

Data Lake Insight

SDK Reference

Issue 01
Date 2023-09-16



Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI 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.

Contents

1 Introduction to DLI SDKs.....	1
1.1 What Is DLI SDK.....	1
1.2 Content Navigation.....	1
2 Preparing the SDK Environment.....	3
2.1 Prerequisites.....	3
2.2 Configuring the Java Environment.....	7
2.3 Configuring the Python Environment.....	9
3 Mapping Between DLI SDKs and APIs.....	12
4 Java SDK.....	18
4.1 Instructions.....	18
4.2 Initializing the DLI Client.....	18
4.3 OBS Authorization.....	19
4.4 Queue-Related SDKs.....	20
4.5 Resource-Related SDKs.....	21
4.6 SDKs Related to SQL Jobs.....	22
4.6.1 Database-Related SDKs.....	22
4.6.2 Table-Related SDKs.....	23
4.6.3 Job-related SDKs.....	26
4.7 SDKs Related to Flink Jobs.....	31
4.8 SDKs Related to Spark Jobs.....	34
4.9 SDKs Related to Flink Job Templates.....	36
5 Python SDK.....	38
5.1 Instructions.....	38
5.2 Initializing the DLI Client.....	38
5.3 Queue-Related SDKs.....	39
5.4 Resource-Related SDKs.....	40
5.5 SDKs Related to SQL Jobs.....	41
5.5.1 Database-Related SDKs.....	41
5.5.2 Table-Related SDKs.....	42
5.5.3 Job-related SDKs.....	44
5.6 SDKs Related to Spark Jobs.....	46

A Change History..... 48

1 Introduction to DLI SDKs

1.1 What Is DLI SDK

Introduction to DLI

Data Lake Insight (DLI) is a serverless data processing and analysis service fully compatible with [Apache Spark](#), [Trino](#), and [Apache Flink](#) ecosystems. It frees you from managing any server. DLI supports standard SQL and is compatible with Spark and Flink SQL. It also supports multiple access modes, and is compatible with mainstream data formats. DLI supports SQL statements and Spark applications for heterogeneous data sources, including CloudTable, RDS, DWS, CSS, OBS, custom databases on ECSs, and offline databases.

Introduction to DLI SDKs

Data Lake Insight Software Development Kit (DLI SDK) is the encapsulation of RESTful APIs provided by DLI to simplify user development. Users can directly call API functions provided by DLI SDK to use DLI.

 NOTE

The DLI SDK calls APIs using HTTPS. The certificate for using the server is provided.

1.2 Content Navigation

This development guide describes how to install and configure an environment and call functions provided by DLI SDK for secondary development.

Chapter	Content
Overview	This topic describes the concepts of DLI and DLI SDKs.

Chapter		Content
Preparing the Environment	Prerequisites	This topic describes the configuration process of the running environment of the DLI SDKs.
	Configuring the Java Environment	
	Configuring the Python Environment	
Mapping Between DLI SDK and APIs	Mapping Between DLI SDK and APIs	This topic describes the DLI SDKs currently under development, along with its corresponding APIs.
Java SDK	Java SDK	This topic describes the DLI Java SDKs.
Python SDK	Python SDK	This topic describes the DLI Python SDKs.

2 Preparing the SDK Environment

2.1 Prerequisites

Registering a Huawei ID and Enabling Huawei Cloud Services

1. Log in to the Huawei Cloud website.
2. Click **Register** at the upper right corner of the page.
3. Complete the registration as instructed.
Select the agreement as prompted to enable Huawei Cloud services.

Enabling DLI

1. Log in to the management console and click **Service List** from the top menu.
2. Click **Data Lake Insight** under **Enterprise Intelligence**.
3. Click **Access Console** to switch to the DLI management console.

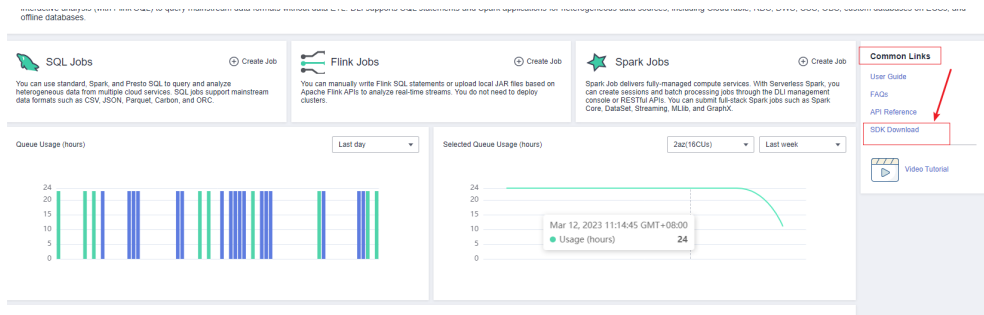
Downloading SDK

1. Log in to the DLI management console.
2. On the **Overview** page, click **SDK Download** under **Helpful Links**.

NOTE

You are advised to download the .sha256 file corresponding to the SDK installation package. This file can be used to verify the integrity of the package.

Figure 2-1 Downloading the SDK installation package



3. On the displayed **DLI SDK DOWNLOAD** page, click the target link to obtain the desired SDK installation package.

Currently, DLI provides the SDK installation packages of the Java and Python versions.

- Java SDK

Obtain the **dli-sdk-java-x.x.x.zip** package and decompress it. The following table shows the directory structure of the package.

Table 2-1 Directory structure

Parameter	Description
jars	SDK and its dependent JAR packages.
maven-install	Script and JAR package that are installed in the local Maven repository.
dli-sdk-java.version	Java SDK version description.

If Maven is used, add the following Maven configuration items on which **huaweicloud-dli-sdk-java** depends:

```
<dependency>
  <groupId>com.huawei.dli</groupId>
  <artifactId>huaweicloud-dli-sdk-java</artifactId>
  <version>x.x.x</version>
</dependency>
```

NOTE

You can obtain the latest version number from the [Maven configuration](#).

DLI depends on SDKs (for example, OBS SDK), which can be downloaded by configuring the Maven image source repository of Huawei Cloud.

- Use the Huawei image source as the main repository:

For details about how to configure the Huawei Maven image source, visit Huawei open source image site, select Huawei SDK, and click HuaweiCloud SDK.

If you build a project with Maven, modify the **settings.xml** file by adding the following content:

- 1) Add the following content to the **profiles** node:

```
<profile>
  <id>MyProfile</id>
  <repositories>
    <repository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</pluginRepositories>
```



```
<pluginRepository>
  <id>HuaweiCloudSDK</id>
  <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
```

2) Add the following information to the **mirrors** node:

```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*,!HuaweiCloudSDK</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven</url>
</mirror>
```

3) Add the **activeProfiles** tag to activate the configurations.

```
<activeProfiles>
  <activeProfile>MyProfile</activeProfile>
</activeProfiles>
```

- Use a non-Huawei image source as the main repository (for example, a user-defined image source) to use HuaweiCloud SDK: If you build a project with Maven, modify the **settings.xml** file as follows:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://
maven.apache.org/xsd/settings-1.0.0.xsd">
  <servers>
    <server>
      <id>demo-releases</id>
      <username>deployment</username>
      <password><![CDATA[xxx]]></password>
    </server>
  </servers>
  <mirrors>
    <mirror>
      <id>demo-releases</id>
      <mirrorOf>*,!HuaweiCloudSDK</mirrorOf>
      <url>http://maven.demo.com:8082/demo/content/groups/public</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>demo</id>
      <activation>
        <activeByDefault>true</activeByDefault>
        <jdk>1.8</jdk>
      </activation>
      <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
      </properties>
      <repositories>
        <repository>
          <id>demo-releases</id>
          <url>http://demo-releases</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>
```

