GES SDK Reference

Issue 01

Date 2023-01-16

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Overview....1
2 Preparations....2
3 Importing a Project....3
4 Using the Service Plane SDK....4
4.1 Initializing the Client of the GES Service Plane....4
4.2 Querying Vertices That Meet Filtering Conditions....5
4.3 Querying Edges That Meet Filtering Conditions....5
4.4 Querying Vertex Details....6
4.5 Querying Edge Details....6
4.6 Querying Graph Metadata Details....6
4.7 Querying General Information About a Graph....6
4.8 Executing Gremlin Queries....7
4.9 Running Algorithms....7
4.9.1 PageRank....7
4.9.2 PersonalRank....7
4.9.3 K-hop....8
4.9.4 K-core....8
4.9.5 Shortest Path....8
4.9.6 All Shortest Paths....9
4.9.7 Filtered Shortest Path....9
4.9.8 SSSP....9
4.9.9 Shortest Path of Vertex Sets....10
4.9.10 n Paths....10
4.9.11 Closeness....11
4.9.12 Label Propagation....11
4.9.13 Louvain Algorithm....11
4.9.14 Link Prediction....11
4.9.15 Node2vec....12
4.9.16 Real-time Recommendation....12
4.9.17 Common Neighbors....12
4.9.18 Connected Component....13
4.9.19 Degree Correlation Algorithm....13

4.9.20 Triangle Count....13
4.9.21 Cluster Coefficient....14
4.9.22 Filtered Circle Detection....14
4.9.23 Subgraph Matching....14
4.9.24 Filtered All Pairs Shortest Paths....15
4.9.25 Filtered All Shortest Paths....15
4.9.26 TopicRank....15
4.9.27 Filtered n Paths....16
4.9.28 Common Neighbors of Vertex Sets....16
4.9.29 All Shortest Paths of Vertex Sets....17
4.10 Querying Job Status....17
4.11 Canceling a Job....17
4.12 Adding a Vertex....18
4.13 Deleting a Vertex....18
4.14 Adding an Edge....18
4.15 Deleting an Edge....19
4.16 Adding an Index....19
4.17 Deleting an Index....19
4.18 Querying Indexes....20
4.19 Exporting a Graph....20
4.20 Removing a Graph....20
4.21 Adding a Property....20
4.22 Querying Path Details....21
4.23 Incrementally Importing Data to Graphs....21
4.24 Adding Vertices in Batches....21
4.25 Deleting Vertices in Batches....22
4.26 Adding Edges in Batches....22
4.27 Deleting Edges in Batches....23
4.28 Updating Vertex Properties in Batches....23
4.29 Updating Edge Properties in Batches....24
5 Using the Management Plane SDK....25
5.1 Initializing the Client of the GES Management Plane....25
5.2 Querying Quotas....25
5.3 Verifying Metadata Files....26
5.4 Querying the Graph List....26
5.5 Querying Graph Details....26
5.6 Creating a Graph....26
5.7 Closing a Graph....27
5.8 Starting a Graph....27
5.9 Deleting a Graph....27
5.10 Incrementally Importing Data to a Graph....27
5.11 Exporting a Graph....28

5.12 Clearing a Graph....28
5.13 Upgrading a Graph....28
5.14 Binding EIPs....28
5.15 Unbinding an EIP....29
5.16 Querying Backups of All Graphs....29
5.17 Querying the Backup List of a Graph....29
5.18 Adding a New Backup....29
5.19 Deleting a Backup....30
5.20 Querying Job Status....30
5.21 Query Information in the Task Center....30
6 Using Cypher JDBC Driver to Access GES....31
7 Relationships Between SDKs and REST APIs....35

1 Overview

Introduction

Graph Engine Service (GES) facilitates query and analysis of multi-relational graph data structures. It is particularly well suited for scenarios requiring analysis of rich relationships, including social network analysis, marketing recommendations, social listening, information distribution, and fraud detection.

Development Guide Overview

GES Software Development Kit (SDK) is the encapsulation of RESTful APIs provided by GES to simplify your development. You can directly call APIs provided by the GES SDK to use GES functions.

Content Navigation

This development guide describes how to install and configure an environment and call GES APIs for secondary development.

Chapter	Describes
Overview	GES and development concepts
Preparations	Required environment configuration and method to download the GES SDK
Importing a Project	How to import a project
Using the Service Plane SDK	Common operations using the GES SDK on the service plane
Using the Management Plane SDK	Common operations using the GES SDK on the management plane
Relationships Between SDKs and REST APIs	Relationships between GES SDKs and REST APIs

2 Preparations

Preparing the Environment

- Download JDK1.8 or a later version from the **Oracle official website**, install the JDK, and configure Java environment variables.
- Download Eclipse IDE for Java Developers of the latest version from the **Eclipse**, and install it.
- Configure the JDK in Eclipse.

