ModelArts

SDK Reference

Issue 01

Date 2023-09-27

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

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Before You Start

This document describes how to install and configure a development environment and call functions provided by ModelArts SDK for secondary development.

Section	Description
SDK Overview	Concepts of ModelArts SDK
Getting Started	How to use ModelArts SDKs for secondary development
(Optional) Installing the ModelArts SDK Locally	How to install ModelArts SDKs locally
(Optional) Session Authentication	How to authenticate public cloud resources and initialize ModelArts SDK Client and OBS Client
Overview of OBS Management	How to call the SDK APIs of Object Storage Service (OBS), including the APIs for creating OBS buckets, uploading and downloading files and folders, as well as deleting OBS objects and buckets
ModelArts SDK operations: Data Management Training Management (New Version) Training Management (Old Version) Model Management Service Management	Common operations using ModelArts SDK

2 SDK Overview

ModelArts Software Development Kits (ModelArts SDKs) encapsulate ModelArts REST APIs in Python language to simplify application development. You can directly call ModelArts SDKs to easily start AI training, generate models, and deploy the models as real-time services.

ModelArts SDKs support only Python of versions later than 3.7.x and earlier than 3.10.x. Version 3.7.x is recommended.

Scenarios

ModelArts SDKs can be used only in the ModelArts development environment notebook and local PC environment.

NOTICE

ModelArts SDKs cannot be used in training jobs or real-time services.

- ModelArts SDKs have been integrated into ModelArts notebook and can be directly used without session authentication.

  Log in to the ModelArts management console, choose **DevEnviron** > **Notebook** in the navigation pane, create a notebook instance, and call the ModelArts SDKs on the terminal or IPYNB file. You can call SDKs on a notebook instance to perform operations such as OBS management, job management, model management, and service management by referring to the SDK reference.
- ModelArts SDKs can be installed, configured, and then used in local environments after session authentication.

  a. Install the SDKs in a local path. Install the SDKs locally by referring to **(Optional) Installing the ModelArts SDK Locally**. If the SDKs have been installed in a local path, skip this step.

  b. Perform session authentication by referring to **(Optional) Session Authentication**. The SDKs can be used after the authentication is complete.

SDK Versions

Table 2-1 ModelArts SDK versions

Released On	Version	Description
2023-04	1.4.18	Optimized and integrated on the basis of earlier versions, including DLI Spark task submission and service deployment in new-version dedicated resource pools for inference.

Supported Regions

The following regions are supported: CN-Hong Kong (ap-southeast-1), AP-Bangkok (ap-southeast-2), AP-Singapore (ap-southeast-3), and LA-Santiago (la-south-2).

3 Getting Started

ModelArts SDKs can be used only in the ModelArts development environment notebook and local PC environment.

NOTICE

ModelArts SDKs cannot be used in training jobs or real-time services.

- ModelArts SDKs have been integrated into ModelArts notebook and can be directly used without session authentication.

  Log in to the ModelArts console, choose **DevEnviron** > **Notebook** in the navigation pane, create a notebook instance, and call the ModelArts SDKs on the terminal or IPYNB file. You can call SDKs on a notebook instance to perform operations such as OBS management, job management, model management, and service management by referring to the SDK reference.
- ModelArts SDKs can be installed, configured, and then used in local environments after session authentication.

  a. Install the SDKs in a local path. Install the SDKs locally by referring to **(Optional) Installing the ModelArts SDK Locally**. If the SDKs have been installed in a local path, skip this step.

  b. Perform session authentication by referring to **(Optional) Session Authentication**. The SDKs can be used after the authentication is complete.

4 (Optional) Installing the ModelArts SDK Locally

To use the ModelArts SDK on a PC or VM, install the ModelArts SDK in the target environment. After the installation, you can call the ModelArts SDK to easily manage datasets, create ModelArts training jobs and AI applications, and deploy the AI applications as real-time services.

Procedure

To install the ModelArts SDK locally, perform the following operations:

- **Step 1: Download the ModelArts SDK Package**
- **Step 2: Configure the Runtime Environment**
- **Step 3: Install the ModelArts SDK**

NOTE

ModelArts SDKs can be installed in Windows and Linux.

If an error occurred during the ModelArts SDK installation in Windows, rectify the fault by referring to **FAQ: An Error Occurred During ModelArts SDK Installation in Windows**.

Step 1: Download the ModelArts SDK Package

1. **Download the ModelArts SDK software package** of the latest version.
2. (Optional) Verify the software package signature.
   a. **Download the signature verification file of the software package**.
   b. Run the following command to install OpenSSL and verify software consistency:

```
openssl cms -verify -binary -in D:\modelarts-latest-py2.py3-none-any.whl.cms -inform DER -content D:\modelarts-latest-py2.py3-none-any.whl  -noverify > ./test
```

NOTE

Replace the software package path in the example to the actual path.

Step 2: Configure the Runtime Environment

1. Check whether Python has been installed locally. If not, download Python of a proper version at the **Python official website** and install it. The Python version must be later than 3.7.x and earlier than 3.10.x. Version 3.7.x is recommended.

   Run the **python --version** command in the local environment. If the following information is displayed, Python has been installed:

   ```
   C:\Users\xxx>python --version
   Python *.*.*
   ```

2. Check whether pip, package installer for Python, has been installed. If not, install pip by following the instructions provided at the **pip official website** after you install Python.

   Run the **pip --version** command in the local environment. If the following information is displayed, pip has been installed:

   ```
   C:\Users\xxx>pip --version
   pip **.*.* from c:\users\xxx\appdata\local\programs\python\python**\lib\site-packages\pip (python *.*)
   ```

   NOTE

   In Windows, if a message is displayed indicating that the command is not an internal or external command, add the Python and pip installation paths to **Path** in the environment variable. The pip installation path is typically the **Scripts** folder in the directory where Python is located.

   1. Press **Win+R**, enter **sysdm.cpl** in the **Run** dialog box, and click **OK**.
   2. In the **System Properties** dialog box, click the **Advanced** tab and click **Environment Variables**.
   3. In the **User variables for** area, double-click **Path**. In the **Edit environment variable** dialog box, click **New** and add the Python and pip installation paths. The installation path must point to the **Scripts** folder, for example, **C:\python\python**\Scripts**.

3. Configure the pip source. The following uses Windows as an example to describe how to configure the pip source:

   a. Create a **pip** folder. Start **cmd** and run the **set** command to view the **AppData** path. Create a **pip** folder in the obtained **AppData** path. An example is provided as follows:

   ```
   C:\Users\xxx>set
   ALLUSERSPROFILE=C:\ProgramData
   APPDATA=C:\Users\xxx\AppData\Roaming
   ```

   The preceding information indicates that the **pip** folder needs to be created in **C:\Users\xxx\AppData\Roaming**.

   b. Create a text file named **pip** in the **pip** folder and change the file name extension from .txt to .ini. An example is provided as follows:

   index-url is the IP address of the pip source, which needs to be replaced as required. The following uses a Huawei source as an example.

   ```
   [global]
   index-url = https://mirrors.huaweicloud.com/repository/pypi/simple
   trusted-host = mirrors.huaweicloud.com
   disable-pip-version-check = true
   timeout = 120
   [install]
   ignore-installed = true
   no-dependencies = yes
   ```

4. Start **cmd** and run the following command to download the package of the required pip source:

   ```
   C:\Users\xxx>pip install numpy      # Replace numpy with the package you want to download.
   ```


Step 3: Install the ModelArts SDK

Start **cmd** and run the following command to install the ModelArts SDK:

pip install *{Path to the SDK software package}***modelarts-latest-py2.py3-none-any.whl**

```
C:\Users\xxx>pip install C:\Users\xxx\Downloads\modelarts-latest-py2.py3-none-any.whl
......
Successfully installed Pillow-*.*.0 ... modelarts-*.*.* ...
```

When SDK is installed, dependency packages are installed by default. If message "Successfully installed" is displayed, the ModelArts SDK has been installed.

NOTE

If an error message is displayed during the installation, indicating that a dependency package is missing, run the following command to install the dependency package as prompted:

pip install xxxx

xxxx is the name of the dependency package.

Follow-Up Operations

After installing ModelArts SDKs locally, you need to complete **session authentication**. After session authentication is complete, you can directly call the ModelArts SDKs.

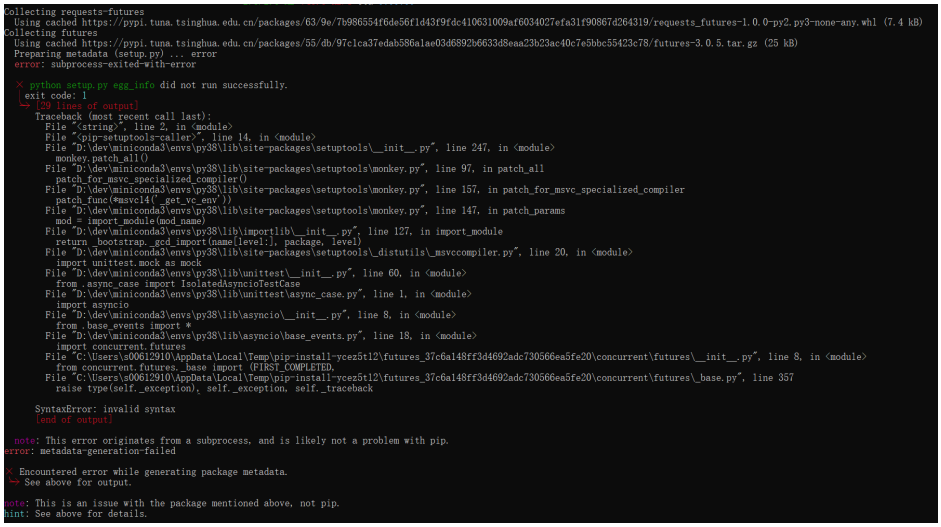
FAQ: An Error Occurred During ModelArts SDK Installation in Windows

When installing the ModelArts SDK in Windows, ensure the Python version is not later than 3.10.x. Python 3.7.x is recommended.

If the error shown in the following figure is displayed when you install the SDK on the local computer, install the **futures** dependency package of version 3.1.1 and then reinstall the SDK.

```
pip install futures==3.1.1
```

Figure 4-1 Error message displayed during ModelArts SDK installation



5 Session Authentication

5.1 (Optional) Session Authentication

Overview

The session module authenticates public cloud resources and initializes ModelArts SDK Client and OBS Client. After a session is set up, you can directly call the ModelArts SDKs.

- ModelArts notebook does not require session authentication. The sample code is as follows:

```
from modelarts.session import Session
session = Session()
```

- Session authentication is required when the local PC uses ModelArts SDKs. You can select either of the following authentication modes:
  - **Authentication Using the Username and Password**: Available for **OBS Management**, **Data Management**, **Training Management (New Version)**, **Training Management (Old Version)**, **Model Management**, and **Service Management**.
  - **AK/SK-based Authentication**: Available for **OBS Management**, **Data Management**, **Training Management (New Version)**, **Training Management (Old Version)**, **Model Management**, and **Service Management**.

Authentication Using the Username and Password

After installing the ModelArts SDK on the local PC, you can perform session authentication using the username and password. The sample code is as follows:

- Authentication using an account

  Set **username** to your account name.

```
from modelarts.session import Session

# Hardcoded or plaintext password is risky. For security, encrypt your password and store it in the configuration file or environment variables.
# In this example, the password is stored in environment variables for identity authentication. Before running this example, set environment variable HUAWEICLOUD_SDK_PASSWORD.
__PASSWORD = os.environ["HUAWEICLOUD_SDK_PASSWORD"]
```



```
# Decrypt the password if it is encrypted.
session = Session(username='***',  password=__PASSWORD, region_name='***', project_id='***')
```

- Authentication using an IAM user

  Set **account** to your account name and **username** to your IAM username.

```
from modelarts.session import Session

# Hardcoded or plaintext password is risky. For security, encrypt your password and store it in the configuration file or environment variables.
# In this example, the password is stored in environment variables for identity authentication. Before running this example, set environment variable HUAWEICLOUD_SDK_PASSWORD.
__PASSWORD = os.environ["HUAWEICLOUD_SDK_PASSWORD"]
# Decrypt the password if it is encrypted.
session = Session(account='***', username='***',  password=__PASSWORD, region_name='***', project_id='***')
```

NOTE

For the concepts of the account and user, see **Basic Concepts of IAM**. For details about how to obtain your account and username, see **Obtaining the Username, User ID, Project Name, and Project ID**.