```
</repository>
</repositories>
<pluginRepositories>
  <pluginRepository>
    <id>demo-releases</id>
    <url>http://demo-releases</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>
</profile>
<profile>
  <id>huaweicloudrepo</id>
  <repositories>
    <repository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
</profile>
</profiles>
<activeProfiles>
  <activeProfile>demo</activeProfile>
  <activeProfile>huaweicloudrepo</activeProfile>
</activeProfiles>
</settings>
```

– Python SDK

Obtain the **dli-sdk-python-x.x.x.zip** package and decompress it. The following table shows the directory structure of the package.

Table 2-2 Directory structure

Parameter	Description
dli	DLI Python SDK package.
examples	Example of using the SDK.
pyDLI	Connector for Python to connect to DLI SQL, which is implemented based on pyHive.
setup.py	SDK installation script.

2.2 Configuring the Java Environment

Scenario

Table 2-3 describes the environment required for secondary development.

Table 2-3 Development environment

Item	Description
Installing the Operating System (OS)	Windows OS. Windows 7 or later is recommended.
Installing the JDK	Basic configurations of the development environment. JDK 1.7 or 1.8 is required. You are advised to use JDK 1.8 for better compatibility of later versions.
Eclipse installation and configuration	It is a tool used to develop DLI applications.

Procedure

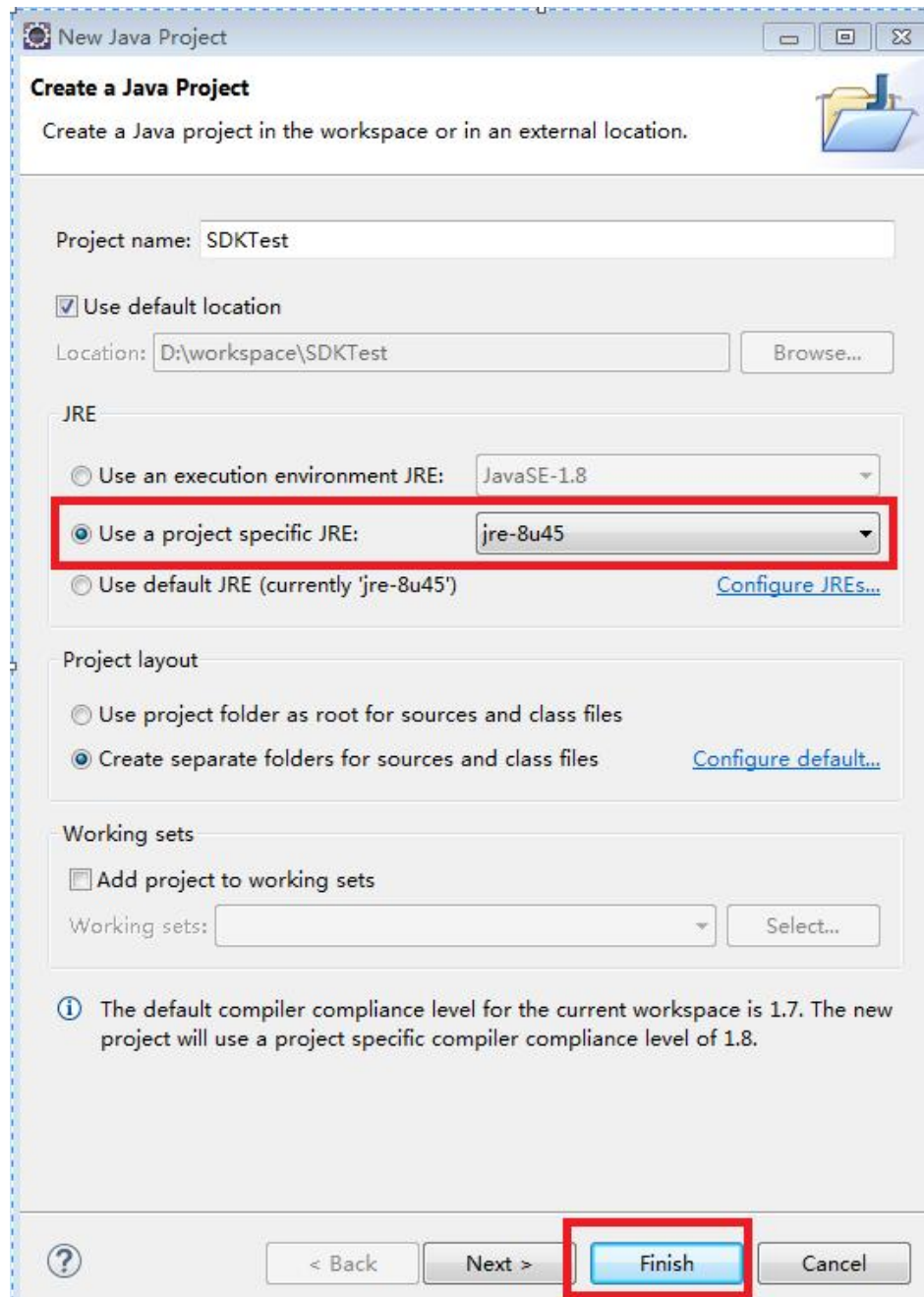
- Step 1** Download JDK of 1.8 or a later version from the [Oracle's official website](#), install the JDK, and configure Java environment variables.
1. Install the JDK.
 2. Set environment variables. Specifically, choose **Start > Control Panel**. Click **System > Advanced system settings**. In the displayed **System Properties** dialog box, click **Advanced** and then click **Environment Variables** to switch to the **Environment Variables** dialog box.
 3. Create the system variable **JAVA_HOME** with **Value** set to the JDK installation path, for example, **D:\Java\jdk1.8.0_45**.
 4. Edit the **Path** variable and add **%JAVA_HOME%\bin** to **Variable value**.
 5. Create the **CLASSPATH** variable with **Variable value** set to **.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar**.
 6. Check whether the configurations succeed. Specifically, at the Start menu, enter **cmd** and press **Enter** to bring up the command prompt window. Enter **java -version** and press **Enter**. If the version information is displayed, as shown in **Figure 2-2**, the JDK is successfully installed and environment variables are successfully configured.

Figure 2-2 Checking the configuration

```
C:\Users\gwx419194>java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b15)
Java HotSpot(TM) Client VM (build 25.45-b02, mixed mode, sharing)
```

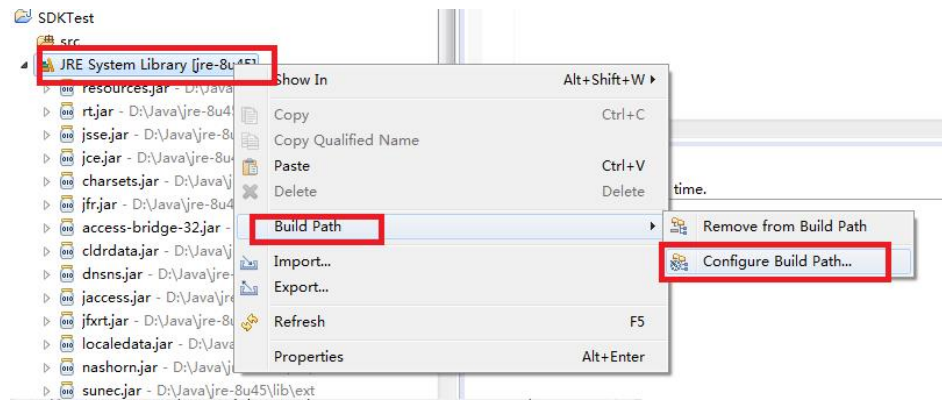
- Step 2** Download Eclipse IDE for Java Developers of the latest version from the [Eclipse's official website](#) and install it. Configure the JDK in Eclipse.
1. Create a project and select the correct JRE version. For details, see [Figure 2-3](#).

Figure 2-3 Creating a project



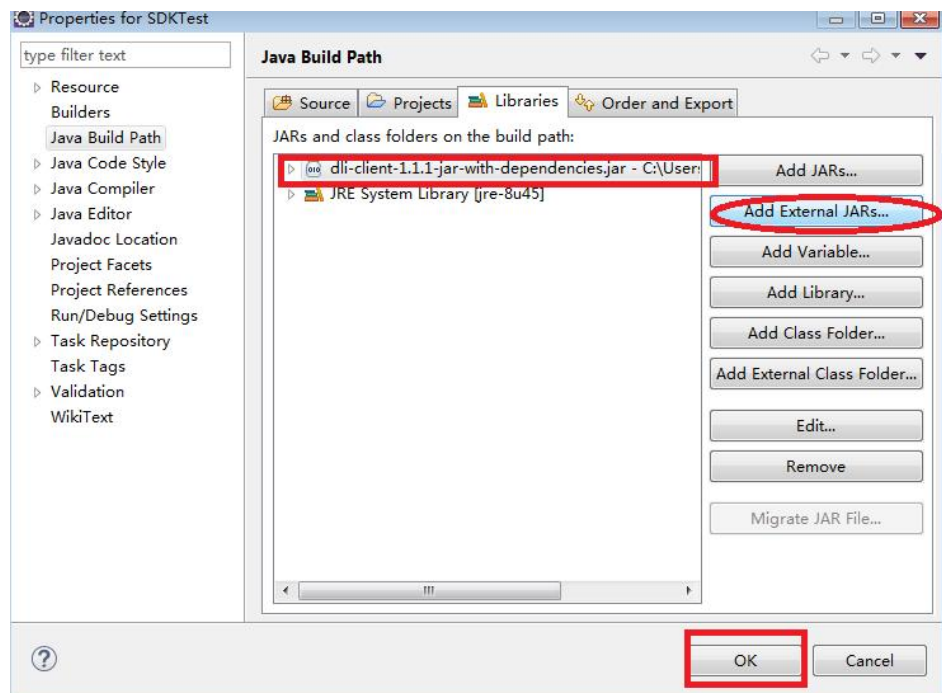
- Step 3** Configure and import the SDK JAR package.
1. Right-click **JRE System Library** and choose **Build Path > Configure Build Path** from the shortcut menu. For details, see [Figure 2-4](#).

Figure 2-4 Configuring the project path



2. Click **Add External JARs**, select the downloaded JAR package obtained in **Downloading SDK**, and click **OK**.

Figure 2-5 Selecting the SDK JAR package



----End

2.3 Configuring the Python Environment

Scenario

If you are preparing for secondary development, you need to deploy the environment as required in [Table 2-4](#).

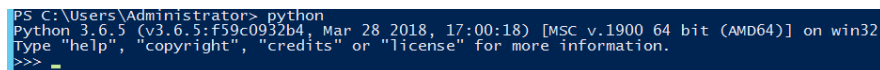
Table 2-4 Development environment

Item	Description
Operating system (OS)	Windows OS. Windows 7 or later is recommended.
Python	Python 2.7.10, 3.4.0, or later versions are required. The Visual C++ compilation environment is required, and Visual C++ build tools or Visual Studio must be installed.
Python dependencies	The DLI Python SDK dependencies include urllib3 1.15 or later, six 1.10 or later, certifi, and python-dateutil.

Procedure

Step 1 Download Python from [Python's official website](#) and install it.

1. Install Python according to the Python official guidelines.
2. Check whether the configurations succeed. Specifically, at the **Start** menu, enter **cmd** and press **Enter** to bring up the command prompt window. Enter **python** and press **Enter**. If the version information is displayed, as shown in [Figure 2-6](#), Python is successfully installed and configured.

Figure 2-6 Checking the configuration

```
PS C:\Users\Administrator> python
Python 3.6.5 (v2.6.5: f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

NOTE

If an error similar to "error: Microsoft Visual C++ xx.x is required. Get it with Build Tools for Visual Studio" is reported when you install the Python application package, the C++ compiler may not be installed. Install the Visual Studio compiler of the required version. For some OSs, restart the system after Visual Studio is installed

Step 2 Install DLI Python SDK.

1. Select the package of the [Downloading SDK](#) and decompress the installation package.
Decompress **dli-sdk-python-<version>.zip** to a local directory that can be adjusted.
2. Install the SDK.
 - a. On a PC running the Windows OS, choose **Start > Run**, enter **cmd**, and press **Enter**.
 - b. In the command-line interface (CLI) window, access the **windows** directory in the directory where **dli-sdk-python-<version>.zip** is decompressed. For example, **D:\tmp\dli-sdk-python-1.0.8**.
 - c. Run the following command to install DLI Python SDK (During the installation, the third-party dependencies are automatically downloaded):

python setup.py install

Figure 2-7 shows the installation result.