Downloading SDKs

Log in to the GES management console. In the left navigation pane, click **Connection Management**. The **Connection Management** page is displayed. For details, see **Managing Connections**.

3 Importing a Project

Importing External Jar Packages to the Local Maven Repository

1. Decompress **huaweicloud-ges-sdk-java-xxx.zip**, open it, go to the **maven-install** directory, and execute the **ges-sdk-java-maven-install.bat** or **ges-sdk-java-maven-install.sh** file to install **sdk-common-xxx.jar**, **ges-sdk-xxx.jar**, **graph-sdk-xxx.jar**, **cypher-jdbc-driver-xxx.jar**, and **java-sdk-core-xxx.jar** to the local Maven repository.
2. Create a Maven project and add the following dependency to the POM file.

```
<dependency>
   <groupId>com.huawei.ges</groupId>
   <artifactId>ges-sdk</artifactId>
   <version>xxx</version>  // Enter the version number of the current management plane SDK.
</dependency>

<dependency>
   <groupId>com.huawei.ges.graph</groupId>
   <artifactId>graph-sdk</artifactId>
   <version>xxx</version>  // Enter the version number of the current service plane SDK.
</dependency>
<!-- Import the dependency if the Cypher JDBC driver is used.-->
<dependency>
   <groupId>com.huawei.ges</groupId>
   <artifactId>cypher-jdbc-driver</artifactId>
  <version>xxx</version>  // Enter the version number of the Cypher JDBC driver.
</dependency>
```

Importing the JAR Package from the Local PC

Create a project, decompress the **huaweicloud-ges-sdk-java-xxx.zip** package, open it, and import **ges-sdk-xxx-jar-with-dependencies.jar** and **graph-sdk-xxx-jar-with-dependencies.jar** files in the **huaweicloud-ges-sdk-java-xxx** directory to the project or import all packages in the **ges-sdk-xxx.jar**, **graph-sdk-xxx.jar**, **cypher-jdbc-driver-xxx.jar**, and **libs** directories to the project.

4 Using the Service Plane SDK

4.1 Initializing the Client of the GES Service Plane

To use the GES SDK to access GES, you need to initialize the GES client.

- When you access a graph through the intranet, the endpoint is the intranet IP address on the GES console or the **privateIp** field in the return result of the graph details query.
- When you access a graph through the Internet, the endpoint is the Internet IP address on the GES console or the **publicIp** field in the return result of the graph details query.

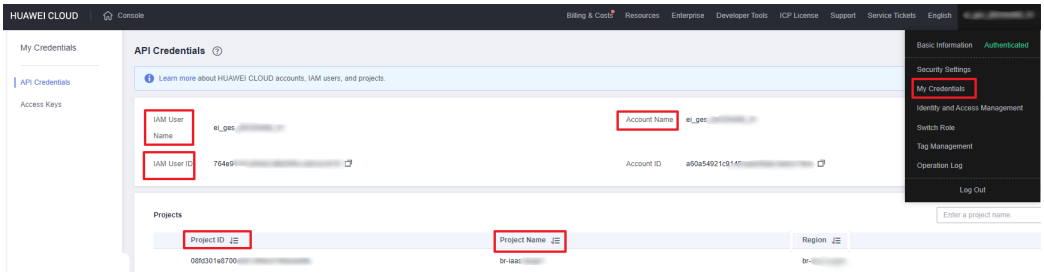
The sample code is as follows:

```
String endPoint = "endpoint";
String version = "v1.0";
String projectId = "project_id";
String graphName = "graph_name";
// Note: When the public network is used for access, token authentication must be used, and the following dominName, userName, and password must be provided.
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);
// Generate and initialize the graph client. During initialization, an object with the same name as the graph is generated by default. This object can be directly used in Gremlin commands.
GraphClient graphClient = new GraphClient(AuthenticationMode.TOKEN,graphInfo);
```

NOTE

- You can view the graph name and floating and public IP addresses on the **Graph Management** page of the GES console. For details, see **Graph Management Overview**.
- To obtain the region name, domain name, username, user ID, and project ID, move the cursor on your username in the upper right corner of the GES console and select My Credentials from the drop-down list. On the displayed page, the **Account Name** is the domain name, the **IAM User Name** is the username, and the **IAM User ID** is the user ID. In the **Projects** pane, the **Project Name** is the region name, and the value of **Project ID** is the project ID.