If your Huawei Cloud account has been upgraded to a HUAWEI ID, account authentication will be unavailable. In this case, create an IAM user and use it for authentication.

AK/SK-based Authentication

After installing the ModelArts SDK on the local PC, you can perform session authentication using the AK/SK. The sample code is as follows:

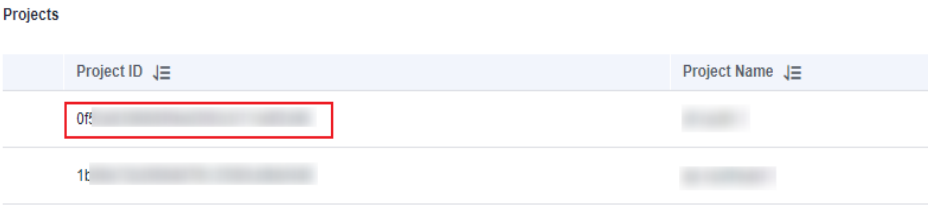
```
from modelarts.session import Session

# Hardcoded or plaintext AK/SK is risky. For security, encrypt your AK/SK and store them in the configuration file or environment variables.
# In this example, the AK/SK are stored in environment variables for identity authentication. Before running this example, set environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
__AK = os.environ["HUAWEICLOUD_SDK_AK"]
__SK = os.environ["HUAWEICLOUD_SDK_SK"]
# Decrypt the password if it is encrypted.
session = Session(access_key=__AK,secret_key=__SK, project_id='***', region_name='***')
```

Parameters are as follows:

- To obtain **access_key** and **secret_key**, follow these steps:
  a. Log in to the management console, hover over your username in the upper right corner and choose **My Credentials**. The **My Credentials** page is displayed.
  b. Choose **Access Keys** and click **Create Access Key**.
  c. In the **Create Access Key** dialog box, enter the description and click **OK**. Click **Download** to download the key. The access key file will be saved in the default download folder of the browser. Open the **credentials.csv** file to view the access key (**Access Key Id** and **Secret Access Key**).
- **project_id** indicates the project ID. To obtain the project ID, do as follows:

  On the **My Credentials** page, click **API Credentials**. In the project list, view the project ID and name. If there are multiple projects, unfold the target region and obtain the project ID from the **Project ID** column.

Figure 5-1 Viewing a project ID



- **region_name** indicates the region ID. For details about how to obtain a region ID, see **Obtaining Region Information**.

5.2 Authentication Using the Username and Password

This authentication method is available for **OBS Management**, **Training Management**, **Model Management**, and **Service Management**.

Sample Code

For details about the concepts of the account and user, see **Basic Concepts of IAM**. For details about how to obtain your account and username information, see **Obtaining Account, IAM User, Group, Project, Region, and Agency Information**.

- Authentication using an account

  Set **username** to your account name.

  ```
  from modelarts.session import Session
  session = Session(username='***',  password='***', region_name='***', project_id='***')
  ```

  NOTE

  If your HUAWEI CLOUD account has been upgraded to a HUAWEI CLOUD account, account authentication will be unavailable. In this case, create an IAM user and use it for authentication.

- Authentication using an IAM user

  Set **account** to your account name and **username** to your IAM username.

  ```
  from modelarts.session import Session
  session = Session(account='***', username='***',  password='***', region_name='***', project_id='***')
  ```

5.3 AK/SK-based Authentication

This authentication method is available for **OBS Management**, **Training Management**, **Model Management**, and **Service Management**.

Sample Code

```
from modelarts.session import Session
session = Session(access_key='***',secret_key='***', project_id='***', region_name='***')
```

Parameters in this command are described as follows:

- To obtain **access_key** and **secret_key**, follow these steps:

  a. Log in to the management console, hover over your username in the upper right corner and choose **My Credentials**. The **My Credentials** page is displayed.

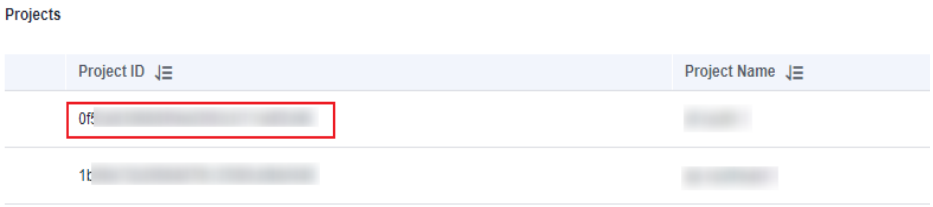
b. Choose **Access Keys** and click **Create Access Key**.

c. In the **Create Access Key** dialog box, enter the description and click **OK**. Click **Download** to download the key. The access key file will be saved in the default download folder of the browser. Open the **credentials.csv** file to view the access key (**Access Key Id** and **Secret Access Key**).

- **project_id** indicates the project ID. To obtain the project ID, do as follows:

  On the **My Credentials** page, click **API Credentials**. In the project list, view the project ID and name. If there are multiple projects, unfold the target region and obtain the project ID from the **Project ID** column.

  Figure 5-2 Viewing a project ID

- **region_name** indicates the region ID. For details about how to obtain a region ID, see **Obtaining Region Information**.

6 OBS Management

6.1 Overview of OBS Management

ModelArts SDK 1.1.3 supports OBS management, including uploading and downloading files and folders. The operations are as follows:

- **Uploading a File to OBS**
- **Uploading a Folder to OBS**
- **Downloading a File from OBS**
- **Downloading a Folder from OBS**

6.2 Transferring Files (Recommended)

NOTE

Through file transferring, local files and folders can be uploaded to OBS, and the files and folders in OBS can be downloaded to a local path.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
session = Session()
# 1. Upload a local file to OBS.
session.obs.copy(src_path='/home/ma-user/file1.txt', dst_path='obs://bucket-name/dir1/file1.txt')

# 2. Download a file from OBS to a local path.
session.obs.copy(src_path='obs://bucket-name/dir1/file1.txt', dst_path='/home/ma-user/file1.txt')

# 3. Upload a local folder to OBS.
session.obs.copy(src_path='/home/ma-user', dst_path='obs://bucket-name/dir1', keep_last_dir=True)

# 4. Download a folder from OBS to a local path.
session.obs.copy(src_path='obs://bucket-name/dir1', dst_path='/home/ma-user', keep_last_dir=True)
```


Table 6-1 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object.
src_path	Yes	String	Path to the source file or folder. If the source path is an OBS path, the path prefix must be **obs://**.
dst_path	Yes	String	Path to the destination file or folder. If the destination path is an OBS path, the path prefix must be **obs://**.
keep_last_dir	No	Boolean	Whether to copy the last-level directory of the source folder to the destination folder. The default value is **True**. This parameter is valid only for copying folders.

Table 6-2 Failure parameters

Parameter	Type	Description
error_code	String	Error code when calling the SDK failed. This parameter is unavailable for a successful call.
error_msg	String	Error message when calling the SDK failed. This parameter is unavailable for a successful call.

6.3 Uploading a File to OBS

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
session = Session()
session.obs.upload_file(src_local_file='/home/ma-user/file1.txt', dst_obs_dir='obs://bucket-name/dir1/')
```

After the sample code is executed, the local source file **file1.txt** is uploaded to the **dir1** folder in the **bucket-name** bucket. The path is **obs://bucket-name/dir1/file1.txt**. The bucket name and folder name are user-defined.

Parameters

Table 6-3 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object
src_local_file	Yes	String	Path to the local file to be uploaded
dst_obs_dir	Yes	String	Path to the target OBS bucket. The path must start with **obs://** and end with a slash (/).

Table 6-4 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

6.4 Uploading a Folder to OBS

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
session = Session()
session.obs.upload_dir(src_local_dir='/home/ma-user/', dst_obs_dir='obs://bucket-name/dir1/')
```

After the sample code is executed, the local source folder **/ma-user/** is uploaded to the **dir1** folder in the **bucket-name** bucket. The path is **obs://bucket-name/dir1/ma-user/**. The bucket name and folder name are user-defined.

Parameters

Table 6-5 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object
src_local_dir	Yes	String	Path to the local folder to be uploaded. If the folder to be uploaded is empty or contains multiple empty folders, no empty folders are created in the corresponding OBS path.
dst_obs_dir	Yes	String	Path to the target OBS bucket. The path must start with **obs://** and end with a slash (/).

Table 6-6 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

6.5 Downloading a File from OBS

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
session = Session()
session.obs.download_file(src_obs_file="obs://bucket-name/dir1/file1.txt", dst_local_dir="/home/ma-user/")
```

After the sample code is executed, source file **file1.txt** is downloaded from OBS to **/home/ma-user/file1.txt**.

Parameters

Table 6-7 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object
src_obs_file	Yes	String	Path to the source file to be downloaded from OBS. The path must start with **obs://**.
dst_local_dir	Yes	String	Path to the target local folder. The path must end with a slash (/).

Table 6-8 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

6.6 Downloading a Folder from OBS

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
session = Session()
session.obs.download_dir(src_obs_dir="obs://bucket-name/dir1/", dst_local_dir="/home/ma-user/work/")
```

After the sample code is executed, source folder **dir1** is downloaded from OBS to **/home/ma-user/work/dir1/**.

CAUTION

You must have the write permission on the local path.

Parameters

Table 6-9 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object
src_obs_dir	Yes	String	Path to the source folder to be downloaded from OBS. The path must start with **obs://** and end with a slash (/). If the downloaded folder contains empty folders, no empty folders are created in the corresponding local path.
dst_local_dir	Yes	String	Path to the target local folder. The path must end with a slash (/).

Table 6-10 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

7 Data Management

7.1 Managing Datasets

7.1.1 Querying a Dataset List

Obtain a dataset list by page.

```
list_datasets(session, dataset_type=None, dataset_name=None, offset=None, limit=None)
```

Sample Code

- Example 1: Obtain a dataset list.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()
# Obtain a dataset list.
dataset_list = Dataset.list_datasets(session)
print(dataset_list) # Print the query result.
```

- Example 2: Obtain a dataset list by dataset type.

```
# Obtain image classification datasets.
dataset_list = Dataset.list_datasets(session, dataset_type=0)
print(dataset_list)
```

- Example 3: Obtain a dataset list by dataset name.

```
# Obtain the datasets with dataset contained in dataset names.
dataset_list = Dataset.list_datasets(session, dataset_name="dataset")
print(dataset_list)
```

- Example 4: Obtain a dataset list by page.

```
# By default, 10 dataset records are returned at a time. You can set limit and offset for query by page.
dataset_list = Dataset.list_datasets(session, offset=0, limit=50) # Obtain the 1st to 50th records.
print(dataset_list)
dataset_list = Dataset.list_datasets(session, offset=1, limit=50) # Obtain the 51st to 100th records.
print(dataset_list)
```


Parameters

Table 7-1 Request parameters

Name	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
dataset_type	No	Integer	Obtain a dataset list by dataset type. By default, this parameter is left blank. The options are as follows: • **0**: image classification • **1**: object detection • **3**: image segmentation • **100**: text classification • **101**: named entity recognition • **102**: text triplet • **200**: sound classification • **201**: speech content • **202**: speech paragraph labeling • **400**: table dataset • **600**: video labeling • **900**: custom format
dataset_name	No	String	Fuzzy search keyword. By default, this parameter is left blank.
offset	No	Integer	Start page for pagination display. The default value is **0**.
limit	No	Integer	Maximum number of records returned on each page. The value ranges from 1 to 100. The default value is **10**.