Figure 2-7 Installing Python SDK

```
D:\tmp\qli-sdk-python-1.0.8>python setup.py install
running install
running bdist_egg
running egg_info
creating ql_i_sdk_python.egg-info
writing ql_i_sdk_python.egg-info\PKG-INFO
writing dependency_links to ql_i_sdk_python.egg-info\dependency_links.txt
writing requirements to ql_i_sdk_python.egg-info\requires.txt
writing top-level names to ql_i_sdk_python.egg-info\top_level.txt
writing manifest file 'ql_i_sdk_python.egg-info\SOURCES.txt'
reading manifest file 'ql_i_sdk_python.egg-info\SOURCES.txt'
writing manifest file 'ql_i_sdk_python.egg-info\SOURCES.txt'
installing library code to build\bdist.win-amd64\egg
```

----End

3 Mapping Between DLI SDKs and APIs

OBS Authorization SDKs

Table 3-1 Mapping between OBS authorization APIs and SDKs

Class	Method	Java Method	Python Method	API
Authorize	OBS authorization	authorizeBucket	-	POST /v1.0/{project_id}/dli/obs-authorize

Queue-related SDKs

Table 3-2 Mapping between queue-related APIs and SDKs

Class	Method	Java Method	Python Method	API
Queue	Creating a Queue	createQueue	-	POST /v1.0/{project_id}/queues
	Deleting a Queue	deleteQueue	-	DELETE /v1.0/{project_id}/queues/{queue_name}
	Obtaining the Default Queue	getDefaultQueue	-	-
	Querying All Queues	listAllQueues	list_queues	GET/v1.0/{project_id}/queues

Resource-related SDKs

Table 3-3 Mapping between resource-related APIs and SDKs

Class	Method	Java Method	Python Method	API
package Resources	Uploading a Resource Package	uploadResources	upload_resource	POST /v2.0/{project_id}/resources
	Deleting a Resource Package	deleteResource	delete_resource	DELETE /v2.0/{project_id}/resources/{resource_name}
	Querying All Resource Packages	listAllResources	list_resources	GET /v2.0/{project_id}/resources
	Querying a Specified Resource Package	getResource	get_package_resource	GET /v2.0/{project_id}/resources/{resource_name}

SDKs Related to SQL Jobs

Table 3-4 Mapping between SQL job APIs and SDKs

Class	Method	Java Method	Python Method	API
Database	Creating a Database	createDatabase	create_database	POST /v1.0/{project_id}/databases
	Deleting a Database	deleteDatabase	delete_database	DELETE /v1.0/{project_id}/databases/{database_name}
	Querying All Databases	listAllDatabases	list_databases	GET /v1.0/{project_id}/databases
	Modifying a Database User	-	-	PUT /v1.0/{project_id}/databases/{database_name}/owner
Table	Creating a DLI Table	createDLITable	create_dli_table	POST /v1.0/{project_id}/databases/{database_name}/tables
	Creating an OBS Table	createObsTable	create_obs_table	POST /v1.0/{project_id}/databases/{database_name}/tables

Class	Method	Java Method	Python Method	API
	Deleting a Table	deleteTable	delete_table	DELETE /v1.0/{project_id}/databases/{database_name}/tables/{table_name}
	Querying All Tables	listAllTables	list_tables	GET /v1.0/{project_id}/databases/{database_name}/tables?keyword=tb&with-detail=true
	Describing Table Information	getTableDetail	get_table_schema	GET /v1.0/{project_id}/databases/{database_name}/tables/{table_name}
	Previewing a Table	-	-	GET /v1.0/{project_id}/databases/{database_name}/tables/{table_name}/preview
	Modifying a Table User	-	-	PUT /v1.0/{project_id}/databases/{database_name}/tables/{table_name}/owner
Job	Importing Data	submit	import_table	POST /v1.0/{project_id}/jobs/import-table
	Exporting Data	submit	export_table	POST /v1.0/{project_id}/jobs/export-table
	Submitting a Job	submit	execute_sql	POST /v1.0/{project_id}/jobs/submit-job
	Canceling a Job	cancelJob	-	DELETE /v1.0/{project_id}/jobs/{job_id}
	Querying All Jobs	listAllJobs	-	GET /v1.0/{project_id}/jobs?page-size={size}¤t-page={page_number}&start={start_time}&end={end_time}&job-type={QUERY}&queue_name={test}&order={duration_desc}
	Querying Job Results	queryJobResultInfo	-	GET /v1.0/{project_id}/jobs/{job_id}?page-size={size}¤t-page={page_number}

Class	Method	Java Method	Python Method	API
	Querying Job Status	-	-	GET /v1.0/{project_id}/jobs/{job_id}/status
	Querying Job Details	-	-	GET /v1.0/{project_id}/jobs/{job_id}/detail
	Querying Jobs of the SQL Type	listSQLJobs	-	-
	Checking the SQL Syntax	-	-	POST /v1.0/{project_id}/jobs/check-sql
	Exporting Search Results	-	-	POST /v1.0/{project_id}/jobs/{job_id}/export-result

SDKs Related to Flink Jobs

Table 3-5 Mapping between Java and Python SDKs and APIs

Class	Method	Java Method	Python Method	API
Job	Creating a Flink SQL Job	submitFlinkSqlJob	-	POST /v1.0/{project_id}/streaming/sql-jobs
	Creating a Custom Flink Job	createFlinkJarJob	-	POST /v1.0/{project_id}/streaming/flink-jobs
	Updating a Flink SQL Job	updateFlinkSqlJob	-	PUT /v1.0/{project_id}/streaming/sql-jobs/{job_id}
	Updating a Custom Flink Job	updateFlinkJarJob	-	PUT /v1.0/{project_id}/streaming/flink-jobs/{job_id}
	Querying the List of Jobs	getFlinkJobs	-	GET /v1.0/{project_id}/streaming/jobs
	Querying Flink Job Details	getFlinkJobDetail	-	GET /v1.0/{project_id}/streaming/jobs/{job_id}

Class	Method	Java Method	Python Method	API
	Querying the Flink Job Execution Plan Diagram	getFlinkJobExecuteGraph	-	GET /v1.0/{project_id}/streaming/jobs/{job_id}/execute-graph
	Querying Flink Job Monitoring Information	getFlinkJobsMetrics	-	POST /v1.0/{project_id}/streaming/jobs/metrics
	Querying the APIG Address of a Flink Job	getFlinkApigSinks	-	GET /v1.0/{project_id}/streaming/jobs/{job_id}/apig-sinks
	Running a Flink Job	runFlinkJob	-	POST /v1.0/{project_id}/streaming/jobs/run
	Stopping a Flink Job	stopFlinkJob	-	POST /v1.0/{project_id}/streaming/jobs/stop
	Deleting Flink Jobs in Batches	deleteFlinkJobsInBatch	-	POST /v1.0/{project_id}/streaming/jobs/delete

SDKs Related to Spark Jobs

Table 3-6 Mapping between Spark job APIs and SDKs

Class	Method	Java Method	Python Method	API
BatchJob	Submitting Batch Jobs	asyncSubmit	submit_spark_batch_job	POST /v2.0/{project_id}/batches
	Deleting Batch Jobs	deleteBatchJob	del_spark_batch_job	DELETE /v2.0/{project_id}/batches/{batch_id}
	Querying All Batch Jobs	listAllBatchJobs	-	GET /v2.0/{project_id}/batches
	Querying Batch Job Details	-	-	GET /v2.0/{project_id}/batches/{batch_id}

Class	Method	Java Method	Python Method	API
	Querying a Batch Job Status	getStateBatchJob	-	GET /v2.0/{project_id}/batches/{batch_id}/state
	Querying Batch Job Logs	getBatchJobLog	-	GET /v2.0/{project_id}/batches/{batch_id}/log

SDKs Related to Flink Job Templates

Table 3-7 Mapping between Java and Python SDKs and APIs

Class	Java Method	Python Method	API
Template	createFlinkJobTemplate	-	POST /v1.0/{project_id}/streaming/job-templates
	updateFlinkJobTemplate	-	PUT /v1.0/{project_id}/streaming/job-templates/{template_id}
	deleteFlinkJobTemplate	-	DELETE /v1.0/{project_id}/streaming/job-templates/{template_id}
	getFlinkJobTemplates	-	GET /v1.0/{project_id}/streaming/job-templates

4 Java SDK

4.1 Instructions

- Refer to [Downloading SDK](#), obtain the **dli-sdk-java-x.x.x.zip** package, and download the SDK file.
- Refer to [Configuring the Java Environment](#) and configure the Java development environment.

4.2 Initializing the DLI Client

To use DLI SDK to access DLI, you need to initialize the DLI client. During DLI client initialization, you can perform authentication using the Access Key ID/Secret Access Key (AK/SK) or token. The sample code is as follows:

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Sample code for AK/SK authentication

```
String ak = "ak";  
String sk = "sk";  
String regionName = "regionname";  
String projectId = "project_id";  
DLIInfo dliInfo = new DLIInfo(regionName, ak, sk, projectId);  
DLIClient client = new DLIClient(AuthenticationMode.AKSK, dliInfo);
```

 NOTE

You can perform the following operations to obtain the **Access Keys**, **project ID**, and **Region**:

1. Log in to the management console.
2. Hover the cursor on the username in the upper right corner and select **My Credentials** from the drop-down list.
3. In the navigation pane on the left, choose **Access Keys** and click **Create Access Key**. Confirm that you want to proceed with the operation and click **OK**.
4. On the displayed page, click **Download**. Open the credential file to obtain the AK/SK information.
5. In the navigation pane on the left, choose **API Credentials**. In the **Projects** pane, locate **project_id** and obtain the region information.

Sample code for token-based authentication

```
String domainName = "domainname";
String userName = "username";
String password = "password";
String regionName = "regionname";
String projectId = "project_id";
DLIInfo dliInfo = new DLIInfo(regionName, domainName, userName, password, projectId);
DLIClient client = new DLIClient(AuthenticationMode.TOKEN, dliInfo);
```

 NOTE

You can change the endpoint in **set** mode. Run the following statement:
dliInfo.setServerEndpoint(endpoint).

4.3 OBS Authorization

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Example Code

You can use the API to grant operation rights on OBS buckets to DLI to save data and run logs of users' jobs. The example code is as follows:

```
private static void authorizeBucket(DLIClient client) throws DLIException {
    String bucketName = "obs_name";
    ObsBuckets obsBuckets = new ObsBuckets();
    obsBuckets.addObsBucketsItem(bucketName);
    GlobalResponse res = client.authorizeBucket(obsBuckets);
    System.out.println(res);
}
```

4.4 Queue-Related SDKs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Creating a Queue

You can use the API provided by DLI to create a queue. The example code is as follows:

```
private static void createQueue(DLIClient client) throws DLIException {
    //Call the createQueue method of the DLIClient object to create a queue.
    String qName = "queueName";
    int cu = 16;
    String description = "test for sdk";
    Queue queue = client.createQueue(qName, cu, mode, description);
    System.out.println("----- createQueue success -----");
}
```

Deleting a Queue

You can use the API provided by DLI to delete the queue. The example code is as follows:

```
private static void deleteQueue(DLIClient client) throws DLIException {
    //Call the getQueue(queueName) method of the DLIClient object to obtain queue queueName.
    String qName = "queueName";
    Queue queue = client.getQueue(qName);
    //Call the deleteQueue() method to delete queue queueName.
    queue.deleteQueue();
}
```

Obtaining the Default Queue

DLI provides an API for querying the default queue. You can use the default queue to submit jobs. The example code is as follows:

```
private static void getDefaultQueue(DLIClient client) throws DLIException{
    //Call the getDefaultQueue method of the DLIClient object to query the default queue.
    Queue queue = client.getDefaultQueue();
    System.out.println("defaultQueue is:"+ queue.getQueueName());
}
```

NOTE

All users can use the default queue. However, DLI limits the maximum number of times a user can use the default queue.