4.2 Querying Vertices That Meet Filtering Conditions

You can use a specific API provided by GES to query the vertex set that meet the filtering conditions. The sample code is as follows:

```
 public static void vertexQuery(GraphClient graphClient) throws ApiException
{
//Request of querying vertices that meet the filtering conditions
VertexFilterQueryReq req = new VertexFilterQueryReq();
 List<String> labels = new ArrayList<>();
 labels.add("movie");
//Filter vertices by label.
req.setLabels(labels);
//Set the offset and limit.
req.setOffset(0);
 req.setLimit(5);
//Send the request to obtain the asynchronous API result.
AsyncAPIResp asyncAPIResp = graphClient.vertexQuery(req);
//This API is an asynchronous API. Obtain the jobId.
String jobId = asyncAPIResp.getJobId();
//Vertex query job request
VertexQueryJobReq req1 = new VertexQueryJobReq();
//Set the jobId.
req1.setJobId(jobId);
//Obtain the job query result.
JobResp<QueryData<VertexQueryResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```

4.3 Querying Edges That Meet Filtering Conditions

You can use a specific API provided by GES to query the edge set that meet the filtering conditions. The sample code is as follows:

```
 public static void edgeQuery(GraphClient graphClient) throws ApiException
{
//Request of querying edges that meet the filtering conditions
EdgeFilterQueryReq req = new EdgeFilterQueryReq();
//Filter edges by label.
List<String> lables = new ArrayList<>();
```

```
 lables.add("rate");
 req.setLabels(lables);
//Set the offset and limit to limit the number of results.
 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 Querying Vertex Details

You can use a specific API provided by GES to query vertex details. The sample code is as follows:

```
public static  void getVertices(GraphClient graphClient) throws ApiException
{
 VertexDetailReq req = new VertexDetailReq();
 req.setVertexIds("46");// Replace with the actual vertex ID.
 VertexDetailResult vertexDetailResult = graphClient.getVertices(req);
 System.out.print(vertexDetailResult);
}
```

4.5 Querying Edge Details

You can use a specific API provided by GES to query edge details. The sample code is as follows:

```
 public static void getEdge(GraphClient graphClient) throws ApiException
{
 EdgeDetailReq req = new EdgeDetailReq();
 req.setSource("46");//Replace with the actual vertex ID.
 req.setTarget("38");//Replace with the actual vertex ID.
 req.setIndex(0);
 EdgeDetailResult edgeDetailResult = graphClient.getEdge(req);
 System.out.print(edgeDetailResult);
}
```

4.6 Querying Graph Metadata Details

You can use a specific API provided by GES to query metadata details. The sample code is as follows:

```
 public static void getSchema(GraphClient graphClient) throws ApiException
{   Map result = graphClient.getGraphSchema();
    //Output the result in a specific format using the MapUtils tool class.
    System.out.print(MapUtils.map2json(result));
}
```

4.7 Querying General Information About a Graph

You can use a specific API provided by GES to query general information of a graph. The sample code is as follows:

```
  public static void getSummary(GraphClient graphClient) throws ApiException
{E
    Map result = graphClient.getGraphSummary();
```



```
    //Output the result in a specific format using the MapUtils tool class.
    System.out.print(MapUtils.map2json(result));
}
```

4.8 Executing Gremlin Queries

You can use a specific API provided by GES to perform Gremlin queries. The sample code is as follows:

NOTE

During graph client initialization, an object with the same name as the graph has been generated in the system. You can use the graph name to query the object.

```
 public static void executeGremlinQuery(GraphClient graphClient,String graphName) throws ApiException
{   //Gremlin command. An object with the same name as the graph is generated in the system during graph client initialization. You can directly use it as follows:
    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 Running Algorithms

4.9.1 PageRank

You can call a GES API to run the PageRank algorithm. The sample code is as follows:

```
 public void static executeAlgorithm(GraphClient graphClient) throws ApiException
{
// Set PageRank algorithm parameters.
PageRankParameters parameters = new PageRankParameters();
 parameters.setMaxIterations(1000);
// Create an algorithm request.
AlgorithmReq req = new AlgorithmReq();
req.setAlgorithmName(AlgorithmNames.PAGE_RANK);
 req.setParameters(parameters);
// Run the algorithm.
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
// Obtain the JobId.
String jobId = asyncAPIResp.getJobId();
 PageRankJobReq req1 = new PageRankJobReq();
 req1.setJobId(jobId);
// Query the algorithm execution result using the JobId.
JobResp<QueryData<PageRankResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```

4.9.2 PersonalRank

You can use a specific API provided by GES to run the PersonalRank algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the K-hop algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the K-core algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the Shortest Path algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the All Shortest Paths algorithm. The sample code is as follows:

```
  public void static executeAlgorithm(GraphClient graphClient) throws ApiException
{
AllShortestPathParameters parameters = new AllShortestPathParameters();
 parameters.setSource("46");
 parameters.setTarget("35");
//Algorithm request
AlgorithmReq req = new AlgorithmReq();
//Set the algorithm name.
req.setAlgorithmName(AlgorithmNames.ALL_SHORTEST_PATHS);
 req.setParameters(parameters);
//Run the algorithm.
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//Obtain the JobId.
String jobId = asyncAPIResp.getJobId();
 AllShortestPathJobReq req1 = new AllShortestPathJobReq();
 req1.setJobId(jobId);
//Query the algorithm execution result based on the JobId.
 System.out.println(graphClient.queryJobStatus(req1));
}
```