7.1.2 Creating a Dataset

Create a dataset whose data can be imported from OBS.

```
create_dataset(session, dataset_name=None, data_type=None, data_sources=None, work_path=None, dataset_type=None, **kwargs)
```

Use either of the following methods to create a dataset:

- Create a dataset based on the labeling type. One dataset supports only one labeling task type.


```
create_dataset(session,dataset_name=None, dataset_type=None, data_sources=None,
work_path=None, **kwargs)
```

- Create a dataset based on the data type. You can create different types of labeling tasks on the same dataset. For example, create image classification and object detection labeling tasks on an image dataset.

```
create_dataset(session,dataset_name=None, data_type=None, data_sources=None, work_path=None,
**kwargs)
```

NOTE

You are advised to create a dataset based on the data type. Creating a dataset based on the labeling type will be terminated.

Sample Code

- Example 1: Create an image dataset based on the data type.

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()

dataset_name = "dataset-image"  # Dataset name
data_type = "IMAGE"             # Dataset type, which is an image dataset
data_sources = dict()           # Dataset data source
data_sources["type"] = 0        # Data source type. Value 0 indicates OBS.
data_sources["path"] = "/obs-gaia-test/data/image/image-classification/" # Path for storing data in
OBS
work_path = dict()              # Work directory of the dataset
work_path['type'] = 0           # Working directory type of the dataset. Value 0 indicates OBS.
work_path['path'] = "/obs-gaia-test/data/output/work_path/"  # Path for the working directory of the
dataset in OBS
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
data_type=data_type,
                                             data_sources=data_sources, work_path=work_path)
```

- Example 2: Create an image dataset based on the data types (labels imported).

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()

dataset_name = "dataset-image-with-annotations"
data_type = "IMAGE"
data_sources = dict()
data_sources["type"] = 0
data_sources["path"] = "/obs-gaia-test/data/image/image-classification/"
annotation_config = dict()      # Labeling format of the source data
annotation_config['scene'] = "image_classification" # Image classification labeling
annotation_config['format_name'] = "ModelArts image classification 1.0" # Labeling format of
ModelArts image classification 1.0
data_sources['annotation_config'] = annotation_config
work_path = dict()
work_path['type'] = 0
work_path['path'] = "/obs-gaia-test/data/output/work_path/"
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
data_type=data_type,
                                             data_sources=data_sources, work_path=work_path)
```

- Example 3: Create a table dataset based on the data type.

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()

dataset_name = "dataset-table"
data_type = "TABLE"
data_sources = dict()
```



```
data_sources["type"] = 0
data_sources["path"] = "/obs-gaia-test/data/table/table0/"
data_sources['with_column_header'] = True
work_path = dict()
work_path['type'] = 0
work_path['path'] = "/obs-gaia-test/data/output/work_path/"
# Schema information of the table data needs to be specified for the table dataset.
schema0 = dict()
schema0['schema_id'] = 0
schema0['name'] = "name"
schema0['type'] = "STRING"
schema1 = dict()
schema1['schema_id'] = 1
schema1['name'] = "age"
schema1['type'] = "STRING"
schema2 = dict()
schema2['schema_id'] = 2
schema2['name'] = "label"
schema2['type'] = "STRING"
schemas = []
schemas.append(schema0)
schemas.append(schema1)
schemas.append(schema2)
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
data_type=data_type,
                                             data_sources=data_sources, work_path=work_path, schema=schemas)
```

- Example 4: Create an image classification dataset based on the labeling type.

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()

dataset_name = "dataset-image-classification"
dataset_type = 0   # Dataset labeling type. Value 0 indicates image classification.
data_sources = dict()
data_sources["path"] = "/obs-gaia-test/data/image/image-classification/"
data_sources["type"] = "0"
work_path = dict()
work_path['type'] = 0
work_path['path'] = "/obs-gaia-test/data/output/work_path/"
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
dataset_type=dataset_type, data_sources=data_sources, work_path=work_path)
```

- Example 5: Create a text triplet dataset based on the labeling type.

```
dataset_name = "dataset-text-triplet"
dataset_type = 102   # Dataset labeling type. Value 102 indicates text triplet.
data_sources = dict()
data_sources['type'] = 0
data_sources['path'] = "/obs-gaia-test/data/text/text-classification/"
work_path = dict()
work_path['type'] = 0
work_path['path'] = "/obs-gaia-test/data/output/work_path/"

# Create a dataset of the text triplet labeling type with labels imported.
label_entity1 = dict()    # Label object
label_entity1['name'] = "Disease"    # Label name
label_entity1['type'] = 101     # Label type. Value 101 indicates an entity.
label_entity2 = dict()
label_entity2['name'] = "Disease alias"
label_entity2['type'] = 101
label_relation1 = dict()
label_relation1['name'] = "Also called"
label_relation1['type'] = 102    # Label type. Value 102 indicates relational.
property = dict()    # For a relational label, the start entity label and end entity label must be
specified in label properties.
property['@modelarts:from_type'] = "Disease"    # Start entity label
property['@modelarts:to_type'] = "Disease alias"    # End entity label
label_relation1['property'] = property
labels = []
```



```
labels.append(label_entity1)
labels.append(label_entity2)
labels.append(label_relation1)
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
dataset_type=dataset_type, data_sources=data_sources, work_path=work_path, labels=labels)
```

- Example 6: Create a table dataset based on the labeling type.

```
dataset_name = "dataset-table"
dataset_type = 400    # Dataset labeling type. Value 400 indicates a table dataset.
data_sources = dict()
data_sources['type'] = 0
data_sources['path'] = "/obs-gaia-test/data/table/table0/"
data_sources['with_column_header'] = True    # Whether the table data contains a table header
work_path = dict()
work_path['type'] = 0
work_path['path'] = "/obs-gaia-test/data/output/work_path/"

# The table header of the table data needs to be imported to the table dataset.
schema0 = dict()    # Table header
schema0['schema_id'] = 0    # Header of the first column
schema0['name'] = "name"    # Table header name, which is name in the column
schema0['type'] = "STRING"    # Data type of the table header, indicating a character string
schema1 = dict()
schema1['schema_id'] = 1
schema1['name'] = "age"
schema1['type'] = "STRING"
schema2 = dict()
schema2['schema_id'] = 2
schema2['name'] = "label"
schema2['type'] = "STRING"
schemas = []
schemas.append(schema0)
schemas.append(schema1)
schemas.append(schema2)
create_dataset_resp = Dataset.create_dataset(session, dataset_name=dataset_name,
dataset_type=dataset_type, data_sources=data_sources, work_path=work_path, schema=schemas)
```

Parameters

Table 7-2 Request parameters

Name	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
dataset_name	Yes	String	Dataset name

Name	Mandatory	Type	Description
data_type	No	String	Data type of a dataset. Either **data_type** or **dataset_type** must be specified. **data_type** is recommended. The options are as follows: • **IMAGE**: image • **TEXT**: text • **AUDIO**: audio • **TABLE**: table • **VIDEO**: video • **PLAIN**: custom format
dataset_type	No	Integer	Obtain a dataset list based on the dataset type. Either **data_type** or **dataset_type** must be specified. The options are as follows: • **0**: image classification • **1**: object detection • **3**: image segmentation • **100**: text classification • **101**: named entity recognition • **102**: text triplet • **200**: sound classification • **201**: speech content • **202**: speech paragraph labeling • **400**: table dataset • **600**: video labeling • **900**: custom format

Name	Mandatory	Type	Description
data_sources	Yes	Table 7-3	Input dataset path, which is used to synchronize source data (such as images, text files, and audio files) in the directory and its subdirectories to the dataset. For a table dataset, this parameter indicates the import directory. The work directory of a table dataset cannot be an OBS path in a KMS-encrypted bucket.
work_path	Yes	Table 7-7	Output dataset path, which is used to store output files such as label files.
labels	No	List of Table 7-8	Dataset labels. This parameter must be imported when you create a text triplet dataset.
schema	No	List of Table 7-10	Schema list, which is used to specify the name and type of the table header of a table dataset
description	No	String	Dataset description consisting of 0 to 256 characters without special characters (^!<>=&'"). The parameter is left blank by default.

Table 7-3 DataSource parameters

Name	Mandatory	Type	Description
type	Yes	Integer	Data type. The options are as follows: • **0**: OBS bucket (default value) • **5**: Dataset downloaded from AI Gallery

Name	Mandatory	Type	Description
path	Yes	String	Data source path • Newline characters (\n), carriage return characters (\r), and tab characters (\t) are not allowed.
content_info	No	Table 7-4	Dataset asset downloaded from the AI Gallery
annotation_config	No	Table 7-5	Data labeling format, which can be: • Image classification • Object detection • Text classification • Sound classification
with_column_header	No	Boolean	Whether the first row of a table is the table header. This parameter is mandatory for table datasets. • **True**: The first row of a table is used as the table header. • **False**: The first row of a table is not used as the table header, but only as sample data.

Table 7-4 ContentInfo parameters

Name	Mandatory	Type	Description
content_id	Yes	String	Dataset asset ID in AI Gallery
version_id	Yes	String	Dataset asset version ID in AI Gallery

Table 7-5 AnnotationConfig parameters

Name	Mandatory	Type	Description
scene	Yes	String	Supported labeling scenarios. The options are as follows: • **image_classification** • **object_detection** • **text_classification** • **audio_classification**
format_name	Yes	String	Labeling format in different scenarios. The options are as follows: • **image_classification** – ModelArts imageNet 1.0 – ModelArts image classification 1.0 • **object_detection** – ModelArts PASCAL VOC 1.0 – YOLO • **text_classification** – ModelArts text classification 1.0 – ModelArts text classification combine 1.0 • **audio_classification** – ModelArts audio classification dir 1.0
parameters	No	**Table 7-6**	Advanced labeling format parameters, such as the sample separator

Table 7-6 AnnotationConfigParam parameters

Name	Mandatory	Type	Description
included_labels	No	List of **Table 7-8**	Import only samples with specified labels.

Name	Mandatory	Type	Description
sample_label_separator	No	String	Separator between text and labels. The separator contains only one character, which must be a letter, digit, or one of the following characters (@#¥%^&*_=|?/':.;,). The separator must be escaped.
label_separator	No	String	Separator between labels. The separator contains only one character, which must be a letter, digit, or one of the following characters (@#¥%^&*_=|?/':.;,). The separator must be escaped.
difficult_only	No	Boolean	Whether to import only hard examples.

Table 7-7 WorkPath parameters

Parameter	Mandatory	Type	Description
type	Yes	Integer	Data type. The options are as follows: • **0**: OBS bucket (default value)

Parameter	Mandatory	Type	Description
path	Yes	String	Output dataset path, which is used to store output files such as label files. • The format is "/Bucket name/File path", for example, **/obs-bucket/flower/rose/** (directory used as the path). • A bucket cannot be used as a path. • The output path must be different from the input path and its subdirectories. • The parameter consists of 3 to 700 characters. • Newline characters (\n), carriage return characters (\r), and tab characters (\t) are not allowed.