Querying All Queues

You can use the API provided by DLI to query the queue list and select the corresponding queue to execute the job. The example code is as follows:

```
private static void listAllQueues(DLIClient client) throws DLIException {
    System.out.println("list all queues...");
}
```



```
//Call the listAllQueues method of the DLIClient object to query the queue list.
List<Queue> queues = client.listAllQueues();
for (Queue queue : queues) {
    System.out.println("Queue name:" + queue.getQueueName() + " " + "cu:" + queue.getCuCount());
}
}
```

4.5 Resource-Related SDKs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Uploading a Resource Package

You can use the interface provided by the DLI to upload resource packages. The code example is as follows:

```
private static void uploadResources(DLIClient client) throws DLException {
    String kind = "jar";
    String[] paths = new String[1];
    paths[0] = "https://bucketname.obs.com/jarname.jar";
    String description = "test for sdk";
    //Call the uploadResources method of the DLIClient object to upload resources.
    List<PackageResource> packageResources = client.uploadResources(kind, paths, description);
    System.out.println("----- uploadResources success -----");
}
```

NOTE

The following describes the request parameters. For details, see [Instructions](#).

- **kind**: resource package type. The options are as follows:
 - **jar**: JAR file
 - **Pyfile**: User Python file
 - **file**: User file
 - **modelfile**: User AI model file
- **paths**: OBS path of the resource package. The parameter format is {bucketName}.{obs domain name}/{jarPath}/{jarName}.
- **description**: description of the resource package

Querying All Resource Packages

You can use the API provided by DLI to query the list of uploaded resources. Sample code is as follows:

```
private static void listAllResources(DLIClient client) throws DLException {
    System.out.println("list all resources...");
    //Call the listAllResources method of the DLIClient object to query the queue resource list.
    Resources resources = client.listAllResources();
    for (PackageResource packageResource : resources.getPackageResources()) {
        System.out.println("Package resource name:" + packageResource.getResourceName());
    }
    for (ModuleResource moduleResource : resources.getModuleResources()) {
        System.out.println("Module resource name:" + moduleResource.getModuleName());
    }
}
```

Querying a Specified Resource Package

You can call an API to query information about the specified resource package. The sample code is as follows:

```
private static void getResource(DLIClient client) throws DLIException {
    String resourceName = "xxxxx";
    // group: If the resource package is not in the group, leave this parameter empty.
    String group= "xxxxxx";
    // Call getResource on the DLIClient object to query a specified resource package.
    PackageResource packageResource=client.getResource(resourceName,group);
    System.out.println(packageResource);
}
```

Deleting a Resource Package

You can call an API to delete an uploaded resource package. Sample code is as follows:

```
private static void deleteResource(DLIClient client) throws DLIException {
    String resourceName = "xxxxx";
    // group: If the resource package is not in the group, leave this parameter empty.
    String group= "xxxxxx";
    //Call deleteResource on the DLIClient object to upload resources.
    client.deleteResource(resourceName,group);
    System.out.println("----- deleteResource success -----");
}
```

4.6 SDKs Related to SQL Jobs

4.6.1 Database-Related SDKs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#) and created queues by following the instructions provided in [Queue-Related SDKs](#).

Creating a Database

DLI provides an API for creating a database. You can use the API to create a database. The sample code is as follows:

```
private static Database createDatabase(DLIClient client) throws DLIException {
    //Call the createDatabase method of the DLIClient object to create a database.
    String dbName = "databasename";
    Database database = client.createDatabase(dbName);
    System.out.println("create database:" + database);
    return database;
}
```

NOTE

The **default** database is a built-in database. You are not allowed to create a database named **default**.

Deleting a Database

DLI provides an API for deleting a database. The example code is as follows:

```
//Call the deleteDatabase interface of the Database object to delete a database.
//Call the getDatabase(String databaseName) interface of the DLIClient object to obtain the Database
object.
private static void deletedatabase(Database database) throws DLIException {
    String dbName = "databasename";
    database=client.getDatabase(dbName);
    database.deleteDatabase();
    System.out.println("delete db " + dbName);
}
```

NOTE

- A database that contains tables cannot be deleted. To delete a database that contains tables, delete the tables first.
- A deleted database cannot be restored. Therefore, exercise caution when deleting a database.

Querying All Databases

You can use the API provided by DLI to query the list of created databases. The example code is as follows:

```
private static void listDatabases(DLIClient client) throws DLIException {
    //Call the listAllDatabases method of the DLIClient object to query the database list.
    List<Database> databases = client.listAllDatabases();
    for (Database db : databases) {
        System.out.println("dbName:" + db.getDatabaseName() + " " + "tableCount:" + db.getTableCount());
    }
}
```

4.6.2 Table-Related SDKs

Creating a DLI Table

DLI provides an API for creating DLI tables. You can use it to create a table for storing DLI data. Sample code is as follows:

```
private static Table createDLITable(Database database) throws DLIException {
    // Construct a table column set and instantiate the Column object to construct columns.
    List<Column> columns = new ArrayList<Column>();
    Column c1 = new Column("c1", DataType.STRING, "desc for c1");
    Column c2 = new Column("c2", DataType.INT, "desc for c2");
    Column c3 = new Column("c3", DataType.DOUBLE, "desc for c3");
    Column c4 = new Column("c4", DataType.BIGINT, "desc for c4");
    Column c5 = new Column("c5", DataType.SHORT, "desc for c5");
    Column c6 = new Column("c6", DataType.LONG, "desc for c6");
    Column c7 = new Column("c7", DataType.SMALLINT, "desc for c7");
    Column c8 = new Column("c8", DataType.BOOLEAN, "desc for c8");
    Column c9 = new Column("c9", DataType.DATE, "desc for c9");
    Column c10 = new Column("c10", DataType.TIMESTAMP, "desc for c10");
    Column c11 = new Column("c11", DataType.DECIMAL, "desc for c11");
    columns.add(c1);
    columns.add(c2);
    columns.add(c3);
    columns.add(c4);
    columns.add(c5);
    columns.add(c6);
    columns.add(c7);
    columns.add(c8);
    columns.add(c9);
}
```

```
columns.add(c10);
columns.add(c11);

List<String> sortColumns = new ArrayList<String>();
sortColumns.add("c1");
String DLITblName = "tablename";
String desc = "desc for table";
// Call the createDLITable method of the Database object to create a DLI table.
Table table = database.createDLITable(DLITblName, desc, columns, sortColumns);
System.out.println(table);
return table;
}
```

NOTE

The default precision of **DataType.DECIMAL** is **(10,0)**. To set the precision of data of the decimal type, perform the following operation:

```
Column c11 = new Column("c11", new DecimalTypeInfo(25,5), "test for c11");
```

Creating an OBS Table

DLI provides an API for creating OBS tables. You can use it to create a table for storing OBS data. The example code is as follows:

```
private static Table createObsTable(Database database) throws DLIException {
    // Construct a table column set and instantiate the Column object to construct columns.
    List<Column> columns = new ArrayList<Column> ();
    Column c1 = new Column("c1", DataType.STRING, "desc for c1");
    Column c2 = new Column("c2", DataType.INT, "desc for c2");
    Column c3 = new Column("c3", DataType.DOUBLE, "desc for c3");
    Column c4 = new Column("c4", DataType.BIGINT, "desc for c4");
    Column c5 = new Column("c5", DataType.SHORT, "desc for c5");
    Column c6 = new Column("c6", DataType.LONG, "desc for c6");
    Column c7 = new Column("c7", DataType.SMALLINT, "desc for c7");
    Column c8 = new Column("c8", DataType.BOOLEAN, "desc for c8");
    Column c9 = new Column("c9", DataType.DATE, "desc for c9");
    Column c10 = new Column("c10", DataType.TIMESTAMP, "desc for c10");
    Column c11 = new Column("c11", DataType.DECIMAL, "desc for c11");
    columns.add(c1);
    columns.add(c2);
    columns.add(c3);
    columns.add(c4);
    columns.add(c5);
    columns.add(c6);
    columns.add(c7);
    columns.add(c8);
    columns.add(c9);
    columns.add(c10);
    columns.add(c11);
    CsvFormatInfo formatInfo = new CsvFormatInfo();
    formatInfo.setWithColumnHeader(true);
    formatInfo.setDelimiter(",");
    formatInfo.setQuoteChar("\"");
    formatInfo.setEscapeChar("\\");
    formatInfo.setDateFormat("yyyy/MM/dd");
    formatInfo.setTimestampFormat("yyyy-MM-dd HH:mm:ss");
    String obsTblName = "tablename";
    String desc = "desc for table";
    String dataPath = "OBS path";
    // Call the createObsTable method of the Database object to create an OBS table.
    Table table = database.createObsTable(obsTblName, desc, columns, StorageType.CSV, dataPath,
formatInfo);
    System.out.println(table);
    return table;
}
```

 NOTE

The default precision of **DataType.DECIMAL** is **(10,0)**. To set the precision of data of the decimal type, perform the following operation:

```
Column c11 = new Column("c11", new DecimalTypeInfo(25,5), "test for c11");
```

Deleting a Table

DLI provides an API for deleting tables. You can use it to delete all the tables in a database. The example code is as follows:

```
private static void deleteTables(Database database) throws DLIException {  
    // Call the listAllTables interface of the Database object to query all tables.  
    List<Table> tables = database.listAllTables();  
    for (Table table : tables) {  
        // Traverse tables and call the deleteTable interface of the Table object to delete tables.  
        table.deleteTable();  
        System.out.println("delete table " + table.getTableName());  
    }  
}
```

 NOTE

A deleted table cannot be restored. Therefore, exercise caution when deleting a table.

Querying All Tables

DLI provides an API for querying tables. You can use it to query all tables in a database. The example code is as follows:

```
private static void listTables(Database database) throws DLIException {  
    // Call the listAllTables method of the Database object to query all tables in a database.  
    List<Table> tables = database.listAllTables(true);  
    for (Table table : tables) {  
        System.out.println(table);  
    }  
}
```

Querying the Partition Information of a Table (Including the Creation and Modification Time of the Partition).