4.9.7 Filtered Shortest Path

You can use a specific API provided by GES to run the filteredShortestPath algorithm. The sample code is as follows:

```
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); // Algorithm name
algorithmReq.setParameters(filteredShortestPathParameters); // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.8 SSSP

You can use a specific API provided by GES to run the SSSP algorithm. The sample code is as follows:


```
 public void static executeAlgorithm(GraphClient graphClient) throws ApiException
 {
 SSSPParameters parameters = new SSSPParameters();
 parameters.setSource("46");
//Algorithm request
 AlgorithmReq req = new AlgorithmReq();
//Set the algorithm name.
 req.setAlgorithmName(AlgorithmNames.SSSP);
 req.setParameters(parameters);
//Run the algorithm.
 AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//Obtain the JobId.
 String jobId = asyncAPIResp.getJobId();
 SSSPJobReq req1 = new SSSPJobReq();
 req1.setJobId(jobId);
//Query the algorithm execution result based on the JobId.
 System.out.println(graphClient.queryJobStatus(req1));
 }
```

4.9.9 Shortest Path of Vertex Sets

You can use a specific API provided by GES to run the shortestPathsOfVertexSets algorithm. The sample code is as follows:

```
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);  //Algorithm
name
    algorithmReq.setAlgorithmName(AlgorithmNames.SHORTEST_PATH_OF_VERTEX_SETS);  // Algorithm
parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.10 n Paths

You can use a specific API provided by GES to run the nPaths algorithm. The sample code is as follows:

```
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);  // Algorithm name
    algorithmReq.setParameters(nPathsParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);
```



```
    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.11 Closeness

You can use a specific API provided by GES to run the Closeness algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the Label Propagation algorithm. The sample code is as follows:

```
 public void static 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 Algorithm

You can use a specific API provided by GES to run the Louvain algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the Link Prediction algorithm. The sample code is as follows:


```
 public void static 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

You can use a specific API provided by GES to run the Node2vec algorithm. The sample code is as follows:

```
 public void static 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 Real-time Recommendation

You can use a specific API provided by GES to run the Real-time Recommendation algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the Common Neighbors algorithm. The sample code is as follows:

```
 public void static 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 Connected Component

You can use a specific API provided by GES to run the Connected Component algorithm. The sample code is as follows:

```
 public void static 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 Algorithm

You can use a specific API provided by GES to run the Degree Correlation algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the Triangle Count algorithm. The sample code is as follows:

```
 public void static executeAlgorithm(GraphClient graphClient) throws ApiException
{
//Algorithm request
AlgorithmReq req = new AlgorithmReq();
req.setAlgorithmName(AlgorithmNames.TRIANGLE_COUNT);
//Run the algorithm.
AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(req);
//Obtain the JobId.
String jobId = asyncAPIResp.getJobId();
 TriangleCountJobReq req1 = new TriangleCountJobReq();
 req1.setJobId(jobId);
//Query the algorithm execution result based on the JobId.
JobResp<QueryData<TriangleCountResult>> resp = graphClient.queryJobStatus(req1);
 System.out.println(resp);
}
```


4.9.21 Cluster Coefficient

You can use a specific API provided by GES to run the Cluster Coefficient algorithm. The sample code is as follows:

```
 public void static 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

You can use a specific API provided by GES to run the filteredCircleDetection algorithm. The sample code is as follows:

```
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);  // Algorithm name
    algorithmReq.setParameters(filteredCircleDetectionParameters);  // Algorithm parameters
    algorithmReq.addFilter("out", FilterQueryType.EDGE_FILTER, "label_name","labelName", "=", "transfer",
5);

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.23 Subgraph Matching

You can use a specific API provided by GES to run the subgraphMatching algorithm. The sample code is as follows:

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

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.SUBGRAPH_MATCHING);  // Algorithm name
    algorithmReq.setParameters(subgraphMatchingParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    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

You can use a specific API provided by GES to run the filteredAllPairsShortestPaths algorithm. The sample code is as follows:

```
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);  //
Algorithm name
    algorithmReq.setParameters(filteredAllPairsShortestPathsParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.25 Filtered All Shortest Paths

You can use a specific API provided by GES to run the filteredAllShortestPaths algorithm. The sample code is as follows:

```
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);  // Algorithm
name
    algorithmReq.setParameters(filteredAllShortestPathsParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.26 TopicRank

