }
```

5.16 查询所有备份列表

您可以使用GES提供的接口查询当前租户所有图的备份列表。示例代码如下：

```
private static void listBackups(GesClient client) throws GesSdkException
    {
        GesBackUpList gesBackUpList = client.queryBackups();
        if (null == gesBackUpList.getBackupList())
        {
            System.out.println("backup is null");
        }
        else
        {
            for(GesBackUp backUp: gesBackUpList.getBackupList())
            {
                System.out.println(backUp);
            }
        }
    }
```

5.17 查询某个图的备份列表

您可以使用GES提供的接口查询当前租户某个图的备份列表。示例代码如下：

```
private static void listBackupsByGraphId(GesClient client, String graphId) throws GesSdkException
    {
        GesBackUpList gesBackUpList = client.queryBackupById(graphId);
        if (null == gesBackUpList.getBackupList())
        {
            System.out.println("backup is null");
        }
        else
        {
            for(GesBackUp backUp: gesBackUpList.getBackupList())
            {
                System.out.println(backUp);
            }
        }
    }
```

5.18 新增备份

您可以使用GES提供的接口新增图备份。示例代码如下：

```
private static void createBackup(GesClient client, String graphId) throws GesSdkException
    {
        Job job = client.addBackup(graphId);
        System.out.println(job);
    }
```

5.19 删除备份

您可以使用GES提供的接口将删除图备份。示例代码如下：

```
private static void deleteBackup(GesClient client, String graphId, String backupId) throws GesSdkException
    {
```



```
    boolean b = client.deleteBackup(graphId, backupId);
    System.out.println(b);
}
```

5.20 查询 Job 状态

您可以使用GES提供的接口查询Job状态。示例代码如下：

```
private static void getJobStatus(GesClient client, String graphId, String jobId) throws GesSdkException
 {
    JobResp jobsResp = client.queryJobStats(graphId, jobId);
    System.out.println(jobsResp);
 }
```

说明

对停止图、启动图、恢复图、删除图、创建备份等异步API，命令下发后，会返回jobId，可以通过jobId查询任务的执行状态。

5.21 查询任务中心

您可以使用GES提供的接口查询任务中心。示例代码如下：

```
private static void queryJobs(GesClient client) throws GesSdkException
 {
    JobsResp jobResp = client.queryJobs();
    System.out.println(jobResp);
 }
```

说明

对停止图、启动图、恢复图、删除图、创建备份等异步API，命令下发后，会返回jobId，可以通过jobId查询任务的执行状态。

6 使用 Cypher JDBC Driver 访问 GES

功能介绍

GES Cypher JDBC Driver是专为GES编写的JDBC驱动，基于Neo4j JDBC Driver中的接口，提供了使用JDBC访问GES并进行cypher查询的一种方法。

尤其是当cypher请求返回数据量较大、并发数高、JVM缓存完整请求体有困难的场景下，该组件内置了一种可以流式解析响应body体的方法，与获得整个body体再解析相比，极大地降低了cpu和内存的占用。

依赖配置

使用前需进行工程导入，并配置maven工程，pom依赖配置如下：

```
<dependency>
    <groupId>com.huawei.ges</groupId>
    <artifactId>cypher-jdbc-driver</artifactId>
    <version>1.1.0</version>
</dependency>
```

使用参数

表 6-1 JDBC getConnection 参数说明

参数	释义
url	GES Cypher API的URL，添加前缀jdbc:ges:http(s)为前缀以方便JDBC Driver识别，是DriverManager.getConnection的第一个参数。
prop	Properties对象，包含连接GES API所需的各项配置，详见表6-2。

表 6-2 Properties 参数信息

参数	释义
X-Auth-Token	通过iam鉴权接口获取到的token。

parse-json	是否转换点边对象，默认值为"false", 取值为false时，cypher返回体中的点和边将以map形式返回，为true时，以GesElement对象的形式返回。

使用示例