DLI provides an API for querying table partition information. You can use it to query the partition information (including the creation time and modification time) in the database table. The example code is as follows:

```
private static void showPartitionsInfo(DLIClient client) throws DLIException {  
    String databaseName = "databasename";  
    String tableName = "tablename";  
    // Call the showPartitions method of the DLIClient object to query the partition information (including  
    the partition creation and modification time) in the database table.  
    PartitionResult partitionResult = client.showPartitions(databaseName, tableName);  
    PartitionListInfo partitonInfos = partitionResult.getPartitions();  
    // Obtain the creation and modification time of the partition.  
    Long createTime = partitonInfos.getPartitionInfos().get(0).getCreateTime().longValue();  
    Long lastAccessTime = partitonInfos.getPartitionInfos().get(0).getLastAccessTime().longValue();  
    System.out.println("createTime:"+createTime+"\nlastAccessTime:"+lastAccessTime);  
}
```

Describing Table Information

You can call an API to obtain the metadata description of a table. The example code is as follows:

```
private static void getTableDetail(Table table) throws DLIException {
    // Call getTableDetail on the Table object to obtain table information.
    // TableSchema tableSchema=table.getTableDetail();
    // Output
    System.out.println(List<Column> columns = tableSchema.getColumns());
    System.out.println(List<String> sortColumns = tableSchema.getSortColumns());
    System.out.println(String createTableSql = tableSchema.getCreateTableSql());
    System.out.println(String tableComment = tableSchema.getTableComment());
}
```

4.6.3 Job-related SDKs

Importing Data

DLI provides an API for importing data. You can use it to import data stored in OBS to a created DLI or OBS table. The example code is as follows:

```
//Instantiate the importJob object. The input parameters of the constructor include the queue, database
name, table name (obtained by instantiating the Table object), and data path.
private static void importData(Queue queue, Table DLITable) throws DLIException {
    String dataPath = "OBS Path";
    queue = client.getQueue("queueName");
    CsvFormatInfo formatInfo = new CsvFormatInfo();
    formatInfo.setWithColumnHeader(true);
    formatInfo.setDelimiter(",");
    formatInfo.setQuoteChar("\"");
    formatInfo.setEscapeChar("\\");
    formatInfo.setDateFormat("yyyy/MM/dd");
    formatInfo.setTimestampFormat("yyyy-MM-dd HH:mm:ss");
    String dbName = DLITable.getDb().getDatabaseName();
    String tableName = DLITable.getTableName();
    ImportJob importJob = new ImportJob(queue, dbName, tableName, dataPath);
    importJob.setStorageType(StorageType.CSV);
    importJob.setCsvFormatInfo(formatInfo);
    System.out.println("start submit import table: " + DLITable.getTableName());
    //Call the submit interface of the ImportJob object to submit the data importing job.
    importJob.submit(); //Call the getStatus interface of the ImportJob object to query the status of the data
importing job.
    JobStatus status = importJob.getStatus();
    System.out.println("Job id: " + importJob.getJobId() + ", Status : " + status.getName());
}
```

NOTE

- Before submitting the data importing job, you can set the format of the data to be imported. In the sample code, the `setStorageType` interface of the `ImportJob` object is called to set the data storage type to `csv`. The data format is set by calling the `setCsvFormatInfo` interface of the `ImportJob` object.
- Before submitting the data import job, you can set the partition of the data to be imported and whether to overwrite the data. You can call the `setPartitionSpec` API of the `ImportJob` object to set the partition information, for example, `importJob.setPartitionSpec(new PartitionSpec("part1=value1,part2=value2"))`. You can also create the partition using parameters when creating the `ImportJob` object. By default, data is appended to an import job. To overwrite the existing data, call the `setOverWrite` API of the `ImportJob` object, for example, `importJob.setOverWrite(Boolean.TRUE)`.
- If a folder and a file under an OBS bucket directory have the same name, data is preferentially loaded to the file, instead of the folder. It is recommended that the files and folders of the same level have different names when you create an OBS object.

Importing the Partition Data

DLI provides an API for importing data. You can use it to import data stored in OBS to a specified partition of the created DLI or OBS table. The example code is as follows:

```
//Instantiate the importJob object. The input parameters of the constructor include the queue, database name, table name (obtained by instantiating the Table object), and data path.
private static void importData(Queue queue, Table DLITable) throws DLIException {
    String dataPath = "OBS Path";
    queue = client.getQueue("queueName");
    CsvFormatInfo formatInfo = new CsvFormatInfo();
    formatInfo.setWithColumnHeader(true);
    formatInfo.setDelimiter(",");
    formatInfo.setQuoteChar("\"");
    formatInfo.setEscapeChar("\\");
    formatInfo.setDateFormat("yyyy/MM/dd");
    formatInfo.setTimestampFormat("yyyy-MM-dd HH:mm:ss");
    String dbName = DLITable.getDb().getDatabaseName();
    String tableName = DLITable.getTableName();
    PartitionSpec partitionSpec = new PartitionSpec("part1=value1,part2=value2");
    Boolean isOverWrite = true;
    ImportJob importJob = new ImportJob(queue, dbName, tableName, dataPath, partitionSpec,
isOverWrite);
    importJob.setStorageType(StorageType.CSV);
    importJob.setCsvFormatInfo(formatInfo);
    System.out.println("start submit import table: " + DLITable.getTableName());
    //Call the submit interface of the ImportJob object to submit the data importing job.
    importJob.submit(); //Call the getStatus interface of the ImportJob object to query the status of the data
importing job.
    JobStatus status = importJob.getStatus();
    System.out.println("Job id: " + importJob.getJobId() + ", Status : " + status.getName());
}
```

NOTE

- When the **ImportJob** object is created, the partition information **PartitionSpec** can also be directly transferred as the partition character string.
- If some columns are specified as partition columns during **partitionSpec** import but the imported data contains only the specified partition information, the unspecified partition columns after data import contain abnormal values such as null.
- In the example, **isOverWrite** indicates whether to overwrite data. The value **true** indicates that data is overwritten, and the value **false** indicates that data is appended. Currently, **overwrite** is not supported to overwrite the entire table. Only the specified partition can be overwritten. To append data to a specified partition, set **isOverWrite** to **false** when creating the import job.

Exporting Data

DLI provides an API for exporting data. You can use it to export data from a DLI table to OBS. The example code is as follows:

```
//Instantiate the ExportJob object and transfer the queue, database name, table name (obtained by instantiating the Table object), and storage path of the exported data. The table type must be MANAGED.
private static void exportData(Queue queue, Table DLITable) throws DLIException {
    String dataPath = "OBS Path";
    queue = client.getQueue("queueName");
    String dbName = DLITable.getDb().getDatabaseName();
    String tableName = DLITable.getTableName();
    ExportJob exportJob = new ExportJob(queue, dbName, tableName, dataPath);
    exportJob.setStorageType(StorageType.CSV);
    exportJob.setCompressType(CompressType.GZIP);
    exportJob.setExportMode(ExportMode.ERRORIFEXISTS);
    System.out.println("start export DLI Table data...");
    // Call the submit interface of the ExportJob object to submit the data exporting job.
    exportJob.submit();
}
```

```
// Call the getStatus interface of the ExportJob object to query the status of the data exporting job.
JobStatus status = exportJob.getStatus();
System.out.println("Job id: " + exportJob.getJobId() + ", Status : " + status.getName());
}
```

NOTE

- Before submitting the data exporting job, you can optionally set the data format, compression type, and export mode. In the preceding sample code, the `setStorageType`, `setCompressType`, and `setExportMode` interfaces of the `ExportJob` object are called to set the data format, compression type, and export mode, respectively. The `setStorageType` interface supports only the CSV format.
- If a folder and a file under an OBS bucket directory have the same name, data is preferentially loaded to the file, instead of the folder. It is recommended that the files and folders of the same level have different names when you create an OBS object.

Submitting a Job

DLI provides APIs for submitting and querying jobs. You can submit a job by calling the API. You can also call the API to query the job result. The example code is as follows:

```
//Instantiate the SQLJob object and construct input parameters for executing SQL, including the queue,
database name, and SQL statements.
private static void runSqlJob(Queue queue, Table obsTable) throws DLIException {
    String sql = "select * from " + obsTable.getTableName();
    String queryResultPath = "OBS Path";
    SQLJob sqlJob = new SQLJob(queue, obsTable.getDb().getDatabaseName(), sql);
    System.out.println("start submit SQL job...");
    // Call the submit interface of the SQLJob object to submit the querying job.
    sqlJob.submit();
    // Call the getStatus interface of the SQLJob object to query the status of the querying job.
    JobStatus status = sqlJob.getStatus();
    System.out.println(status);
    System.out.println("start export Result...");
    //Call the exportResult interface of the SQLJob object to export the query result. queryResultPath refers
to the path of the data to be exported.
    sqlJob.exportResult(queryResultPath, StorageType.CSV,
        CompressType.GZIP, ExportMode.ERRORIFEXISTS, null);
    System.out.println("Job id: " + sqlJob.getJobId() + ", Status : " + status.getName());
}
```

Canceling a Job

DLI provides an API for canceling jobs. You can use it to cancel all jobs in the **Launching** or **Running** state. The following sample code is used for canceling jobs in the **Launching** state:

```
private static void cancelSqlJob(DLIClient client) throws DLIException {
    List<JobResultInfo> jobResultInfos = client.listAllJobs(JobType.QUERY);
    for (JobResultInfo jobResultInfo : jobResultInfos) {
        //Cancel jobs in the LAUNCHING state.
        if (JobStatus.LAUNCHING.equals(jobResultInfo.getJobStatus())) {
            //Cancel the job of a specific job ID.
            client.cancelJob(jobResultInfo.getJobId());
        }
    }
}
```

Querying All Jobs

DLI provides an API for querying jobs. You can use it to query all jobs of the current project. The example code is as follows:


```
private static void listAllSqlJobs(DLIClient client) throws DLIException {
    //Return the collection of JobResultInfo lists.
    List < JobResultInfo > jobResultInfos = client.listAllJobs();
    //Traverse the JobResultInfo lists to view job information.
    for (JobResultInfo jobResultInfo: jobResultInfos) {
        //job id
        System.out.println(jobResultInfo.getJobId());
        //Job description
        System.out.println(jobResultInfo.getDetail());
        //job status
        System.out.println(jobResultInfo.getJobStatus());
        //job type
        System.out.println(jobResultInfo.getJobType());
    }
    //Filter the query result by job type.
    List < JobResultInfo > jobResultInfos1 = client.listAllJobs(JobType.DDL);
    //Filter the query result by job type and start time that is in the Unix timestamp format.
    List < JobResultInfo > jobResultInfos2 = client.listAllJobs(1502349803729L, 1502349821460L,
JobType.DDL);
    //Filter the query result by page.
    List < JobResultInfo > jobResultInfos3 = client.listAllJobs(100, 1, JobType.DDL);
    //Filter the query result by page, start time, and job type.
    List < JobResultInfo > jobResultInfos4 = client.listAllJobs(100, 1, 1502349803729L, 1502349821460L,
JobType.DDL);

    // Use Tags to query jobs that meet the conditions.
    JobFilter jobFilter = new JobFilter();
    jobFilter.setTags("workspace=space002,jobName=name002");
    List < JobResultInfo > jobResultInfos1 = client.listAllJobs(jobFilter);
    // Use Tags to query target jobs of a specified page.
    JobFilter jobFilter = new JobFilter();
    jobFilter.setTags("workspace=space002,jobName=name002");
    jobFilter.setPageSize(100);
    jobFilter.setCurrentPage(0);
    List < JobResultInfo > jobResultInfos1 = client.listJobsByPage(jobFilter);
}
```

NOTE

- Parameters in the OVERRIDE method can be set to **null**, indicating that no filter conditions are specified. Ensure that all parameters are set to valid values. If the page parameter is set to **-1**, the query will fail.
- APIs in this SDK do not support SQL patterns. You cannot match SQL patterns for job query.