You can use a specific API provided by GES to run the topicrank algorithm. The sample code is as follows:

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

    AlgorithmReq algorithmReq = new AlgorithmReq();
    algorithmReq.setAlgorithmName(AlgorithmNames.TOPICRANK);  // Algorithm name
```



```
    algorithmReq.setParameters(topicrankParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.9.27 Filtered n Paths

You can use a specific API provided by GES to run the filteredNPaths algorithm. The sample code is as follows:

```
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);  // Algorithm name
    algorithmReq.setParameters(filteredNPathsParameters);  // Algorithm parameters
    algorithmReq.addFilter("out", FilterQueryType.EDGE_FILTER, "label_name","labelName", "=", "transfer",
5);

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    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

You can use a specific API provided by GES to run the commonNeighborsOfVertexSets algorithm. The sample code is as follows:

```
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);  //
Algorithm name
    algorithmReq.setParameters(commonNeighborsOfVertexSetsParameters);  // Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    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

You can use a specific API provided by GES to run the allShortestPathsOfVertexSets algorithm. The sample code is as follows:

```
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);  //
Algorithm name
    algorithmReq.setAlgorithmName(AlgorithmNames.ALL_SHORTEST_PATH_OF_VERTEX_SETS);  //
Algorithm parameters

    // Execute the algorithm based on the specified parameters.
    AsyncAPIResp asyncAPIResp = graphClient.algorithmQuery(algorithmReq);

    // Query job status by job ID.
    QueryJobReq queryJobReq = new QueryJobReq();
    queryJobReq.setJobId(asyncAPIResp.getJobId());
    GesResponse gesResponse =  graphClient.queryAsyncTask(queryJobReq);
    System.out.println(gesResponse);
}
```

4.10 Querying Job Status

You can use a specific API provided by GES to query job status. The sample code is as follows:

```
  public static void queryAsyncTask(GraphClient graphClient,String jobId) throws ApiException
{
VertexQueryJobReq req = new VertexQueryJobReq();
//Set the pagination query parameters. The default value of offset is 0, and the default value of limit is
100000.
req.setOffset(0);
req.setLimit(100);
req.setJobId(jobId);
JobResp<QueryData<VertexQueryResult>> resp = graphClient.queryJobStatus(req);
System.out.println(resp);
}
```

NOTE

After asynchronous APIs such as those for querying vertices and edges or executing algorithms are used, job IDs are returned. You can use the job ID to query the execution status of each job.

4.11 Canceling a Job

You can use a specific API provided by GES to cancel a job. Sample code is as follows:

```
public static void stopAsyncJob(GraphClient graphClient,String jobId) throws ApiException
{
    Map result = graphClient.deleteJob(jobId);
    //Output the result in a specific format using the MapUtils tool class.
    System.out.print(MapUtils.map2json(result));
}
```


NOTE

After asynchronous APIs such as those for querying vertices and edges or executing algorithms are used, job IDs are returned. You can use the job ID to cancel the job.

4.12 Adding a Vertex

You can use a specific API provided by GES to add a vertex. The sample code is as follows:

```
 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();
   //Set the vertex ID.
    req.setVertexId("Lily");
    //Set the label.
    req.setLabel("user");
    //Set the properties.
    req.setProperties(properties);
    graphClient.addVertex(req);

}
```

4.13 Deleting a Vertex

You can use a specific API provided by GES to delete a vertex. The sample code is as follows:

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

4.14 Adding an Edge

You can use a specific API provided by GES to add an edge. The sample code is as follows:

```
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 Deleting an Edge

You can use a specific API provided by GES to delete an edge. The sample code is as follows:

```
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 Adding an Index

You can use a specific API provided by GES to add an index. The sample code is as follows:

```
public void excuteCreateIndex(GraphClient graphClient) throws ApiExcepti
 {
     CreateIndexReq req = new CreateIndexReq();
    //Whether a label is available. The value is case-insensitive. If this parameter is left blank, the default value false is used.
     req.setHasLabel("true");
    //Index name. It can contain only letters and digits. No default value is available.
     req.setIndexName("ageIndex");
    //Element type. Possible values are vertex and edge. The value is case-insensitive and does not have a default value.
     req.setElementType("vertex");
   // Index type. Possible values are GlobalCompositeVertexIndex and GlobalCompositeEdgeIndex. The value is case-insensitive and does not have a default value.
   //The value of indexType must correspond to that of elementType. For example, if elementType is set to vertex, indexType must be set to GlobalCompositeVertexIndex. If elementType is set to edge, indexType must be set to GlobalCompositeEdgeIndex. Otherwise, the index fails to be created.
     req.setIndexType("GlobalCompositeVertexIndex");
    //Property list of indexes (Supported property types include int, float, double, long, enum, char array, string, date.)
    //If hasLabel is false or null, this item is mandatory.
     req.setIndexProperty(req.setIndexProperty(new String[] {"age"});)
     Map<String, Object> result = graphClient.createIndex(req);
     System.out.print(MapUtils.map2json(result));
 }
```