Table 7-8 Label parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Label name
type	Yes	Integer	Label type. The options are as follows: • **0**: image classification • **1**: object detection • **3**: image segmentation • **100**: text classification • **101**: named entity • **102**: text triplet relationship • **200**: sound classification • **201**: speech content • **202**: speech paragraph labeling • **600**: video labeling
property	No	Table 7-9	Basic attribute key-value pair of a label, such as color

Table 7-9 LabelProperty parameters

Parameter	Mandatory	Type	Description
@modelarts:color	No	String	(Built-in attribute) Label color, which is a hexadecimal code of the color. By default, this parameter is left blank. For example, **#FFFFF0**.
@modelarts:from_type	No	String	(Built-in attribute) Type of the head entity in a triplet relationship label. This attribute must be specified when a relationship label is created. This parameter is only used in text triplet datasets.
@modelarts:to_type	No	String	(Built-in attribute) Type of the tail entity in a triplet relationship label. This attribute must be specified when a relationship label is created. This parameter is only used in text triplet datasets.

Table 7-10 Schema parameters

Parameter	Mandatory	Type	Description
schema_id	No	Integer	Schema ID
name	No	String	Schema name

Parameter	Mandatory	Type	Description
type	No	String	Schema value type. The options are as follows: • STRING • SHORT • INT • LONG • DOUBLE • FLOAT • BYTE • DATE • TIMESTAMP • BOOLEAN
description	No	String	Schema description

7.1.3 Querying Details About a Dataset

Obtain details about a dataset, including the samples and versions of the dataset.

```
dataset.get_dataset_info()
```

Sample Code

Obtain details about a dataset.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
dataset_info = dataset.get_dataset_info()
print(dataset_info) # Output the detailed information about the dataset.
```

Parameters

None

7.1.4 Modifying a Dataset

Change the name or modify the description of a dataset.

```
dataset.update_dataset(dataset_name=None, description=None)
```

Sample Code

Change a dataset name.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()
```



```
dataset = Dataset(session, dataset_id)
dataset.update_dataset(dataset_name = "new-dataset-name")
```

Parameters

Table 7-11 Request parameters

Parameter	Mandatory	Type	Description
dataset_name	No	String	Dataset name
description	No	String	Dataset description

7.1.5 Deleting a Dataset

Delete a dataset based on the dataset ID.

```
delete_dataset(session, dataset_id)
```

Sample Code

Delete a dataset.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

Dataset.delete_dataset(session, dataset_id="68ZXdK6CZwgvUICOOdC")
```

Parameters

Table 7-12 Request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
dataset_id	Yes	String	Dataset ID

7.2 Managing Dataset Versions

7.2.1 Obtaining a Dataset Version List

Obtain a list of dataset versions.

```
dataset.list_versions()
```

Sample Code

Obtain a dataset version list.


```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
version_list = dataset.list_versions()
print(version_list) # Print the dataset version list.
```

Parameters

None

7.2.2 Creating a Dataset Version

Create a new version for a dataset.

```
dataset.create_version(name=None, version_format=None, label_task_type=None, label_task_id=None,
**kwargs)
```

Sample Code

Example 1: Create a new version for a dataset.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
create_version_resp = dataset.create_version(name="V001", version_format="Default", label_task_type=0,
description="version 001")
```

Example 2: Create a dataset based on a labeling task.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
create_version_resp = dataset.create_version(label_task_id="IbAhFai5KXWC3gthUfz", description="dataset
version from label task")
```

Parameters

Table 7-13 Request parameters

Parameter	Mandatory	Type	Description
name	No	String	Version name that consists of 1 to 32 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed.
version_format	No	String	Format of a dataset version. The options are as follows: • **Default**

Parameter	Mandatory	Type	Description
label_task_type	No	Integer	Labeling type of a dataset version. The options are as follows: • **0**: image classification • **1**: object detection • **3**: image segmentation • **100**: text classification • **101**: named entity recognition • **102**: text triplet • **200**: sound classification • **201**: speech content • **202**: speech paragraph labeling • **400**: table dataset • **600**: video labeling • **900**: custom format
label_task_id	No	String	ID of a labeling task based on which a dataset version is created.
description	No	String	Version description consisting of 0 to 256 characters without special characters (!<>=&"'). The parameter is left blank by default.

7.2.3 Querying Details About a Dataset Version

Obtain details about a dataset version based on the version ID.

```
dataset.get_version_info(version_id)
```

Sample Code

Obtain details about a dataset version.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
version_info = dataset.get_version_info(version_id="cSPuXPgnYp7ObRs6LaR")
print(version_info) # Print details about the dataset version.
```


Parameters

Table 7-14 Request parameters

Parameter	Mandatory	Type	Description
version_id	Yes	String	Dataset version ID

7.2.4 Deleting a Dataset Version

Delete a specified dataset version.

```
dataset.delete_version(version_id)
```

Sample Code

Delete a specified dataset version.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
dataset.delete_version(version_id="cSPuXPgnYp7ObRs6LaR")
```

Parameters

Table 7-15 Request parameters

Parameter	Mandatory	Type	Description
version_id	Yes	String	Dataset version ID

7.3 Managing Samples

7.3.1 Querying a Sample List

Obtain the sample list of a dataset. Table datasets are not supported.

```
dataset.list_samples(version_id=None, offset=None, limit=None)
```

Sample Code

- Example 1: Obtain a dataset sample list.

  ```
  from modelarts.session import Session
  from modelarts.dataset import Dataset
  session = Session()

  dataset = Dataset(session, dataset_id)
  list_samples_resp = dataset.list_samples()
  print(list_samples_resp) # Print the sample list.
  ```


- Example 2: Obtain the sample list of a specified dataset version.

```
list_samples_resp = dataset.list_samples(version_id = "cSPuXPgnYp7ObRs6LaR")
print(list_samples_resp)
```

Parameters

Table 7-16 Request parameters

Parameter	Mandatory	Type	Description
version_id	No	String	Dataset version ID, which can be used for obtaining the sample list of this dataset version.
offset	No	Integer	Start page for pagination display. The default value is **0**.
limit	No	Integer	Maximum number of records returned on each page. The value ranges from 1 to 100. The default value is **10**.

7.3.2 Querying Details About a Sample

Obtain details about a specified sample in a dataset based on the sample ID.

```
dataset.get_sample_info(sample_id)
```

Sample Code

Obtain details about a specified sample in a dataset based on the sample ID.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
sample_info = dataset.get_sample_info(sample_id="2551e78974aed9b60156d8376232f6bd")
print(sample_info) # Print the detailed information about the sample.
```

Parameters

Table 7-17 Request parameters

Parameter	Mandatory	Type	Description
sample_id	Yes	String	Sample ID

7.3.3 Deleting Samples in a Batch

Delete samples from a dataset in a batch based on the sample ID list.

```
dataset.delete_samples(samples)
```


Sample Code

Delete samples from a dataset in a batch.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
samples = []
samples.append("2551e78974aed9b60156d8376232f6bd")
samples.append("0d315fec1efc7568de5cccf522c10a1b")
dataset.delete_samples(samples)
```

Parameters

Table 7-18 Request parameters

Parameter	Mandatory	Type	Description
samples	Yes	List of String	IDs of the samples to be deleted

7.4 Managing Dataset Import Tasks

7.4.1 Querying a Dataset Import Task List

Obtain a dataset import task list.

```
dataset.list_import_tasks()
```

Sample Code

Obtain a dataset import task list.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
list_tasks_resp = dataset.list_import_tasks()
print(list_tasks_resp) # Print the import task list.
```

Parameters

None

7.4.2 Creating a Dataset Import Task

You can import new data from OBS through an OBS path or a manifest file.

```
dataset.import_data(path=None,  anntation_config=None, **kwargs)
```

Table 7-19 lists the import modes supported by datasets.

Table 7-19 Import modes supported by datasets

Dataset Type	From an OBS Path	From a Manifest File	Remarks
Image classification	Supported	Supported	None
Object detection	Supported	Supported	None
Image segmentation	Supported	Supported	None
Text classification	Supported	Supported	None
Named entity recognition	Not supported	Supported	None
Text triplet	Not supported	Supported	None
Sound classification	Supported	Supported	None
Speech labeling	Not supported	Supported	None
Speech paragraph labeling	Not supported	Supported	None
Table dataset	Supported	Not supported	The schema of the newly imported table data is the same as that of the dataset.
Video labeling	Not supported	Supported	None

Sample Code

- Example 1: Import an object detection dataset from an OBS path.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
annotation_config = dict()
annotation_config['scene'] = "object_detection"
annotation_config['format_name'] = "ModelArts PASCAL VOC 1.0"
import_resp = dataset.import_data(path="/obs-gaia-test/data/image/image-detection/",
annotation_config=annotation_config)
```

- Example 2: Import an object detection dataset from a manifest file.

```
annotation_config = dict()    # Task with data imported from a manifest file. annotation_config is used to import labels.
import_resp = dataset.import_data(
        path="/obs-gaia-test/data/output/work_path/dataset-5932-Qdd1RUZ3wqBQrwrTr3v/annotation/V001/V001.manifest",annotation_config=annotation_config)
```

- Example 3: Import a table dataset from an OBS path.

```
import_resp = dataset.import_data(
        path="/obs-gaia-test/data/table/table1/", with_column_header=True)
```


Parameters

Table 7-20 Request parameters

Parameter	Mandatory	Type	Description
path	Yes	String	OBS path or manifest file path for importing data • If data is to be imported from a manifest file, ensure the manifest file is specified in the path. • If data is to be imported from an OBS path, ensure only image classification, object detection, image segmentation, text classification, sound classification, and table datasets are supported. • Newline characters (\n), carriage return characters (\r), and tab characters (\t) are not allowed.

Parameter	Mandatory	Type	Description
annotation_config	No	Table 7-5	Data labeling format. If this parameter is set to **None**, no labels will be imported. If data is to be imported from a manifest file, import an empty dict object so that labels can be imported. The following labeling formats are supported: • Image classification • Object detection • Sound classification • Text classification
with_column_header	No	Boolean	Whether the first row of a table is the table header. This parameter is mandatory for table datasets. • **True**: The first row of a table is used as the table header. • **False**: The first row of a table is not used as the table header, but only as sample data.

7.4.3 Querying the Status of a Dataset Import Task

Obtain the status and details of a dataset import task based on the task ID.

```
dataset.get_import_task_info(task_id)
```


Sample Code

Obtain details about a dataset import task.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
task_info = dataset.get_import_task_info(task_id="r4R52nJ4VJKcivuioCU")
print(task_info) # Print the detailed information about the import task.
```

Parameters

Table 7-21 Request parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	ID of an import task

7.5 Managing Export Tasks

7.5.1 Querying a Dataset Export Task List

Obtain a dataset export task list.

```
dataset.list_export_tasks()
```

Sample Code

Obtain a dataset export task list.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
list_tasks_resp = dataset.list_export_tasks()
print(list_tasks_resp) # Print the export task list.
```

Parameters

None

7.5.2 Creating a Dataset Export Task

Export the samples of a dataset to a specified OBS path. This function is only supported by image classification, object detection, image segmentation, and custom format datasets.

```
dataset.export_data(path)
```

Sample Code

Export the samples of a dataset to an OBS path.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
```



```
session = Session()

dataset = Dataset(session, dataset_id)
export_resp = dataset.export_data("/obs-gaia-test/data/output/export-test/")
```

Parameters

Table 7-22 Request parameters

Parameter	Mandatory	Type	Description
path	Yes	String	OBS path for storing the exported data

7.5.3 Querying the Status of a Dataset Export Task

Obtain the status and details of a dataset export task based on the task ID.

```
dataset.get_export_task_info(task_id)
```

Sample Code

Obtain the status of a dataset export task.

```
from modelarts.session import Session
from modelarts.dataset import Dataset
session = Session()

dataset = Dataset(session, dataset_id)
task_info = dataset.get_export_task_info(task_id="iuHALF6xdkSAGKVN2jD")
print(task_info) # Print the detailed information about the export task.
```

Parameters

Table 7-23 Request parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	ID of an export task

7.6 Managing Manifest Files

7.6.1 Overview of Manifest Management

When using ModelArts, perform operations such as labeling data, training a model, performing inference, managing datasets, and publishing a model in AI Gallery. All these operations are based on datasets. To standardize the use of datasets in various application scenarios and ensure the flexibility of dataset management, this document describes the manifest file with dataset management APIs and specifications included.