To query DLI jobs, use the [Querying All Jobs](#) API.

Querying Job Results

DLI provides an API for querying job results. You can use it to query information about a job of the specific job ID. The example code is as follows:

```
private static void getJobResultInfo(DLIClient client) throws DLIException {
    String jobId = "4c4f7168-5bc4-45bd-8c8a-43dfc85055d0";
    JobResultInfo jobResultInfo = client.queryJobResultInfo(jobId);
    //View information about a job.
    System.out.println(jobResultInfo.getJobId());
    System.out.println(jobResultInfo.getDetail());
    System.out.println(jobResultInfo.getJobStatus());
    System.out.println(jobResultInfo.getJobType());
}
```

Querying Jobs of the SQL Type

DLI provides an API for querying SQL jobs. You can use it to query information about recently executed jobs submitted using SQL statements in the current project. The example code is as follows:

```
private static void getJobResultInfos(DLIClient client) throws DLIException {  
    //Return the collection of JobResultInfo lists.  
    List<JobResultInfo> jobResultInfos = client.listSQLJobs();  
    //Traverse the list to view job information.  
    for (JobResultInfo jobResultInfo : jobResultInfos) {  
        //job id  
        System.out.println(jobResultInfo.getJobId());  
        //Job description  
        System.out.println(jobResultInfo.getDetail());  
        //job status  
        System.out.println(jobResultInfo.getJobStatus());  
        //job type  
        System.out.println(jobResultInfo.getJobType());  
    }  
  
    // Use Tags to query SQL jobs that meet the conditions.  
    JobFilter jobFilter = new JobFilter();  
    jobFilter.setTags("workspace=space002,jobName=name002");  
    List < JobResultInfo > jobResultInfos1 = client.listAllSQLJobs(jobFilter);  
    // Use Tags to query target SQL jobs of a specified page.  
    JobFilter jobFilter = new JobFilter();  
    jobFilter.setTags("workspace=space002,jobName=name002");  
    jobFilter.setPageSize(100);  
    jobFilter.setCurrentPage(0);  
    List < JobResultInfo > jobResultInfos1 = client.listSQLJobsByPage(jobFilter);  
}
```

Exporting Query Results

DLI provides an API for exporting query results. You can use the API to export the query job result submitted in the editing box of the current project. The example code is as follows:

```
//Instantiate the SQLJob object and construct input parameters for executing SQL, including the queue,  
database name, and SQL statements.  
private static void exportSqlResult(Queue queue, Table obsTable) throws DLIException {  
    String sql = "select * from " + obsTable.getTableName();  
    String queryResultPath = "OBS Path";  
    SQLJob sqlJob = new SQLJob(queue, obsTable.getDb().getDatabaseName(), sql);  
    System.out.println("start submit SQL job...");  
    //Call the submit interface of the SQLJob object to submit the querying job.  
    sqlJob.submit();  
    //Call the getStatus interface of the SQLJob object to query the status of the querying job.  
    JobStatus status = sqlJob.getStatus();  
    System.out.println(status);  
    System.out.println("start export Result...");  
    //Call the exportResult interface of the SQLJob object to export the query result. exportPath indicates  
the path for exporting data. JSON indicates the export format. queueName indicates the queue for  
executing the export job. limitNum indicates the number of results of the export job. 0 indicates that all  
data is exported.  
    sqlJob.exportResult(exportPath + "result", StorageType.JSON, CompressType.NONE,  
        ExportMode.ERRORIFEXISTS, queueName, true, 5);  
}
```

Previewing Job Results

DLI provides an API for previewing job results. You can call this API to obtain the first 1000 records in the result set.

```
// Initialize a SQLJob object and pass the queue, database name, and SQL statement to execute the SQL.  
private static void getPreviewJobResult(Queue queue, Table obsTable) throws DLIException {  
    String sql = "select * from " + obsTable.getTableName();  
    SQLJob sqlJob = new SQLJob(queue, obsTable.getDb().getDatabaseName(), sql);  
    System.out.println("start submit SQL job...");  
    // Call the submit method on the SQLJob object.  
    sqlJob.submit();  
    // Call the previewJobResult method on the SQLJob object to query the first 1000 records in the result set.
```

```
List<Row> rows = sqlJob.previewJobResult();
if (rows.size() > 0) {
    Integer value = rows.get(0).getInt(0);
    System.out.println("Obtain the data value in the first column at the first row." + value);
}
System.out.println("Job id: " + sqlJob.getJobId() + ", previewJobResultSize : " + rows.size());
}
```

Deprecated API

The **getJobResult** method has been discarded. You can call **DownloadJob** instead to obtain the job result.

For details about the **DownloadJob** method, obtain the **dli-sdk-java-x.x.x.zip** package by referring to section [Downloading SDK](#) and decompress the package.

4.7 SDKs Related to Flink Jobs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#) and created queues by following the instructions provided in [Queue-Related SDKs](#).

Creating a SQL Job

DLI provides an API for creating a Flink streaming SQL job. You can use it to create a Flink streaming SQL job and submit it to DLI. Sample code is as follows:

```
private static void createSQLJob(DLIClient client) throws DLIException {
    SubmitFlinkSqlJobRequest body = new SubmitFlinkSqlJobRequest();
    body.name("job-name");
    body.runMode(SubmitFlinkSqlJobRequest.RunModeEnum.SHARED_CLUSTER);
    body.checkpointEnabled(false);
    body.checkpointMode(1);
    body.jobType(SubmitFlinkSqlJobRequest.JobTypeEnum.JOB);
    JobStatusResponse result = client.submitFlinkSqlJob(body);
    System.out.println(result);
}
```

Customizing a Job

DLI provides an API for creating a user-defined Flink job. Currently, the job supports the JAR format and runs in dedicated queues. The example code is as follows:

```
private static void createFlinkJob(DLIClient client) throws DLIException {
    CreateFlinkJarJobRequest body = new CreateFlinkJarJobRequest();
    body.name("jar-job");
    body.cuNumber(2);
    body.managerCuNumber(1);
    body.parallelNumber(1);
    body.entrypoint("dli/WindowJoin.jar");
    JobStatusResponse result = client.createFlinkJarJob(body);
    System.out.println(result);
}
```

Updating a SQL Job

DLI provides an API for updating Flink streaming SQL jobs. You can use it to update a Flink streaming SQL job. Sample code is as follows:

```
private static void updateSQLJob(DLIClient client) throws DLIException {
    UpdateFlinkSqlJobRequest body = new UpdateFlinkSqlJobRequest();
    body.name("update-job");
    JobUpdateResponse result = client.updateFlinkSqlJob(body,203L);
    System.out.println(result);
}
```

Updating a Custom Job

DLI provides an API for updating user-defined Flink jobs. You can use it to update custom jobs, which currently support the JAR format and run in dedicated queues. The example code is as follows:

```
private static void updateFlinkJob(DLIClient client) throws DLIException {
    UpdateFlinkJarJobRequest body = new UpdateFlinkJarJobRequest();
    body.name("update-job");
    JobUpdateResponse result = client.updateFlinkJarJob(body,202L);
    System.out.println(result);
}
```

Querying the List of Jobs

DLI provides an API for querying the Flink job list. The following parameters are involved in this API: **name**, **status**, **show_detail**, **cursor**, **next**, **limit**, and **order**. In this example, the query results are displayed in descending order and information about the jobs whose IDs are less than the value of **cursor** is displayed. The example code is as follows:

```
private static void QueryFlinkJobListResponse(DLIClient client) throws DLIException {
    QueryFlinkJobListResponse result = client.getFlinkJobs(null, "job_init", null, true, 0L, 10, null,
    null,null,null,null);
    System.out.println(result);
}
```

Querying Job Details

DLI provides an API for querying Flink job details. The example code is as follows:

```
private static void getFlinkJobDetail(DLIClient client) throws DLIException {
    Long jobId = 203L; //Job ID
    GetFlinkJobDetailResponse result = client.getFlinkJobDetail(jobId);
    System.out.println(result);
}
```

Querying the Job Execution Plan Diagram

DLI provides an API for querying the execution plan of a Flink job. The example code is as follows:

```
private static void getFlinkJobExecuteGraph(DLIClient client) throws DLIException {
    Long jobId = 203L; //Job ID
    FlinkJobExecutePlanResponse result = client.getFlinkJobExecuteGraph(jobId);
    System.out.println(result);
}
```

Querying Job Monitoring Information

DLI provides an API for querying Flink job monitoring information. Monitoring information about multiple jobs can be queried at the same time. The example code is as follows:

```
public static void getMetrics(DLIClient client) throws DLIException{
    List< Long > job_ids = new ArrayList< > ();
    Long jobId = 6316L; //Job 1 ID
    Long jobId2 = 6945L; //Job 2 ID
    job_ids.add(jobId);
    job_ids.add(jobId2);
    GetFlinkJobsMetricsBody body = new GetFlinkJobsMetricsBody();
    body.jobIds(job_ids);
    QueryFlinkJobMetricsResponse result = client.getFlinkJobsMetrics(body);
    System.out.println(result);
}
```

Querying the APIG Address of a Job

DLI provides an API for querying the APIG access address of a Flink job. The example code is as follows:

```
private static void getFlinkApigSinks(DLIClient client) throws DLIException {
    Long jobId = 59L; //Job 1 ID
    FlinkJobApigSinksResponse result = client.getFlinkApigSinks(jobId);
    System.out.println(result);
}
```

Running a Job

DLI provides APIs for running Flink jobs. The example code is as follows:

```
public static void runFlinkJob(DLIClient client) throws DLIException{
    RunFlinkJobRequest body = new RunFlinkJobRequest();
    List<Long> jobIds = new ArrayList<>();
    Long jobId = 59L; //Job 1 ID
    Long jobId2 = 192L; //Job 2 ID
    jobIds.add(jobId);
    jobIds.add(jobId2);
    body.resumeSavepoint(false);
    body.jobIds(jobIds);
    List<GlobalBatchResponse> result = client.runFlinkJob(body);
    System.out.println(result);
}
```

Stopping a Job

DLI provides an API for stopping Flink jobs. The example code is as follows:

```
public static void stopFlinkJob(DLIClient client) throws DLIException{
    StopFlinkJobRequest body = new StopFlinkJobRequest();
    List<Long> jobIds = new ArrayList<>();
    Long jobId = 59L; //Job 1 ID
    Long jobId2 = 192L; //Job 2 ID
    jobIds.add(jobId);
    jobIds.add(jobId2);
    body.triggerSavepoint(false);
    body.jobIds(jobIds);
    List<GlobalBatchResponse> result = client.stopFlinkJob(body);
    System.out.println(result);
}
```

Deleting Jobs in Batches

DLI provides an API for deleting Flink jobs in batches. You can use the API to batch delete Flink jobs in any status. The example code is as follows:

```
public static void deleteFlinkJob(DLIClient client) throws DLIException{
    DeleteJobInBatchRequest body = new DeleteJobInBatchRequest ();
    List<Long> jobIds = new ArrayList<>();
    Long jobId = 202L; //Job 1 ID
    Long jobId2 = 203L; //Job 2 ID
    jobIds.add(jobId);
    jobIds.add(jobId2);
    body.jobIds(jobIds);
    List<GlobalBatchResponse> result = client. deleteFlinkJobInBatch(body);
    System.out.println(result);
}
```

4.8 SDKs Related to Spark Jobs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#) and created queues by following the instructions provided in [Queue-Related SDKs](#).