4.17 Deleting an Index

You can use a specific API provided by GES to delete an index. The sample code is as follows:

```
public void excuteDeleteIndex(GraphClient graphClient) throws ApiException, GraphSdkException
{
   //Delete the index whose indexName is ageIndex.
    Map<String, Object> result = graphClient.deleteIndex("ageIndex");
    System.out.print(MapUtils.map2json(result));
}
```


4.18 Querying Indexes

You can use a specific API provided by GES to query an index. The sample code is as follows:

```
public void excuteQueryIndex(GraphClient graphClient) throws ApiException
{
  //Query all indexes on the graph.
    QueryIndexResult result = graphClient.queryIndex();
    System.out.print(result);
}
```

4.19 Exporting a Graph

You can use a specific API provided by GES to export graph data. The sample code is as follows:

```
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 Removing a Graph

You can use a specific API provided by GES to remove the graph data. The sample code is as follows:

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

4.21 Adding a Property

You can use a specific API provided by GES to add a property. The sample code is as follows:

```
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 Querying Path Details

You can use a specific API provided by GES to query path details. The sample code is as follows:

```
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 Incrementally Importing Data to Graphs

You can use a specific API provided by GES to incrementally import data to graphs. The sample code is as follows:

```
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);
    //Import data to graphs.
    AsyncAPIResp res = graphClient.incrementImport(importGraphReq);
    //Obtain the JobId.
    String jobId = res.getJobId();
    ImportGraphJobReq req = new ImportGraphJobReq();
    req.setJobId(jobId);
    //Query the import result based on the JobId.
    System.out.println(graphClient.queryJobStatus(req));
}
```

4.24 Adding Vertices in Batches

You can use a specific API provided by GES to add vertices in batches. The sample code is as follows:

```
public static void addBatchVertice(GraphClient graphClient) throws ApiException
{
   // Construct a vertex property list.
   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"));

    // Construct the information of a single vertex.
    AddVertexReq vertex = new AddVertexReq();
    vertex.setVertexId("180");
    vertex.setLabel("movie");
    vertex.setProperties(properties);

    // Form the information of a batch of vertices.
    List<AddVertexReq> vertices = new ArrayList<>();
    vertices.add(vertex);

    // Construct a request for adding vertices in batches.
    AddBatchVertexReq addBatchVertexReq = new AddBatchVertexReq();
    addBatchVertexReq.setVertices(vertices);

    // Execute the request for adding vertices in batches.
    Map<String, Object> result = graphClient.addBatchVertex(addBatchVertexReq);
}
```

4.25 Deleting Vertices in Batches

You can use a specific API provided by GES to delete vertices in batches. The sample code is as follows:

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

    // Construct the ID list of vertices to be deleted.
    List<String> vertices = new ArrayList<>();
    vertices.add(movieVertex);
    vertices.add(userVertex);

    // Construct a request for deleting vertices in batches.
    DeleteBatchVertexReq deleteBatchVertexReq = new DeleteBatchVertexReq();
    deleteBatchVertexReq.setVertices(vertices);

    // Execute the request for deleting vertices in batches.
    Map<String, Object> result = graphClient.deleteBatchVertex(deleteBatchVertexReq);
}
```

4.26 Adding Edges in Batches

You can use a specific API provided by GES to add edges in batches. The sample code is as follows:

```
public static void addBatchEdges(GraphClient graphClient) throws ApiException
{
    // Construct the edge information.
    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);

    // Form the edge list.
    List<Edge> edges = new ArrayList<>();
    edges.add(edge);

    // The default option is to allow repetitive edges.
```



```
    ParallelEdgeOption parallelEdgeOption = new ParallelEdgeOption();

    // Construct a request for adding edges in batches.
    AddBatchEdgeReq addBatchEdgeReq = new AddBatchEdgeReq();
    addBatchEdgeReq.setEdges(edges);
    addBatchEdgeReq.setParallelEdge(parallelEdgeOption);

    // Execute the request for adding edges in batches.
    Map<String, Object> result = graphClient.addBatchEdge(addBatchEdgeReq);
}
```

4.27 Deleting Edges in Batches

You can use a specific API provided by GES to delete edges in batches. The sample code is as follows:

```
public static void deleteBatchEdges(GraphClient graphClient) throws ApiException
{
     // Construct the information of a single edge.
    DeleteEdgeReq edge = new DeleteEdgeReq();
    edge.setSource("46");
    edge.setTarget("39");

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

    // Form the list of edges to be deleted.
    List<DeleteEdgeReq> edges = new ArrayList<>();
    edges.add(edge);
    edges.add(edgeWithIndex);

    // Construct a request for deleting edges in batches.
    DeleteBatchEdgeReq deleteBatchEdgeReq = new DeleteBatchEdgeReq();
    deleteBatchEdgeReq.setEdges(edges);

    // CExecute the request for deleting edges in batches.
    Map<String, Object> result = graphClient.deleteBatchEdge(deleteBatchEdgeReq);
}
```