- A manifest file defines the mapping between labeled objects and content. The manifest file can contain only unlabeled data, for example, a created dataset that has not been labeled.
- A manifest file is encoded using UTF-8. Therefore, the programs processing manifest must support UTF-8.
- In a manifest file, the text classification source allows non-English characters.
- You can create a manifest file or obtain such a file using a third-party tool or ModelArts.
- Any valid file name is allowed for a manifest file.

7.6.2 Parsing a Manifest File

Parse a manifest file in either a local or OBS path. If an OBS path is used, a session is required.

```
manifest.parse_manifest(manifest_path, encoding='utf-8')
```

Sample Code

Parse a manifest file.

```
from modelarts.session import Session
from modelarts.dataset.format.manifest import Manifest

path = "obs://your-obs-bucket/manifest/V001.manifest"
session = Session()
manifest_info= Manifest.parse_manifest(path,session=session)
```

Parameters

Table 7-24 Request parameters

Parameter	Mandatory	Type	Description
manifest_path	Yes	String	Path for storing a manifest file, which can be a local path or an OBS path. If an OBS path is used, a session is required.
encoding	No	String	File encoding format, which defaults to UTF-8.

Table 7-25 manifest_info parameters

Parameter	Type	Description
size	Long	Number of samples.
samples	JSON Array	Sample list. For details, see **Table 7-26**.

Table 7-26 sample parameters

Parameter	Type	Description
source	String	URI of the labeled object. Supported schemes are **OBS**, **HTTPS**, and **Content**. **Content** indicates text, for example, "source": "s3://path-to-jpg" and "source": " content://I love machine learning".
annotations	JSON Array	Sample labels. If this parameter is not specified, the object is not labeled. The **annotations** value is an object list. For details, see **Table 7-27**.
usage	String	What an object is used for, which can be training (**TRAIN**), evaluation (**EVAL**), test (**TEST**), or inference (**INFERENCE**). If this parameter is not specified, you can determine how to use the object.
inference_loc	String	Location of an inference result file. This parameter is available if a manifest file is generated in an inference service.
id	String	Sample ID.
source_type	String	Source type, for example, **csv**.
source_property	String	Attribute of the source.
hard	Boolean	Hard example or not. **true** for hard examples and **false** for not.
hard_coefficient	Double	Difficulty coefficient, ranging from 0 to 1.
hard_reasons	String	Label-level hard example reasons. Use a hyphen (-) to separate reason IDs of a hard example.
source_map	String	Source mapping.

Table 7-27 annotation parameters

Parameter	Type	Description
name	String	Label name
type	String	Label type
id	String	Label ID
annotation_loc	String	Location where a labeled file is stored. This parameter is mandatory only for object detection labeled files.

Parameter	Type	Description
annotation_property	String	Label properties
confidence	Double	Confidence of machine labeling, which is a numeral ranging from 0 to 1
creation_time	String	Time when a label was created, which is the time when the label was written, not the time when the manifest file was generated
annotated_by	String	Annotator
annotation_format	String	Format of a labeled file, which defaults to **PASCAL VOC**
hard	Boolean	Hard example
hard_coefficient	Double	Difficulty level
annotation_loc_map	String	Mapping of the path for storing a labeled file

7.6.3 Creating and Saving a Manifest File

Create an object that contains the manifest information and save the object. For details about the manifest information, see **Table 7-25**. The path can be either a local or OBS path. If an OBS path is used, a session is required.

```
manifest_info.save(path, session=None, save_mode="w")
```

Sample Code

Before saving a manifest file, create an object that contains the manifest information, including the samples and their labels, and then combine the samples into the manifest file. Call the **save** API to save the imported session in a specified path.

```
from modelarts.dataset.format.manifest.annotation import Annotation
from modelarts.dataset.format.manifest import Manifest
from modelarts.dataset.format.manifest.sample import Sample
from modelarts.session import Session

size = 0
sample_list = []
for i in range(19):
    size = size + 1
    source = "s3://obs-path/examples/image-classification/data/image_" + str(i) + ".jpg"
    usage = "TRAIN"
    inference_loc = "s3://obs-path/examples/image-classification/data/image_" + str(i) + ".txt"
    annotations_list = []

    for j in range(1):
        annotation_type = "modelarts/image_classification"
        if 0 == i % 2:
            annotation_name = "Bees"
        else:
            annotation_name = "Rabbits"
        annotation_creation_time = "2019-02-20 08:23:06"
```



```
        annotation_format = "manifest"
        annotation_property = {"color": "black"}
        annotation_confidence = 0.8
        annotated_by = "human"
        annotations_list.append(
            Annotation(name=annotation_name, type=annotation_type,
                       confidence=annotation_confidence,
                       creation_time=annotation_creation_time,
                       annotated_by=annotated_by, annotation_format=annotation_format,
                       annotation_property=annotation_property))
    sample_list.append(
        Sample(source=source, usage=usage, annotations=annotations_list, inference_loc=inference_loc))
manifest_info = Manifest(samples=sample_list, size=size)

path = "obs://your-obs-bucket/manifest/V001.manifest"
session = Session()
manifest_info.save(path, session=session, save_mode="a")
```

Parameters

Table 7-28 Request parameters

Parameter	Mandatory	Type	Description
path	Yes	String	Path for storing a manifest file
session	No	Object	Session object. For details about the initialization method, see **Session Authentication**. This parameter is mandatory when OBS is used.
save_mode	No	String	Save mode. The default value is **w**, indicating rewriting. Value **a** indicates appending.

7.6.4 Parsing a Pascal VOC File

Parse an XML file in either a local or OBS path. If an OBS path is used, a session is required.

```
PascalVoc.parse_xml(xml_file_path, session=None)
```

Sample Code

Specify an XML file path and call **parse_xml** to parse the file.

```
from modelarts.dataset.format.voc.pascal_voc import PascalVoc
from modelarts.session import Session

path = "obs://your-obs-bucket/voc/test.xml"
session = Session()
pascal_voc = PascalVoc.parse_xml(path, session=session)
print(pascal_voc) # Print the parsing result.
```


Parameters

Table 7-29 Request parameters

Parameter	Mandatory	Type	Description
xml_file_path	Yes	String	XML file path
session	No	Object	Session object. For details about the initialization method, see **Session Authentication**. This parameter is mandatory when OBS is used.

Table 7-30 pascal_voc parameters

Parameter	Type	Description
folder	String	Folder name
file_name	String	File name
source	Object	Data source. For details, see **Table 7-31**.
width	Long	Image width
height	Long	Image height
depth	Long	Image depth
segmented	String	Segmentation
mask_source	String	Path for storing the mask file generated after image segmentation. Only PNG images are supported.
voc_objects	JSON Array	Labeled objects. For details, see **Table 7-32**.

Table 7-31 source parameters

Parameter	Type	Description
database	String	Dataset name, for example, **The VOC2007 Database**
annotation	String	Label, for example, **PASCAL VOC2007**
image	String	Image information

Table 7-32 voc_object parameters

Parameter	Type	Description
name	String	Folder name
properties	JSON Array	Properties of a labeled object in key-value pairs. Both key and value are of the string type.
pose	String	Shooting angle of labeled data
truncated	String	Whether a labeled object is truncated (**0** indicates the object is not truncated.)
occluded	String	Whether a labeled object is occluded (**0** indicates the object is not occluded.)
difficult	String	Whether a labeled object is difficult to identify (**0** indicates that the object is easy to identify.)
confidence	Double	Confidence of machine labeling, which is a numeral ranging from 0 to 1
position	Object	Location of a labeled object. For details, see **Table 7-33**.
parts	Object	Built-in voc_object list. For details, see **Table 7-32**.
mask_color	String	Color of the mask image for image segmentation

Table 7-33 Position parameters

Parameter	Shape	Labeling Information
point	Point	Coordinates of a point <x>100<x> <y>100<y>
line	Line	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>200<y2>
bndbox	Rectangle	Coordinates of the lower left and upper right points <x_min>100<x_min> <y_min>100<y_min> <x_max>200<x_max> <y_max>200<y_max>

Parameter	Shape	Labeling Information
polygon	Polygon	Coordinates of points <x1>100<x1> <y1>100<y1> <x2>200<x2> <y2>100<y2> <x3>250<x3> <y3>150<y3> <x4>200<x4> <y4>200<y4> <x5>100<x5> <y5>200<y5> <x6>50<x6> <y6>150<y6>
circle	Circle	Center coordinates and radius <cx>100<cx> <cy>100<cy> <r>50<r>

7.6.5 Creating and Saving a Pascal VOC File

Create an object that contains the Pascal VOC information and save the object. For details about Pascal VOC, see **Table 7-30**. The path can be either a local or OBS path. If an OBS path is used, a session is required.

```
pascal_voc.save_xml(xml_file_path, save_mode='w', session=None)
```

Sample Code

Before saving a Pascal VOC XML file, create an object that contains the Pascal VOC information, including the VOC object. Call the **save_xml** API to save the imported session in a specified path.

```
from modelarts.dataset.format.voc.pascal_voc import PascalVoc
from modelarts.dataset.format.voc.voc_object import VocObject
from modelarts.session import Session

path = "obs://your-obs-bucket/voc/test2.xml"
size_list = [640, 321, 3]
file_name = "000000089955.jpg"
voc_object_tags = ["trafficlight", "trafficlight"]
voc_object_properties = [{"@modelarts:color": "#FFFFF0", "@modelarts:shortcut": "C",
                          "pose": "0", "truncated": "0", "difficult": "0",
                          "@modelarts:shape": "bndbox", "@modelarts:feature": [[347, 186], [382, 249]]},
                         {"@modelarts:color": "#FFFFE0", "@modelarts:shortcut": "D",
                          "pose": "0", "truncated": "0", "difficult": "0",
                          "@modelarts:shape": "bndbox", "@modelarts:feature": [[544, 50], [591, 149]]}]
voc_objects = []
for i in range(len(voc_object_tags)):
    object_tag = voc_object_tags[i]
```

```
    object_properties = voc_object_properties[i]
    voc_objects.append(VocObject(name=object_tag, properties=object_properties))

pascal_voc = PascalVoc(file_name=file_name, width=size_list[0], height=size_list[1], depth=size_list[2],
                       voc_objects=voc_objects)
session = Session()
pascal_voc.save_xml(path, session=session)
```

Parameters

Table 7-34 Request parameters

Parameter	Mandatory	Type	Description
xml_file_path	Yes	String	Path for storing a Pascal VOC XML file.
session	No	Object	Session object. For details about the initialization method, see **Session Authentication**. This parameter is mandatory when OBS is used.
save_mode	No	String	Save mode. The default value is **w**, indicating rewriting. Value **a** indicates appending.

7.7 Managing Labeling Jobs

7.7.1 Creating a Labeling Job

Create a labeling job based on a dataset.

```
dataset.create_label_task(self, task_name=None, task_type=None, **kwargs)
```

Sample Code

Example 1: Create an object detection labeling job based on an image dataset.

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()
dataset = Dataset(session, dataset_id="VukxA2FlaTUm7tkDtq0") # Initialize the dataset.
create_task_resp = dataset.create_label_task(task_name="obj_detection_task", task_type=1,
description="label task")
```


Parameters

Table 7-35 Request parameters

Parameter	Mandatory	Type	Description
task_name	Yes	String	Name of a labeling job
task_type	Yes	Integer	Type of a labeling job. Options: • **0**: image classification • **1**: object detection • **3**: image segmentation • **100**: text classification • **101**: named entity recognition • **102**: text triplet • **200**: sound classification • **201**: speech content • **202**: speech paragraph labeling • **400**: table dataset • **600**: video labeling • **900**: custom format
description	No	String	Description of a labeling job

7.7.2 Obtaining the Labeling Job List of a Dataset

Obtain the labeling job list of a dataset.