Submitting Batch Jobs

DLI provides an API to perform batch jobs. Sample code is as follows:

```
private static void runBatchJob(Cluster cluster) throws DLIException {
    SparkJobInfo jobInfo = new SparkJobInfo();
    jobInfo.setClassName("your.class.name");
    jobInfo.setFile("xxx.jar");
    jobInfo.setCluster_name("queueName");
    // Call the asyncSubmit method on the BatchJob object to submit the batch job.
    BatchJob job = new BatchJob(cluster, jobInfo);
    job.asyncSubmit();
    while (true) {
        SparkJobStatus jobStatus = job.getStatus();
        if (SparkJobStatus.SUCCESS.equals(jobStatus)) {
            System.out.println("Job finished");
            return;
        }
        if (SparkJobStatus.DEAD.equals(jobStatus)) {
            throw new DLIException("The batch has already exited");
        }
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

 NOTE

- A cluster is a queue created by a user.
- The input parameter cannot be in JSON format.
- DLI provides the following two APIs related to batch jobs:
 - `asyncSubmit`: This API is asynchronous. After the API is submitted, the job result is directly returned.
 - `submit`: This API is synchronous. After the API is submitted, the result is returned only after job execution is complete.

Deleting Batch Jobs

DLI provides an API for deleting batch processing jobs. The example code is as follows:

```
private static void deleteBatchJob(DLIClient client) throws DLIException {
    // Submit the ID of the Spark batch processing job.
    String batchId = "0aae0dc5-f009-4b9b-a8c3-28fbee399fa6";
    // Call the delBatch method on the BatchJob object to cancel the batch job.
    MessageInfo messageInfo = client.delBatchJob(batchId);
    System.out.println(messageInfo.getMsg());
}
```

Querying All Batch Jobs

DLI provides an API for querying batch processing jobs. You can use it to query all batch jobs of the current project. The example code is as follows:

```
private static void listAllBatchJobs(DLIClient client) throws DLIException {
    System.out.println("list all batch jobs...");
    // Call the listAllBatchJobs method on the DLIClient object to query the batch jobs.
    String queueName = "queueName";
    int from = 0;
    int size = 1000;
    // Set paging, start page, and size of each page.
    List<SparkJobResultInfo> jobResults = client.listAllBatchJobs(queueName, from, size);
    for (SparkJobResultInfo jobResult : jobResults) {
        // Job ID
        System.out.println(jobResult.getId());
        // Job app ID
        System.out.println(jobResult.getAppId());
        // Job status
        System.out.println(jobResult.getState());
    }
}
```

Querying a Batch Job Status

DLI provides an API for querying status of batch processing jobs. The following sample code calls the API to query the job status:

```
private static void getStateBatchJob(DLIClient client) throws DLIException {
    BatchJob batchJob = null;
    SparkJobInfo jobInfo = new SparkJobInfo();
    jobInfo.setClusterName("queueName");
    jobInfo.setFile("xxx.jar");
    jobInfo.setClassName("your.class.name");
    batchJob = new BatchJob(client.getCluster("queueName"), jobInfo);
    batchJob.asyncSubmit();
    SparkJobStatus sparkJobStatus=batchJob.getStatus();
    System.out.println(sparkJobStatus);
}
```

Querying Batch Job Logs

DLI provides an API for querying logs of batch processing jobs. The following sample code calls the API to query the job logs:

```
private static void getBatchJobLog(DLIClient client) throws DLIException {
    BatchJob batchJob = null;
    SparkJobInfo jobInfo = new SparkJobInfo();
    jobInfo.setClusterName("queueName");
    jobInfo.setFile("xxx.jar");
    jobInfo.setClassName("your.class.name");
    batchJob = new BatchJob(client.getCluster("queueName"), jobInfo);
    batchJob.submit();
    // Call getLog on the BatchJob object to query the batch processing job.
    int from = 0;
    int size = 1000;
    List<String> jobLogs = batchJob.getLog(from,size);
    System.out.println(jobLogs);
}
```

4.9 SDKs Related to Flink Job Templates

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Creating a Job Template

DLI provides an API for creating a Flink job template. The example code is as follows:

```
public static void createFlinkJobTemplate(DLIClient client) throws DLIException{
    CreateFlinkJobTemplateRequest body = new CreateFlinkJobTemplateRequest();
    body.name("template");
    FlinkJobTemplateCreateResponse result = client.createFlinkJobTemplate(body);
    System.out.println(result);
}
```

Updating a Job Template

DLI provides an API for updating a Flink job template. The example code is as follows:

```
public static void updateFlinkJobTemplate(DLIClient client) throws DLIException{
    Long templateId = 277L; //Template ID
    UpdateFlinkJobTemplateRequest body = new UpdateFlinkJobTemplateRequest();
    body.name("template-update");
    GlobalResponse result = client.updateFlinkJobTemplate(body,templateId);
    System.out.println(result);
}
```

Deleting a Job Template

DLI provides an API for deleting a Flink job template. A template used by jobs can also be deleted. The example code is as follows:

```
public static void deleteFlinkJobTemplate(DLIClient client) throws DLIException{
    Long templateId = 277L; //Template ID
```



```
FlinkJobTemplateDeleteResponse result = client.deleteFlinkJobTemplate(templateId);
System.out.println(result);
}
```

Querying the List of Job Templates

DLI provides an API for querying Flink job templates. In this example, the query results are displayed in descending order and information about the job templates whose IDs are less than the value of **cursor** is displayed. The example code is as follows:

```
public static void getFlinkJobTemplates(DLIClient client) throws DLIException{
    Long offset = 789L; // Long | Template offset.
    Integer limit = 56; // Integer | Maximum number of records that can be queried.
    String order = "asc"; // String | Query result display. The query results can be displayed in ascending or
descending order.
    FlinkJobTemplateListResponse result = client.getFlinkJobTemplates(offset,limit,order);
    System.out.println(result);
}
```

5 Python SDK

5.1 Instructions

- For complete sample code, refer to [Downloading SDK](#), and obtain the **dli-sdk-python-<version>.zip** package on the **DLI SDK DOWNLOAD** page. Decompress the **dli-sdk-python-<version>.zip** package and view an example, such as **dli_sdk_example.py**, in the **examples** directory.
- Before using the sample code, refer to [Configuring the Python Environment](#) to configure the running environment and dependencies.

5.2 Initializing the DLI Client

To use DLI Python SDK to access DLI, you need to initialize the DLI client. During DLI client initialization, you can use the AK/SK or token for authentication. For details about the dependencies and complete sample code, see [Instructions](#).

Example Code for AK/SK Authentication

```
def init_aksk_dli_client():
    auth_mode = 'aksk'
    region = 'xxx'
    project_id = 'xxxx'
    ak = 'xxxx'
    sk = 'xxxx'
    dli_client = DliClient(auth_mode=auth_mode, region=region, project_id=project_id, ak=ak, sk=sk)
    return dli_client
```

 NOTE

You can perform the following operations to obtain the **Access Keys**, **project ID**, and **Region**:

1. Log in to the management console.
2. Hover the cursor on the username in the upper right corner and select **My Credentials** from the drop-down list.
3. In the navigation pane on the left, choose **Access Keys** and click **Create Access Key**. Confirm that you want to proceed with the operation and click **OK**.
4. On the displayed page, click **Download**. Open the credential file to obtain the AK/SK information.
5. In the navigation pane on the left, choose **API Credentials**. In the **Projects** pane, locate **project_id** and obtain the region information.

Example Code for Token-based Authentication

```
def init_token_dli_client():
    auth_mode = 'token'
    region = 'xxx'
    project_id = 'xxxx'
    account = 'xxx account'
    user = 'xxxx'
    password = 'xxxx'
    dli_client = DliClient(auth_mode=auth_mode, region=region, project_id=project_id, account=account,
user=user, password=password)
    return dli_client
```

 NOTE

You can change the endpoint in **set** mode. Run the following statement:
dliinfo.setServerEndpoint(endpoint).

5.3 Queue-Related SDKs

Constraints

Jobs created using the SDK cannot run on the default queue.

Querying All Queues

You can use the API provided by DLI to query the queue list and select the corresponding queue to execute the job. Sample code is as follows:

```
def list_all_queues(dli_client):
    try:
        queues = dli_client.list_queues()
    except DliException as e:
        print(e)
        return

    for queue in queues:
        print(queue.name)
```

For details about the dependencies and complete sample code, see [Instructions](#).

5.4 Resource-Related SDKs

Prerequisites

- You have configured the Java SDK environment by following the instructions provided [Instructions](#).
- You have initialized the DLI Client by following the instructions provided in [Initializing the DLI Client](#).

Uploading a Resource Package

You can use the APIs provided by DLI to upload resource packages. The following is an example. For details about the dependencies and complete sample code, see [Instructions](#).