4.28 Updating Vertex Properties in Batches

You can use a specific API provided by GES to update vertex properties in batches. The sample code is as follows:

```
public static void updateBatchVertice(GraphClient graphClient) throws ApiException
{
    // Construct the information of a single vertex property.
    Map<String, List<Object>> properties = new HashMap<>();
    properties.put("age", Arrays.asList(20));
    AddVertexReq vertex = new AddVertexReq();
    vertex.setVertexId("Zhang San1");
    vertex.setProperties(properties);

    Map<String, List<Object>> listProperties = new HashMap<>();
    listProperties.put("name", Arrays.asList("teat","mathematics"));
    AddVertexReq vertexWithListProperty = new AddVertexReq();
    vertexWithListProperty.setVertexId("Zhang San0");
    vertexWithListProperty.setProperties(listProperties);

    Map<String, List<Object>> setProperties = new HashMap<>();
    setProperties.put("name", Arrays.asList("a","d"));
    AddVertexReq vertexWithSetProperty = new AddVertexReq();
    vertexWithSetProperty.setVertexId("Zhang San140");
    vertexWithSetProperty.setProperties(setProperties);
```



```
    // Form the information of a batch of vertex properties.
    List<AddVertexReq> vertices = new ArrayList<>();
    vertices.add(vertex);
    vertices.add(vertexWithListProperty);
    vertices.add(vertexWithSetProperty);

    // Construct a request for updating vertex properties in batches.
    AddBatchVertexReq updateBatchVertexReq = new AddBatchVertexReq();
    updateBatchVertexReq.setVertices(vertices);

    // Execute the request for updating vertex properties in batches.
    Map<String, Object> result = graphClient.updateBatchVertex("batch-update", updateBatchVertexReq);
}
```

4.29 Updating Edge Properties in Batches

You can use a specific API provided by GES to update edge properties in batches. The sample code is as follows:

```
public static void updateBatchEdges(GraphClient graphClient) throws ApiException
{
    // Construct the edge information. Do not specify the 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);

    // Construct the edge information. Specify the index.
    EdgeWithIndex edgeWithIndex = new EdgeWithIndex();
    edgeWithIndex.setSource("46");
    edgeWithIndex.setTarget("38");
    edgeWithIndex.setIndex("0");
    edgeWithIndex.setProperties(properties);

    // Form the edge list.
    List<EdgeWithIndex> edges = new ArrayList<>();
    edges.add(edgeWithoutIndex);
    edges.add(edgeWithIndex);

    // Construct a request for updating edge properties in batches.
    UpdateBatchEdgePropertyReq updateBatchEdgeReq = new UpdateBatchEdgePropertyReq();
    updateBatchEdgeReq.setEdges(edges);

    // Execute the request for updating edge properties in batches.
    Map<String, Object> result = graphClient.updateBatchEdge("batch-update", updateBatchEdgeReq);
}
```


5 Using the Management Plane SDK

5.1 Initializing the Client of the GES Management Plane

When using the GES SDK tool to access the GES management plane, you need to initialize the GES client. Both AK/SK and token authentication modes are supported for initializing the client. The sample code is as follows:

- Sample code for AK/SK authentication

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

- Sample code for token authentication

```
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);
 GesClient client = new GesClient(AuthenticationMode.TOKEN, gesInfo);
```

5.2 Querying Quotas

GES provides a quota query API, with which you can query quotas of the current tenant, including the graph quota and graph backup quota. The sample code is as follows:

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


5.3 Verifying Metadata Files

GES provides an API for verifying metadata files, with which you can check whether the selected data sets match the metadata file. The sample code is as follows:

```
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 Querying the Graph List

You can use a specific API provided by GES to query the graph list. The sample code is as follows:

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

5.5 Querying Graph Details

You can use a specific API provided by GES to query graph details. The sample code is as follows:

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

5.6 Creating a Graph

You can use a specific API provided by GES to create a graph. The sample code is as follows:

```
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 Closing a Graph

You can use a specific API provided by GES to close a graph. The sample code is as follows:

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

5.8 Starting a Graph

You can use a specific API provided by GES to start a graph from the original graph or a backup ID. The sample code is as follows:

```
//Start from the original graph.
 private static void startGraph(GesClient client, String graphId) throws GesSdkException
  {
    Job job = client.startGraph(graphId);
    System.out.println(job);
  }
 //Start from a backup.
 private static void startGraphFromBackup(GesClient client, String graphId, String backUpId) throws
GesSdkException
  {
    Job job = client.startGraph(graphId,backUpId);
    System.out.println(job);
  }
```