```
dataset.get_label_tasks(is_workforce_task=False, **kwargs)
```

Sample Code

- Example 1: Obtain all labeling jobs of a dataset and sort the jobs by creation time in descending order.

```
from modelarts.session import Session
from modelarts.dataset import Dataset

session = Session()
dataset = Dataset(session,dataset_id="VukxA2FlaTUm7tkDtq0")
list_label_task_resp = dataset.get_label_tasks(sort_key="create_time", sort_dir="desc")
print(list_label_task_resp)
```

- Example 2: Obtain all team labeling jobs of a dataset.

```
list_label_task_resp = dataset.get_label_tasks(is_workforce_task=True)
print(list_label_task_resp)
```


Parameters

Table 7-36 Request parameters

Parameter	Mandatory	Type	Description
is_workforce_task	No	Boolean	Filter criteria, specifying whether to obtain only team labeling jobs • **True**: Only team labeling jobs are obtained. • **False**: Obtain all labeling jobs. This is the default value.
sort_key	No	String	Field for sorting. Options: • **create_time**: Sort jobs by creation time. • **task_name**: Sort jobs by job name.
sort_dir	No	String	Sorting method. Options: • **asc**: Labeling jobs are sorted in ascending order. • **desc**: Labeling jobs are sorted in descending order. This is the default value.

7.7.3 Obtaining Details About a Labeling Job

Obtain details about a labeling job.

```
datset.get_label_task_info(task_id=None)
```

Sample Code

Obtain details about a labeling job.

```
task_info = dataset.get_label_task_info(task_id="xs9ZKzLluKzccQfsyi2")
print(task_info)
```

Parameters

Table 7-37 Request parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	ID of a labeling job

8 Training Management (New Version)

8.1 Training Jobs

8.1.1 Creating a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

NOTE

ModelArts SDK cannot be used to create training jobs using algorithms subscribed to in AI Gallery.

- Example 1: **Create a training job using a common AI engine.**

  If both **framework_type** and **framework_version** are specified in estimator, a training job will be created using a common AI engine.

```
from modelarts.session import Session
from modelarts.train_params import TrainingFiles
from modelarts.train_params import OutputData
from modelarts.train_params import InputData
from modelarts.estimatorV2 import Estimator
session = Session()
# Parameters received in the training script (set based on the site requirements):

parameters = [{"name": "mod", "value":"gpu"},
              {"name": "epoc_num", "value":2}]
estimator = Estimator(session=session,
                      training_files=TrainingFiles(code_dir= "obs://bucket_name/code_dir/",
boot_file="boot_file.py"),
                      outputs=[OutputData(obs_path="obs://bucket_name/output/", name="output_dir")],
                      parameters=parameters,
                      framework_type='PyTorch',                    # Common AI engine
                      framework_version='PyTorch-1.4.0-python3.6',   # Version of the AI engine
                      train_instance_type="modelarts.p3.large.public",
                      train_instance_count=1,
                      log_url="obs://bucket_name/log/",
                      env_variables={"USER_ENV_VAR": "customize environment variable"},
                      working_dir="/home/ma-user/modelarts/user-job-dir",
```



```
                    local_code_dir="/home/ma-user/modelarts/user-job-dir",
                    job_description='This is an image net train job')
job_instance = estimator.fit(inputs=[InputData(obs_path="obs://bucket_name/input/",
name="data_url")],
                         job_name="job_name_1")
```

- Example 2: **Create a training job using a custom image.**

  If both **user_image_url** and **user_command** are specified in estimator, a training job will be created using a custom image and started using a custom boot command.

```
from modelarts.session import Session
from modelarts.train_params import TrainingFiles
from modelarts.train_params import OutputData
from modelarts.train_params import InputData
from modelarts.estimatorV2 import Estimator
session = Session()
# Parameters received in the training script (set based on the site requirements):

parameters = [{"name": "mod", "value":"gpu"},
              {"name": "epoc_num", "value":2}]
estimator = Estimator(session=session,
                    training_files=TrainingFiles(code_dir= "obs://bucket_name/code_dir/",
boot_file="boot_file.py"),
                    outputs=[OutputData(obs_path="obs://bucket_name/output/", name="output_dir")],
                    parameters=parameters,
                    user_image_url="sdk-test/pytorch1_4:1.0.1",     # URL of the custom image
                    user_command="/home/ma-user/anaconda3/envs/PyTorch-1.4/bin/python /home/ma-
user/modelarts/user-job-dir/train/test-pytorch.py",  # Custom boot command
                    train_instance_type="modelarts.p3.large.public",
                    train_instance_count=1,
                    log_url="obs://bucket_name/log/",
                    env_variables={"USER_ENV_VAR": "customize environment variable"},
                    working_dir="/home/ma-user/modelarts/user-job-dir",
                    local_code_dir="/home/ma-user/modelarts/user-job-dir",
                    job_description='This is an image net train job')
job_instance = estimator.fit(inputs=[InputData(obs_path="obs://bucket_name/input/",
name="data_url")],
                         job_name="job_name_2")
```

- Example 3: **Creating a training job in a dedicated resource pool**

```
from modelarts.session import Session
from modelarts.train_params import TrainingFiles
from modelarts.train_params import OutputData
from modelarts.train_params import InputData
from modelarts.estimatorV2 import Estimator
session = Session()
# Parameters received in the training script (set based on the site requirements):

parameters = [{"name": "mod", "value":"gpu"},
              {"name": "epoc_num", "value":2}]
estimator = Estimator(session=session,
                    training_files=TrainingFiles(code_dir= "obs://bucket_name/code_dir/",
boot_file="boot_file.py"),
                    outputs=[OutputData(obs_path="obs://bucket_name/output/", name="output_dir")],
                    parameters=parameters,
                    framework_type='PyTorch',
                    framework_version='PyTorch-1.4.0-python3.6',
                    pool_id="your pool id",                              # Dedicated resource pool ID
                    train_instance_type="modelarts.pool.visual.xlarge",     # VM flavor of the dedicated
pool
                    train_instance_count=1,
                    log_url="obs://bucket_name/log/",
                    env_variables={"USER_ENV_VAR": "customize environment variable"},
                    working_dir="/home/ma-user/modelarts/user-job-dir",
                    local_code_dir="/home/ma-user/modelarts/user-job-dir",
                    job_description='This is an image net train job')
job_instance = estimator.fit(inputs=[InputData(obs_path="obs://bucket_name/input/",
name="data_url")],
                         job_name="job_name_3")
```


- Example 4: **Create a training job using a dataset.**

```
from modelarts.session import Session
from modelarts.train_params import TrainingFiles
from modelarts.train_params import OutputData
from modelarts.train_params import InputData
from modelarts.estimatorV2 import Estimator
session = Session()
# Parameters received in the training script (set based on the site requirements):
parameters = [{"name": "model_name", "value":"s"},
              {"name": "batch-size", "value": 32},
              {"name": "epochs", "value":100},
              {"name": "img-size", "value":"640,640"} ]
estimator = Estimator(session=session,
                      training_files=TrainingFiles(code_dir= "obs://bucket_name/code_dir/",
boot_file="boot_file.py"),
                      outputs=[OutputData(obs_path="obs://bucket_name/output/", name="output_dir")],
                      parameters=parameters,
                      framework_type='PyTorch',                    # Common AI engine
                      framework_version='PyTorch-1.4.0-python3.6',    # Version of the AI engine
                      train_instance_type="modelarts.p3.large.public",
                      train_instance_count=1,
                      log_url="obs://bucket_name/log/",
                      working_dir="/home/ma-user/modelarts/user-job-dir",
                      local_code_dir="/home/ma-user/modelarts/user-job-dir",
                      job_description='This is an image net train job')
job_instance = estimator.fit(dataset_id="your dataset id",
                             dataset_version_id="your dataset version id",
                             job_name="job_name_5")
```

Parameters

Table 8-1 Estimator request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
training_files	No	**TrainingFiles** Object	Path to the training script in OBS. For details, see **Table 8-2**.
outputs	No	Array of **OutputData** objects	Training output path. For details, see **Table 8-3**.
parameters	No	JSON Array	Running parameters of a training job. The format is as follows: [{"name":"your name", "value": "your value"}]. The value can be a string or an integer.
train_instance_type	Yes	String	Resource flavor selected for a training job. For details, see **Obtaining Resource Flavors**.
train_instance_count	Yes	Int	Number of compute nodes in a training job

Parameter	Mandatory	Type	Description
framework_type	No	String	Engine type selected for a training job. For details, see **Obtaining Engine Types**.
framework_version	No	String	Engine version selected for a training job. For details, see **Obtaining Engine Types**.
user_image_url	No	String	SWR URL of the custom image used by a training job
user_command	No	String	Command for starting a training job created using a custom image
log_url	No	String	OBS path for storing training job logs, for example, **obs://xx/yy/zz/**
local_code_dir	No	String	Local directory to the training container to which the algorithm code directory is downloaded. Note: • The directory must be under **/home**. • In v1 compatibility mode, this parameter does not take effect. • When **code_dir** is prefixed with **file://**, this parameter does not take effect.
working_dir	No	String	Work directory where an algorithm is executed. Note that this parameter does not take effect in v1 compatibility mode.
job_description	No	String	Description of a training job

volumes	No	JSON Array	Information of the disks attached for a training job in the following example format:
			[{
			"nfs": {
			"local_path": "/xx/yy/zz",
			"read_only": False,
			"nfs_server_path": "xxx.xxx.xxx.xxx:/"
			}
			}]

env_variables	No	Dict	Environment variables of a training job

Parameter	Mandatory	Type	Description
pool_id	No	String	ID of the resource pool for a training job. To obtain the ID, do as follows: Log in to the ModelArts management console, choose **Dedicated Resource Pools** in the navigation pane on the left, and view the resource pool ID in the dedicated resource pool list.

Table 8-2 Parameters for initializing **TrainingFiles**

Parameter	Mandatory	Type	Description
code_dir	Yes	String	Code directory of a training job, which is an OBS path and must start with **obs:/**, for example, **obs://xx/yy/**
boot_file	Yes	String	Boot file of a training job, which must be stored in the code directory. You can enter a relative path, for example, **boot_file.py**, or an absolute path, for example, **obs://xx/yy/boot_file.py**.

Table 8-3 Parameters for initializing **OutputData**

Parameter	Mandatory	Type	Description
obs_path	Yes	String	OBS path to which data is exported
name	Yes	String	Keyword parameter name of the output data, for example, **output_dir**

Table 8-4 fit request parameters

Parameter	Mandatory	Type	Description
inputs	No	Array of **InputData** Object	Input data of a training job stored in OBS Either **inputs** or **dataset_id**/**dataset_version_id** can be configured.

Parameter	Mandatory	Type	Description
wait	No	Boolean	Whether to wait for the completion of a training job. It defaults to **False**.
job_name	No	String	Name of a training job
show_log	No	Boolean	Whether to output training job logs after a job is submitted. It defaults to **False**.
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.

Table 8-5 Parameters for initializing **InputData**

Parameter	Mandatory	Type	Description
obs_path	Yes	String	OBS path to the dataset required by a training job, for example, **obs://xx/yy/**
name	Yes	String	Keyword parameter name of the input data, for example, **data_url**.

Table 8-6 Response for creating a training job

Parameter	Type	Description
TrainingJob	Object	Training object, which contains attributes such as **job_id**. When you perform operations on a training job, for example, obtain information of, update, or delete a training job, you can use **job_instance.job_id** to obtain the ID of the training job.

Table 8-7 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.2 Debugging a Training Job

8.1.2.1 Using the SDK to Debug a Multi-Node Distributed Training Job

Replace the OBS paths in the debugging code with your OBS paths.

PyTorch is used to write debugging code in this document. The process is the same for different AI frameworks. You only need to change the **framework_type** value in **Step 7** and **Step 11**. For example, set **framework_type** to **Ascend-Powered-Engine** for MindSpore.

Step 1 Initialize session. This step is the same as that of **debugging a single-node training job**.

Step 2 Prepare training data. This step is the same as that of **debugging a single-node training job**. The only difference is that **obs_path** must be set here.

Step 3 Prepare the training script.