```
package org.example;

import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Properties;

public class App
{
    static String ip = "${ip}";
    static int port = 80;
    static String projectId = "${projectId}";
    static String graphName = "${graphName}";
    static String token = "${x_auth_token}";
    public static void main(String[] args) throws ClassNotFoundException, IllegalAccessException,
InstantiationException {
        Class.forName("com.huawei.ges.jdbc.Driver").newInstance();
        String url = "jdbc:ges:http://{{graph_ip}}:{{graph_port}}/ges/v1.0/{{project_id}}/graphs/{{graph_name}}/
action?action_id=execute-cypher-query";
        url = url.replace("{{graph_ip}}", ip).replace("{{graph_port}}",port + "").replace("{{project_id}}",
projectId).replace("{{graph_name}}", graphName);
        Properties prop = new Properties();
        prop.setProperty("X-Auth-Token", token);
        prop.setProperty("deserializer-type","lazy");
        prop.setProperty("parse-json","true");
        prop.setProperty("limit","10000");
        try(Connection conn = DriverManager.getConnection(url,prop)){
            String query = "match (m) return m limit 1000";
            try(PreparedStatement stmt = conn.prepareStatement(query)){
                try(ResultSet rs = stmt.executeQuery()){
                    Object o = null;
                    while(rs.next()) {
                        o = rs.getObject("m");
                        processVertex(o);
                    }
                }
            }
        } catch (SQLException e) {
            // here process exception.
            // ...
        }
    }
}
```

鉴权方法

GES Cypher JDBC Driver支持Token和AK/SK两种鉴权模式。

- Token鉴权相关参数详见**使用参数**和**使用示例**。
- AKSK鉴权需要依赖GES业务面SDK获取AK/SK签名后，使用签名进行鉴权操作。

  导入业务面SDK依赖详见**工程导入**，GraphInfo的配置详见**初始化GES业务面客户端**，并且需要您输入获取到的AccessKey，secretKey和regionName参数。

  以AK/SK鉴权方式为例，代码示例如下：


```
import com.huawei.ges.jdbc.io.model.GesElement;
import com.huawei.graph.sdk.GraphInfo;
import com.huawei.graph.sdk.exception.GraphSdkException;
import com.huawei.graph.sdk.utils.HttpRestClient;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;
import java.util.Map;

public class CypherJDBCClientByAKSK {

    private static final Logger logger = LoggerFactory.getLogger("CypherJDBCClientByAKSK");
    private static String ip = "";
    private static int port = 80;
    private static String projectId = "";
    private static String graphName = "";
    String accessKey = "here is access key";
    String secretKey = "here is secret key";
    String regionName = "cn-north-4";
    public static GraphInfo getGraphInfo() {
        //正式代码应该通过正常方式初始化graphInfo对象。
        GraphInfo info = getGraphInfoByYourSelf();
       // 此处需要输出您的AK/SK信息
        info.setAccessKey(accessKey);
        info.setSecretKey(secretKey);
      // 此处需要输出您的regionName
        info.setRegionName(regionName);
        return info;
    }

    public static void main(String[] args) throws ClassNotFoundException, IllegalAccessException,
InstantiationException, GraphSdkException {

        GraphInfo info = getGraphInfo();

        Map<String, String> iamHeader = HttpRestClient.getIamSignHeaders(info);
        Class.forName("com.huawei.ges.jdbc.Driver").newInstance();
        String url = "jdbc:ges:http://{{graph_ip}}:{{graph_port}}/ges/v1.0/{{project_id}}/graphs/
{{graph_name}}/action?action_id=execute-cypher-query";
        url = url.replace("{{graph_ip}}", ip).replace("{{graph_port}}", port + "").replace("{{project_id}}",
projectId).replace("{{graph_name}}", graphName);
        doCypherQuery(url, iamHeader);
    }

    public static void doCypherQuery(String url, Map<String, String> iamHeaders) {
        Properties prop = new Properties();
        for (Map.Entry<String, String> pair : iamHeaders.entrySet()) {
            prop.setProperty(pair.getKey(), pair.getValue());
        }
        prop.setProperty("deserializer-type", "lazy");
        prop.setProperty("parse-json", "true");
        prop.setProperty("limit", "10000");
        try (Connection conn = DriverManager.getConnection(url, prop)) {
            String query = "match (m) return m limit 1";
            try (PreparedStatement stmt = conn.prepareStatement(query)) {
                try (ResultSet rs = stmt.executeQuery()) {
                    Object o = null;
                    while (rs.next()) {
                        GesElement.GesVertex vertex = (GesElement.GesVertex) rs.getObject("m");
                        System.out.println(vertex.getId());
                        System.out.println(vertex.getLabels());
                        System.out.println(vertex.getProperties());
                    }
```