```
def upload_resource(dli_client, kind, obs_jar_paths, group_name):
    try:
        dli_client.upload_resource(kind, obs_jar_paths, group_name)
    except DliException as e:
        print(e)
    return
```

NOTE

The following describes the request parameters. For details, see [Instructions](#).

- **kind**: resource package type. The options are as follows:
 - **jar**: JAR file
 - **Pyfile**: User Python file
 - **file**: User file
 - **modelfile**: User AI model file
- **obs_jar_paths**: OBS path of the resource package. The parameter format is {bucketName}.{obs domain name}/{jarPath}/{jarName}.
Example: "https://bucketname.obs.com/jarname.jar"
- **group_name**: Name of the group to which the resource package belongs

Querying All Resource Packages

You can use the API provided by DLI to query the list of uploaded resources. The example code is as follows:

```
def list_resources(dli_client):
    try:
        resources = dli_client.list_resources()
    except DliException as e:
        print(e)
    return

    for resources_info in resources.package_resources:
        print('Package resource name:' + resources_info.resource_name)

    for group_resource in resources.group_resources:
        print('Group resource name:' + group_resource.group_name)
```

For details about the dependencies and complete sample code, see [Instructions](#).

Querying a Specified Resource Package

You can call an API to query information about the specified resource package. The sample code is as follows:

```
def get_package_resource(dli_client, resource_name, group_name):
    try:
        pkg_resource = dli_client.get_package_resource(resource_name, group_name)
        print(pkg_resource)
    except DliException as e:
        print(e)
    return
```

Deleting a Resource Package

You can call an API to delete an uploaded resource package. The sample code is as follows:

```
def delete_resource(dli_client, resource_name, group_name):
    try:
        dli_client.delete_resource(resource_name, group_name)
    except DliException as e:
        print(e)
    return
```

5.5 SDKs Related to SQL Jobs

5.5.1 Database-Related SDKs

Creating a Database

DLI provides an API for creating a database. You can use the API to create a database. The sample code is as follows:

```
def create_db(dli_client):
    try:
        db = dli_client.create_database('db_for_test')
    except DliException as e:
        print(e)
    return

print(db)
```

NOTE

- The **default** database is a built-in database. You are not allowed to create a database named **default**.
- For details about the dependencies and complete sample code, see [Instructions](#).

Deleting a Database

DLI provides an API for deleting a database. The example code is as follows:

```
def delete_db(dli_client, db_name):
    try:
        dli_client.delete_database(db_name)
    except DliException as e:
        print(e)
    return
```

 NOTE

- A database that contains tables cannot be deleted. To delete a database that contains tables, delete the tables first.
- A deleted database cannot be restored. Therefore, exercise caution when deleting a database.
- For details about the dependencies and complete sample code, see [Instructions](#).

Querying All Databases

DLI provides an API for querying the database list. The example code is as follows:

```
def list_all_dbs(dli_client):
    try:
        dbs = dli_client.list_databases()
    except DliException as e:
        print(e)
        return

    for db in dbs:
        print(db)
```

For details about the dependencies and complete sample code, see [Instructions](#).

5.5.2 Table-Related SDKs

Creating a DLI Table

DLI provides an API for creating DLI tables. Sample code is as follows:

```
def create_dli_tbl(dli_client, db_name, tbl_name):
    cols = [
        Column('col_1', 'string'),
        Column('col_2', 'string'),
        Column('col_3', 'smallint'),
        Column('col_4', 'int'),
        Column('col_5', 'bigint'),
        Column('col_6', 'double'),
        Column('col_7', 'decimal(10,0)'),
        Column('col_8', 'boolean'),
        Column('col_9', 'date'),
        Column('col_10', 'timestamp')
    ]
    sort_cols = ['col_1']
    tbl_schema = TableSchema(tbl_name, cols, sort_cols)
    try:
        table = dli_client.create_dli_table(db_name, tbl_schema)
    except DliException as e:
        print(e)
        return

    print(table)
```

For details about the dependencies and complete sample code, see [Instructions](#).

Creating an OBS Table

DLI provides an API for creating OBS tables. The example code is as follows:

```
def create_obs_tbl(dli_client, db_name, tbl_name):
    cols = [
        Column('col_1', 'string'),
        Column('col_2', 'string'),
```

```
Column('col_3', 'smallint'),
Column('col_4', 'int'),
Column('col_5', 'bigint'),
Column('col_6', 'double'),
Column('col_7', 'decimal(10,0)'),
Column('col_8', 'boolean'),
Column('col_9', 'date'),
Column('col_10', 'timestamp')
]
tbl_schema = TableSchema(tbl_name, cols)
try:
    table = dli_client.create_obs_table(db_name, tbl_schema,
                                      'obs://bucket/obj',
                                      'csv')
except DliException as e:
    print(e)
    return

print(table)
```

NOTE

- You need to create an OBS path in advance and specify it when creating an OBS table.
- For details about the dependencies and complete sample code, see [Instructions](#).

Deleting a Table

DLI provides an API for deleting tables. The example code is as follows:

```
def delete_tbls(dli_client, db_name):
    try:
        tbls = dli_client.list_tables(db_name)
        for tbl in tbls:
            dli_client.delete_table(db_name, tbl.name)
    except DliException as e:
        print(e)
    return
```

NOTE

- A deleted table cannot be restored. Exercise caution when deleting a table.
- For details about the dependencies and complete sample code, see [Instructions](#).

Querying All Tables

DLI provides an API for querying tables. The example code is as follows:

```
def list_all_tbls(dli_client, db_name):
    try:
        tbls = dli_client.list_tables(db_name, with_detail=True)
    except DliException as e:
        print(e)
    return

    for tbl in tbls:
        print(tbl.name)
```

For details about the dependencies and complete sample code, see [Instructions](#).

Describing Table Information

You can call an API to obtain the metadata description of a table. The sample code is as follows:

```
def get_table_schema(dli_client, db_name, tbl_name):
    try:
```

```
table_info = dli_client.get_table_schema(db_name, tbl_name)
print(table_info)
except DliException as e:
    print(e)
return
```

5.5.3 Job-related SDKs

For details about the dependencies and complete sample code, see [Instructions](#).

Importing Data

DLI provides an API for importing data. You can use this API to import OBS to a DLI table. Sample code is as follows:

```
def import_data(dli_client, db_name, tbl_name, queue_name):
    options = {
        "with_column_header": True,
        "delimiter": ",",
        "quote_char": "\"",
        "escape_char": "\\ ",
        "date_format": "yyyy/MM/dd",
        "timestamp_format": "yyyy/MM/dd hh:mm:ss"
    }

    try:
        job_id, status = \
            dli_client.import_table(tbl_name, db_name,
                                   'obs://bucket/obj/data.csv',
                                   'csv',
                                   queue_name=queue_name,
                                   options=options)
    except DliException as e:
        print(e)
        return

    print(job_id)
    print(status)
```

NOTE

- Before submitting the importing job, you can specify the **data_type** parameter to set the type of the data to be imported. For example, set **data_type** to **csv**. Use the **options** parameter to set details about the CSV data format, such as the delimiter and escape character.
- If a folder and a file under an OBS bucket directory have the same name, data is preferentially loaded to the file, instead of the folder. It is recommended that the files and folders of the same level have different names when you create an OBS object.

Exporting Data

DLI provides an API for exporting data. You can use this API to export DLI table data to an OBS bucket. The example code is as follows:

```
def export_data(dli_client, db_name, tbl_name, queue_name):
    try:
        job_id, status = dli_client.export_table(tbl_name, db_name,
                                                'obs://bucket/obj',
                                                queue_name=queue_name)
    except DliException as e:
        print(e)
        return

    print(job_id)
    print(status)
```


 NOTE

- Before submitting the export job, you can set the data format, compression type, and export mode. The data can only be exported in the CSV format.
- If a folder and a file under an OBS bucket directory have the same name, data is preferentially loaded to the file, instead of the folder. It is recommended that the files and folders of the same level have different names when you create an OBS object.

Submitting a Job

DLI provides an API for querying jobs. The example code is as follows:

```
def run_sql(dli_client, db_name, queue_name):
    # execute SQL
    try:
        sql_job = dli_client.execute_sql('select * from tbl_dli_for_test', db_name, queue_name=queue_name)
        result_set = sql_job.get_result(queue_name=queue_name)
    except DliException as e:
        print(e)
        return

    if result_set.row_count == 0:
        return

    for row in result_set:
        print(row)

    # export the query result to obs
    try:
        status = sql_job.export_result('obs://bucket/obj',
                                      queue_name=queue_name)
    except DliException as e:
        print(e)
        return

    print(status)
```

Canceling a Job

DLI provides an API for canceling jobs. You can use it to cancel a submitted job. A job that has been completed or failed cannot be canceled. The example code is as follows:

```
def cancel_sql(dli_client, job_id):
    try:
        dli_client.cancel_sql(job_id)
    except DliException as e:
        print(e)
        return
```

Querying All Jobs

DLI provides an API for querying all jobs. You can use the API to query information about all jobs in the current project and obtain the query result. The example code is as follows:

```
def list_all_sql_jobs(dli_client):
    try:
        sql_jobs = dli_client.list_sql_jobs()
    except DliException as e:
        print(e)
        return
    for sql_job in sql_jobs:
        print(sql_job)
```

 NOTE

APIs in this SDK do not support SQL patterns. You cannot match SQL patterns for job query. To query DLI jobs, use the [Querying All Jobs](#) API.

Querying SQL Jobs

You can call an API to query information about all SQL jobs in the current project and obtain the query result. The sample code is as follows:

```
def list_sql_jobs(dli_client):
    try:
        sql_jobs = dli_client.list_sql_jobs()
    except DliException as e:
        print(e)
    return
```

5.6 SDKs Related to Spark Jobs

For details about the dependencies and complete sample code, see [Instructions](#).

Submitting Batch Jobs

DLI provides an API to perform batch jobs. The example code is as follows:

```
def submit_spark_batch_job(dli_client, batch_queue_name, batch_job_info):
    try:
        batch_job = dli_client.submit_spark_batch_job(batch_queue_name, batch_job_info)
    except DliException as e:
        print(e)
    return

print(batch_job.job_id)
while True:
    time.sleep(3)
    job_status = batch_job.get_job_status()
    print('Job status: {}'.format(job_status))
    if job_status == 'dead' or job_status == 'success':
        break

logs = batch_job.get_driver_log(500)
for log_line in logs:
    print(log_line)
```

Canceling a Batch Processing Job

DLI provides an API for canceling batch processing jobs. If the job execution is complete or fails, you cannot cancel this job. The example code is as follows:

```
def del_spark_batch(dli_client, batch_id):
    try:
        resp = dli_client.del_spark_batch_job(batch_id)
        print(resp.msg)
    except DliException as e:
        print(e)
    return
```

Deleting Batch Processing Jobs

DLI provides an API for deleting batch processing jobs. The following sample code calls the API to delete a batch processing job:

```
def del_spark_batch(dli_client, batch_id):  
    try:  
        resp = dli_client.del_spark_batch_job(batch_id)  
        print(resp.msg)  
    except DliException as e:  
        print(e)  
    return
```

A Change History

Release On	Description
2023-09-16	This issue is the third official release. Added the description of downloading the .sha256 file corresponding to the SDK installation package to Prerequisites .
2023-07-18	This issue is the second official release. Modified the default precision of data of the decimal type in Table-Related SDKs .
2022-9-15	This issue is the first official release.