```
from modelarts.train_params import TrainingFiles
code_dir = os.path.join(base_local_path, "train/")

# The training script has been stored in OBS. The training script can be chosen from any source as long as it can be stored in a notebook instance.

session.obs.download_file(os.path.join(base_bucket_path, "train/test-pytorch.py"), code_dir)
training_file = TrainingFiles(code_dir=code_dir, boot_file="test-pytorch.py", obs_path=base_bucket_path + 'train/')
```

Parameters:

- **code_dir**: Code directory where a training script is stored. The directory must be a notebook directory for local debugging. This parameter is mandatory.
- **boot_file**: Training boot file, which is stored in the **code_dir** directory. This parameter is mandatory.
- **obs_path**: OBS directory. This parameter is mandatory for multi-node distributed debugging. The SDK zips the notebook directory **code_dir** and uploads the ZIP file to **obs_path**.

Step 4 Prepare the training output. This step is the same as **Step 4** for debugging a single-node training job.

Step 5 Check the AI frameworks available for training. This step is the same as **Step 5** for debugging a single-node training job.

Step 6 Save the current notebook instance as a new image. This step is the same as **Step 9** for debugging a single-node training job.

Step 7 Initialize the Estimator.

```
from modelarts.estimatorV2 import Estimator
parameters = []
parameters.append({"name": "data_url", "value": data_local})
parameters.append({"name": "output_dir", "value": os.path.join(base_local_path, "output/")})
parameters.append({"name": "epoc_num", "value": 2})
# For Boolean, use parser.add_argument('--dist', action='store_true') in the boot script for parsing. If the parameter is set to True, the parameter is transferred in the format of the following lines of code.
parameters.append({"name": "dist"})
estimator = Estimator(session=session,
                      training_files=training_file,
                      outputs=[output],
                      parameters=parameters,
                      framework_type='PyTorch',
                      train_instance_type='local',
                      train_instance_count=2,
                      script_interpreter="/home/ma-user/anaconda3/envs/PyTorch-1.4/bin/python",
                      log_url=base_bucket_path + 'log/',
                      job_description='This is a image net train job')
```

Parameters:

- **session**: Initialized data in **Step 1**. This parameter is mandatory.
- **training_files**: Initialized training files in **Step 3**. This parameter is mandatory.
- **outputs**: A list of training outputs. Each element in the list is a training output initialized in **Step 4**. This parameter is optional.
- **parameters**: A list of parameters. Each element in the list is a dictionary that contains the **name** and **value** fields, which are transferred to the training boot file in the form of **-name=value**. This parameter is optional. **value** can be a string, an integer, or a Boolean. For Boolean, use **action='store_true'** in the training script for parsing.
- **framework_type**: Type of the AI framework used for a training job. For details, see the output item in **Step 5**. This parameter is mandatory.
- **train_instance_type**: Type of training instance. If this parameter is set to **local**, the training job is performed in a notebook instance. This parameter is mandatory.
- **train_instance_count**: Number of workers in a training job. Set this parameter to **2** for distributed debugging. When the training job starts, the SDK creates another notebook instance to form a 2-node distributed debugging environment with the current instance. This parameter is mandatory.
- **script_interpreter**: Python environment used for a training job. If this parameter is not set, the current kernel is used by default. This parameter is optional.
- **log_url**: OBS address. The SDK automatically uploads training logs to this address during local training. This parameter must be set only when training jobs run on Ascend.
- **job_description**: describes a training job. This parameter is optional.

Step 8 Start training.


```
estimator.fit(inputs=[input_data], job_name="cifar10-dis")
```

Parameters:

- **inputs**: A list of training inputs. Each element in the list is an input imported in **Step 2**. This parameter is optional.
- **job_name**: Name of a training job. This parameter is optional.

After a local distributed training job starts, the SDK automatically performs the following operations:

1. Zips the training script and uploads the ZIP file to **obs_path** specified in **Step 3**.
2. Zips the data and uploads the ZIP file to the specified **obs_path** if the training data is stored in .
3. Creates another instance to form a two-worker environment for distributed training.
4. Initializes the training job and downloads data to **local_path**.
5. Executes the training job and saves the training outputs in **local_path** specified in **Step 4**.
6. Uploads the training output to **obs_path** specified in **Step 4** and the logs to **log_url** specified in **Step 7**.

Step 9 Perform debugging. This step is the same as **Step 8** for debugging a single-node training job.

Step 10 Obtain the type of compute nodes available for training. This step is the same as **Step 9** for debugging a single-node training job.

Step 11 Submit the remote training job.

```
from modelarts.estimatorV2 import Estimator
parameters = []
parameters.append({"name": "data_url", "value": data_local})
parameters.append({"name": "output_dir", "value": os.path.join(base_local_path, "output/")})
parameters.append({"name": "epoc_num", "value": 2})
# For Boolean, use parser.add_argument('--dist', action='store_true') in the boot script for parsing. If the parameter is set to True, the parameter is transferred in the format of the following lines of code.
parameters.append({"name": "dist"})
estimator = Estimator(session=session,
                      training_files=training_file,
                      outputs=[output],
                      parameters=parameters,
                      framework_type='PyTorch',
                      train_instance_type='modelarts.p3.large.public.distributed',
                      train_instance_count=2,
                      script_interpreter="/home/ma-user/anaconda3/envs/PyTorch-1.4/bin/python",
                      log_url=base_bucket_path + 'log/',
                      job_description='This is a image net train job')
estimator.fit(inputs=[input_data], job_name="cifar10-dis-1")
```

The difference between Estimator initialization and local training lies in the **train_instance_type** parameter. Configure this parameter based on the obtained result in **Step 10**. The value of **train_instance_count** depends on the value of **max_num** in **Step 10**.

After the training job is submitted, the SDK automatically performs the following operations:

1. Zips the training script and uploads the ZIP file to **obs_path** specified in **Step 3**.

2. Zips the data and uploads the ZIP file to the specified **obs_path** if the training data is stored in .
3. Submits the training job to the ModelArts training service. The image of the current instance is used to execute the training job.
4. Uploads the training output to **obs_path** specified in **Step 4** and the logs to the location specified by **log_url**.

   In this step, note the following:

   If you want to create a directory or file in your training script, create it in the following directories:

   (1) **/home/ma-user/work**

   (2) **/cache**

   (3) **local_path** specified in inputs or outputs. For example, if **local_path** is set to **/home/ma-user/work/xx/yy/** during InputData initialization in **Step 2**, you can create directories or files in this directory.

----End

8.1.2.2 Using the SDK to Debug a Single-Node Training Job

Replace the OBS paths in the debugging code with your OBS paths.

PyTorch is used to write debugging code in this document. The process is the same for different AI frameworks. You only need to change the **framework_type** value in **Step 6** and **Step 10**. For example, set **framework_type** to **Ascend-Powered-Engine** for MindSpore.

Step 1 Initialize session.

The following is the sample code.

```
from modelarts.session import Session
session = Session()
```

Step 2 Prepare training data. Three data formats are supported. You can select one of them as required.

```
import os
from modelarts.train_params import InputData
base_bucket_path = "obs://modelarts-xxx-a0de02a6/dis-train/cifar10/"
base_local_path = "/home/ma-user/work/cifar10/"

# Format 1: The data is stored in a compressed file in OBS.
obs_path = os.path.join(base_bucket_path, "dataset-zip/dataset.zip")
data_local = os.path.join(base_local_path, "dataset/")
input_data = InputData(obs_path=obs_path, local_path=data_local, is_local_source=False)

# Format 2: The data is stored in a directory in OBS.
#obs_path = os.path.join(base_bucket_path, "dataset/")
#data_local = os.path.join(base_local_path, "dataset/")
#input_data = InputData(obs_path=obs_path, local_path=data_local, is_local_source=False)

# Format 3: The data is stored in a directory in an SFS system mounted to a notebook instance.
#obs_path = os.path.join(base_bucket_path, "dataset-local/")
#data_local = os.path.join(base_local_path, "dataset/")
#input_data = InputData(obs_path=obs_path, local_path=data_local, is_local_source=True)
```

Parameters:

- **is_local_source**: Location where the training data is stored. The default value is **False** and this parameter is optional.

  - **False**: The training data is stored in the path specified by **obs_path**.
  - **True**: The training data is stored in a notebook instance, which is specified by **local_path**.
- **obs_path**: OBS path. It depends on the value of **is_local_source**.
  - If **is_local_source** is set to **False**, this parameter is mandatory, indicating the location where the training data is stored, which can be a folder or a compressed file.
  - If **is_local_source** is set to **True**, this parameter is optional. If you set this parameter, the training data in the notebook instance is compressed and uploaded to the location. The data cannot be uploaded repeatedly. After data is uploaded for the first time, change **is_local_source** to **False** and set **obs_path** to the location where the compressed file was uploaded. If you do not set this parameter, the file will not be compressed and uploaded.
- **local_path**: Notebook path. Your training script reads data from this path for training. This parameter is mandatory. It depends on the value of **is_local_source**.
  - If **is_local_source** is set to **True**, this parameter indicates the location where the training data is stored, which can be a folder.
  - If **is_local_source** is set to **False**, the SDK downloads data to this location during training. If the training data is compressed files, they will be decompressed after being downloaded.

Step 3 Prepare the training script.

```
from modelarts.train_params import TrainingFiles
code_dir = os.path.join(base_local_path, "train/")

# The training script has been stored in OBS. The training script can be chosen from any source as long as it can be stored in a notebook instance.

session.obs.download_file(os.path.join(base_bucket_path, "train/test-pytorch.py"), code_dir)
training_file = TrainingFiles(code_dir=code_dir, boot_file="test-pytorch.py", obs_path=base_bucket_path + 'train/')
```

Parameters:

- **code_dir**: Code directory where a training script is stored. The directory must be a notebook directory for debugging a training job. This parameter is mandatory.
- **boot_file**: Path of the training boot file. Enter the relative path of **code_dir**. For example, if the absolute path of **boot_file** is **/home/ma-user/work/cifar10/train/test-pytorch.py**, set this parameter to **test-pytorch.py**. This parameter is mandatory.
- **obs_path**: OBS path. This parameter must be set only for remote training. The training script is compressed and uploaded to this path.

Step 4 Prepare the training output. If you do not need to upload the training output to OBS, skip this step.

```
from modelarts.train_params import OutputData
output = OutputData(local_path=os.path.join(base_local_path, "output/"),
obs_path=os.path.join(base_bucket_path, 'output/'))
```

- **local_path**: Notebook path, in which the trained model or other training script data is stored.

- **obs_path**: OBS path. The SDK automatically uploads the model file in **local_path** to this OBS path. This parameter is mandatory.

Step 5 Check the AI frameworks that can be used for training.

```
from modelarts.estimatorV2 import Estimator
Estimator.get_framework_list(session)
```

session is the initialized data in **Step 1**. Skip this step if the AI framework has been specified.

Step 6 Initialize the Estimator.