```
                }
            }
        } catch (SQLException e) {
            logger.info("Execute SQL query error.");
        }
    }
}
```


7 SDK 和 REST API 的关系

表 7-1 SDK 和 REST API 的关系

Interface	Method	API
Graph Business API	addVertex()	POST /ges/v1.0/{projectId}/graphs/{graphName}/vertices
	deleteVertex(GraphClient graphClient)	DELETE /ges/v1.0/{projectId}/graphs/{graphName}/vertices/{vertexId}/
	vertexQuery(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/vertices/action?action_id=query
	getVertices(GraphClient graphClient)	GET /ges/v1.0/{projectId}/graphs/{graphName}/vertices/detail?vertexIds={vertexIds}
	addEdge(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/edges
	deleteEdge(GraphClient graphClient)	DELETE /ges/v1.0/{projectId}/graphs/{graphName}/edges?source={sourceVertex}&target={targetVertex}&index={index}

Interface	Method	API
	edgeQuery(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/edges/action?action_id=query
	getEdge(GraphClient graphClient)	GET /ges/v1.0/{projectId}/graphs/{graphName}/edges/detail?source={sourceVertex}&target={targetVertex}&index={index}
	getSchema(GraphClient graphClient)	GET /ges/v1.0/{projectId}/graphs/{graphName}/schema
	getSummary(GraphClient graphClient)	GET /ges/v1.0/{projectId}/graphs/{graphName}/summary
	executeGremlinQuery(GraphClient graphClient, String graphName)	POST /ges/v1.0/{projectId}/{userId}/graphs/action?action_id=execute-gremlin-query
	addLabel(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/schema?labels
	excuteCreateIndex(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/indices
	excuteDeleteIndex(GraphClient graphClient)	DELETE /ges/v1.0/{projectId}/graphs/{graphName}/indices/{indexName}
	excuteQueryIndex(GraphClient graphClient)	GET /ges/v1.0/{projectId}/graphs/{graphName}/indices
	exportGraph(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/action?action_id=export-graph

Interface	Method	API
	clearGraph(GraphClient graphClient)	POST /ges/v1.0/{projectId}/graphs/{graphName}/action?action_id=clear-graph
	queryAsyncTask(GraphClient graphClient,String jobId)	GET /ges/v1.0/{projectId}/graphs/jobs/{jobId}?offset=offset&limit=limit
	stopAsyncTask(GraphClient graphClient,String jobId)	DELETE /ges/v1.0/{projectId}/graphs/jobs/{jobId}
	executeAlgorithm(GraphClient graphClient, String graphName)	POST /ges/v1.0/{projectId}/{userId}/graphs/action?action_id=execute-algorithm
Graph Management API	getQuotas(GesClient client)	GET /v1.0/{projectId}/graphs/quotas
	checkSchema(GesClient client)	POST /v1.0/{projectId}/graphs/action?action_id=check-schema
	listGraphs(GesClient client)	GET /v1.0/{projectId}/graphs?offset={offset}&limit={limit}
	getGraphDetail(GesClient client, String graphId)	GET /v1.0/{projectId}/graphs/{graphId}
	createGraph(GesClient client)	POST /v1.0/{projectId}/graphs
	stopGraph(GesClient client, String graphId)	POST /v1.0/{projectId}/graphs/{graphId}/action?action_id=stop
	startGraph(GesClient client, String graphId)	POST /v1.0/{projectId}/graphs/{graphId}/action?action_id=start

Interface	Method	API
	deleteGraph(GesClient client, String graphId)	DELETE /v1.0/{projectId}/graphs/{graphId}
	listBackups(GesClient client)	GET /v1.0/{projectId}/graphs/backups?offset={offset}&limit={limit}
	listBackupsByGraphId(GesClient client, String graphId)	GET /v1.0/{projectId}/graphs/{graphId}/backups?offset={offset}&limit={limit}
	createBackup(GesClient client, String graphId)	POST /v1.0/{projectId}/graphs/{graphId}/backups
	deleteBackup(GesClient client, String graphId, String backupId)	DELETE /v1.0/{projectId}/graphs/{graphId}/backups/{backup_id}
	getJobStatus(GesClient client, String graphId, String jobId)	GET /v1.0/{projectId}/graphs/{graphId}/jobs/{jobId}/status