5.9 Deleting a Graph

You can use a specific API provided by GES to delete a graph. The sample code is as follows:

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

5.10 Incrementally Importing Data to a Graph

You can call a GES API to incrementally import data to a graph. The sample code is as follows:

```
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);
    // Import data to graphs.
    AsyncAPIResp res = graphClient.incrementImport(importGraphReq);
    // Obtain the JobId.
    String jobId = res.getJobId();
    ImportGraphJobReq req = new ImportGraphJobReq();
    req.setJobId(jobId);
    //Query the import result based on the JobId.
    System.out.println(graphClient.queryJobStatus(req));
}
```

5.11 Exporting a Graph

You can call a GES API to export graph data. The sample code is as follows:

```
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 Clearing a Graph

You can call a GES API to clear graph data. The sample code is as follows:

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

5.13 Upgrading a Graph

You can call a GES API to delete a graph. The sample code is as follows:

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

5.14 Binding EIPs

You can call a GES API to bind an EIP to a graph. The sample code is as follows:

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



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

5.15 Unbinding an EIP

You can call a GES API to unbind an EIP from a graph. The sample code is as follows:

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

5.16 Querying Backups of All Graphs

You can use a specific API provided by GES to query backups of all graphs of the current tenant. The sample code is as follows:

```
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 Querying the Backup List of a Graph

You can use a specific API provided by GES to query backups of a graph of the current tenant. The sample code is as follows:

```
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 Adding a New Backup

You can use a specific API provided by GES to add a new backup to a graph. The sample code is as follows:


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

5.19 Deleting a Backup

You can use a specific API provided by GES to delete a backup from a graph. The sample code is as follows:

```
private static void deleteBackup(GesClient client, String graphId, String backupId) throws GesSdkException
 {
    boolean b = client.deleteBackup(graphId, backupId);
    System.out.println(b);
 }
```

5.20 Querying Job Status

You can use a specific API provided by GES to query job status. The sample code is as follows:

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

NOTE

Asynchronous APIs that are used to stop, start, restore, and delete graphs, as well as create backups will return job IDs after commands are sent. You can query the job execution status according to the job IDs.

5.21 Query Information in the Task Center

You can call a GES API to query information in the task center. The sample code is as follows:

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

NOTE

Asynchronous APIs that are used to stop, start, restore, and delete graphs, as well as create backups will return job IDs after commands are sent. You can query the job execution status using the job IDs.

6 Using Cypher JDBC Driver to Access GES

Introduction

The Cypher JDBC Driver is designed for GES. It is developed based on Neo4j JDBC Driver and provides a method of using JDBC to access GES and perform cypher queries.

The driver greatly reduces the CPU and memory usage for returning a large amount of data requested by high-concurrent cypher queries to avoid JVM caching of a complete request body. The driver parses a response body into streaming data instead obtaining an entire body and then parsing it.

Configuring Dependencies

Import the project and configure the Maven project parameters. Add the following dependency to the POM file:

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

Parameters

Table 6-1 getConnection parameters

Type	Description
url	URL of the GES Cypher API. This is the first parameter of **DriverManager.getConnection**. Prefix the value with **jdbc:ges:http(s)** so that the JDBC driver can identify the URL.
prop	**Properties** object, including configurations required for connecting to GES APIs. For details, see **Table 6-2**.

Table 6-2 Properties parameters

Type	Description
X-Auth-Token	Token obtained through IAM authentication.
parse-json	Whether to convert data to vertices and edges. **false** (default value): Vertices and edges in the return body are in map format. **true**: Vertices and edges are returned in **GesElement** format.

Example

```
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.
            // ...
        }
    }
}
```

Authentication

You can use the token or the AK/SK to complete authentication for GES Cypher JDBC Driver.

- If you use the token, refer to **Using Parameters** and **Usage Example** to complete the authentication.
- For AK/SK authentication, you need to specify the AK/SK using the GES service plane SDK.

  For details about how to import the service plane SDK dependency, see **Importing a Project**. For details about how to configure GraphInfo, see **Initializing the Client of the GES Service Plane**. You need to enter the obtained AK/SK and the region name.

  The following is an example for AK/SK authentication:

```
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() {
        // Initialize the GraphInfo object in a way it is required (this example is just for reference).
        GraphInfo info = getGraphInfoByYourSelf();
       // Specify the AK/SK.
        info.setAccessKey(accessKey);
        info.setSecretKey(secretKey);
      // Specify the region name.
        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 Relationships Between SDKs and REST APIs

Table 7-1 Relationships between SDKs and REST APIs

Interface	Method	API
Graph Service 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