```
from modelarts.estimatorV2 import Estimator
parameters = []
parameters.append({"name": "data_url", "value": data_local})
parameters.append({"name": "output_dir", "value": os.path.join(base_local_path, "output/")})
parameters.append({"name": "epoc_num", "value": 2})
estimator = Estimator(session=session,
                      training_files=training_file,
                      outputs=[output],
                      parameters=parameters,
                      framework_type='PyTorch',
                      train_instance_type='local',
                      train_instance_count=1,
                      script_interpreter="/home/ma-user/anaconda3/envs/PyTorch-1.4/bin/python",
                      log_url=base_bucket_path + 'log/',
                      job_description='This is a image net train job')
```

Parameters:

- **session**: Initialized data in **Step 1**. This parameter is mandatory.
- **training_files**: Initialized training files in **Step 3**. This parameter is mandatory.
- **outputs**: A list of training outputs. Each element in the list is an initialized training output in **Step 4**. This parameter is optional.
- **parameters**: A list of parameters. This parameter is optional. Each element in the list is a dictionary that contains the **name** and **value** fields, which are transferred to the training boot file in the form of **--name=value**. **value** can be a string, an integer, or a Boolean. For Boolean, use **action='store_true'** in the training script for parsing.
- **framework_type**: Type of the AI framework used for a training job. For details, see the output item in **Step 5**. This parameter is mandatory.
- **train_instance_type**: Training instance type. If this parameter is set to **local**, the training job is performed in a notebook instance. This parameter is mandatory.
- **train_instance_count**: Number of workers in a training job. Set this parameter to **1** for single-node training. The training job runs only in the current notebook instance. This parameter is mandatory.
- **script_interpreter**: Python environment used for a training job. If this parameter is not set, the current kernel is used by default. This parameter is optional.
- **log_url**: OBS address. The SDK automatically uploads training logs to this address during training. This parameter must be set only when training jobs run on Ascend.
- **job_description**: describes a training job. This parameter is optional.

Step 7 Start training.

```
estimator.fit(inputs=[input_data], job_name="cifar10-dis")
```


Parameters:

- **inputs**: A list of training inputs. Each element in the list is an input imported in **Step 2**. This parameter is optional.
- **job_name**: Name of a training job. This parameter is optional.

After a local single-node training job starts, the SDK automatically performs the following operations:

1. Initializes the training job. If the training data imported in **Step 2** is stored in OBS, the data is downloaded to **local_path**.
2. Executes the training job and saves the training outputs in **local_path** specified in **Step 4**.
3. Uploads the training output to **obs_path** specified in **Step 4** and the logs to **log_url** specified in **6**.

In addition, time suffixes are added to the job names.

```
from datetime import datetime, timedelta
import time
base_name = "cifar10-dis"
job_name = base_name + '-' + (datetime.now() + timedelta(hours=8)).strftime('%Y%m%d-%H%M%S')
estimator.fit(inputs=[input_data], job_name=job_name)
```

Step 8 Perform debugging.

In the previous step, the logs of the training script are printed to the console in real time. You can easily detect incorrect code or parameters in the logs. Perform debugging in until you obtain a desired result, then you can go to the next step.

Step 9 Obtain the type and maximum number of compute nodes available for training.

```
from modelarts.estimatorV2 import Estimator
Estimator.get_spec_list(session=session)
```

session is the initialized data in **Step 1**. A dictionary is returned. **flavors** is a list that describes all flavors available for training. **flavor_id** of each element indicates the compute flavors that can be directly used for remote training jobs, and **max_num** indicates the maximum number of compute nodes of the flavors. Skip this step if the compute flavor has been specified.

Step 10 Submit the remote training job.

```
from modelarts.estimatorV2 import Estimator
parameters = []
parameters.append({"name": "data_url", "value": data_local})
parameters.append({"name": "output_dir", "value": os.path.join(base_local_path, "output/")})
parameters.append({"name": "epoch_num", "value": 2})
estimator = Estimator(session=session,
                      training_files=training_file,
                      outputs=[output],
                      parameters=parameters,
                      framework_type='PyTorch',
                      train_instance_type='modelarts.vm.cpu.8u',
                      train_instance_count=1,
                      script_interpreter="/home/ma-user/anaconda3/envs/PyTorch-1.4/bin/python",
                      log_url=base_bucket_path + 'log/',
                      job_description='This is a image net train job')
estimator.fit(inputs=[input_data], job_name="cifar10-dis")
```

After the local debugging is complete, you only need to change **train_instance_type** to the value of **flavor_id** obtained in **Step 9** during Estimator

initialization. After the **fit** function is executed, you can submit the remote training job.

After the training job is submitted, the SDK automatically performs the following operations:

1. Zips the training script and uploads the ZIP file to **obs_path** specified in **Step 3**.
2. Zips the data and uploads the ZIP file to the specified **obs_path** if the training data is stored in .
3. Submits the training job created using a custom image to ModelArts. The image is that of the current instance. This ensures that the environment of the remote training job is the same as that of the training job in the instance.
4. Uploads the training output to **obs_path** specified in **Step 4** and the logs to the location specified by **log_url** in this step.

   In this step, note the following:

   If you want to create a directory or file in your training script, create it in the following directories:

   - **/home/ma-user/work**
   - **/cache**
   - **local_path** specified in inputs or outputs. For example, if **local_path** is set to **/home/ma-user/work/xx/yy/** during InputData initialization in **Step 2**, you can create directories or files in this directory.

----End

8.1.3 Obtaining Training Jobs

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
job_list = Estimator.get_job_list(session=session, offset=10, limit=5, sort_by="create_time", order="asc",
                                  filters=[{"key": "name", "operator": "like", "value": ["trainjob"]}])

print(job_list)
```

Parameters

Table 8-8 get_job_list request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Parameter	Mandatory	Type	Description
offset	No	Integer	Offset for obtaining training jobs. The minimum value is **0**. For example, if this parameter is set to **1**, the query starts from the second one.
limit	No	Integer	Maximum number of training jobs to be obtained. The value ranges from **1** to **50**.
sort_by	No	String	Metric for sorting obtained training jobs. By default, training jobs are sorted by creation time (**create_time**).
order	No	String	Order of obtained training jobs. The default value is **desc**, indicating the descending order. You can also set this parameter to **asc**, indicating the ascending order. Default value: **desc** Options: • **asc**: The query results are displayed in ascending order. • **desc**: The query results are displayed in descending order.
group_by	No	String	Condition for grouping the obtained training jobs.
filters	No	Array of objects	Filter criteria for obtaining training jobs.

Table 8-9 filters

Parameter	Mandatory	Type	Description
key	No	String	Key of the grouping condition.
operator	No	String	The key-value relationship of a grouping condition. Default value: **in** Options: • **like**: similar • **in**: included • **not**: not included • **between**: a range
value	No	Array of strings	Value of the grouping condition key.

Table 8-10 get_job_list response parameters

Parameter	Type	Description
total	Integer	Total number of training jobs of the current user.
count	Integer	Total number of training jobs that meet the search criteria of the current user.
limit	Integer	Maximum number of training jobs to be obtained. The value ranges from **1** to **50**.
offset	Integer	Offset for obtaining training jobs. The minimum value is **0**. For example, if this parameter is set to **1**, the query starts from the second one.
sort_by	String	Metric for sorting obtained training jobs. By default, training jobs are sorted by creation time (**create_time**).
order	String	Order of obtained training jobs. The default value is **desc**, indicating the descending order. You can also set this parameter to **asc**, indicating the ascending order.
group_by	String	Condition for grouping the obtained training jobs.
workspace_id	String	Workspace where a training job is deployed. The default value is **0**.
ai_project	String	AI project to which a training job belongs. The default value is **default-ai-project**.
items	Array of **JobResponse** objects	Details of the training jobs that meet the search criteria of the current user.

Table 8-11 JobResponse

Parameter	Type	Description
kind	String	Training job type, which defaults to **job**. Options: • **job**: training job • **hetero_job**: heterogeneous job • **autosearch_job**: auto search job • **mrs_job**: MRS job • **edge_job**: edge job

Parameter	Type	Description
metadata	**JobMetadata** object	Metadata of a training job.
status	**Status** object	Status of a training job. When creating a training job, you do not need to set this parameter.
algorithm	**JobAlgorithmResponse** object	Algorithm used by a training job. The following formats are supported: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
tasks	Array of **TaskResponse** objects	Tasks of a heterogeneous training job.
spec	**spec** object	Specifications of a training job.

Table 8-12 JobMetadata

Parameter	Type	Description
id	String	Training job ID, which is generated and returned by ModelArts after a training job is created.
name	String	Name of a training job. The value must contain 1 to 64 characters consisting of only digits, letters, underscores (_), and hyphens (-).
workspace_id	String	Workspace where a training job is deployed. Default value: **0**
description	String	Description of a training job, which defaults to **NULL**. The value must contain 0 to 256 characters.
create_time	Long	Time when a training job was created, in milliseconds. The value is generated and returned by ModelArts after a training job is created.
user_name	String	Username for creating a training job. The username is generated and returned by ModelArts after a training job is created.

Parameter	Type	Description
annotations	Map<String,String>	Declaration template of a training job. For heterogeneous jobs, the default value of **job_template** is **Template RL**. For other jobs, the default value is **Template DL**.

Table 8-13 Status

Parameter	Type	Description
phase	String	Level-1 status of a training job. The value will remain unchanged. Options: **Creating**, **Pending**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, and **Abnormal**
secondary_phase	String	Level-2 status of a training job. The value can be changed. Options: **Creating**, **Queuing**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, **CreateFailed**, **TerminatedFailed**, **Unknown**, and **Lost**
duration	Long	Running duration of a training job, in milliseconds
node_count_metrics	Array<Array<Integer>>	Node count changes during the runtime of a training job
tasks	Array of strings	Tasks of a training job
start_time	String	Start time of a training job. The value is in timestamp format.
task_statuses	Array of objects	Status of a training job task

Table 8-14 task_statuses

Parameter	Type	Description
task	String	Task of a training job
exit_code	Integer	Exit code of a training job task
message	String	Error message of a training job task

Table 8-15 JobAlgorithmResponse

Parameter	Type	Description
id	String	Algorithm ID Options: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
name	String	Algorithm name
subscription_id	String	Subscription ID of the subscribed algorithm, which must be used with **item_version_id**
item_version_id	String	Version ID of the subscribed algorithm, which must be used with **subscription_id**
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used with **boot_file**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
boot_file	String	Boot file of a training job, which must be stored in the code directory, for example, **/usr/app/boot.py**. This parameter must be used with **code_dir**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
autosearch_config_path	String	YAML configuration path of an auto search job. An OBS URL is required.
autosearch_framework_path	String	Framework code directory of an auto search job. An OBS URL is required.
command	String	Boot command for starting the container of the custom image used for creating a training job. The value of this parameter can be the same as the **code_dir** value.
parameters	Array of **Parameter** objects	Running parameters of a training job.
policies	**policies** object	Policies supported by a training job.

Parameter	Type	Description
inputs	Array of **Input** objects	Input of a training job.
outputs	Array of **Output** objects	Output of a training job.
engine	**engine** object	Engine of a training job. Leave this parameter blank if the job is created using **id** of the algorithm in algorithm management, or **subscription_id** and **item_version_id** of the subscribed algorithm.
environments	Map<String,String>	Environment variables of a training job in the format of "key":"value". Leave this parameter blank.

Table 8-16 Parameter

Parameter	Type	Description
name	String	Parameter name
value	String	Parameter value
description	String	Parameter description
constraint	**constraint** object	Parameter constraint
i18n_description	**i18n_description** object	Internationalization description

Table 8-17 constraint

Parameter	Type	Description
type	String	Parameter type
editable	Boolean	Whether the parameter is editable
required	Boolean	Whether the parameter is mandatory

Parameter	Type	Description
sensitive	Boolean	Whether the parameter is sensitive
valid_type	String	Valid type
valid_range	Array of strings	Valid range

Table 8-18 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 8-19 policies

Parameter	Type	Description
auto_search	**auto_search** object	Hyperparameter search configuration

Table 8-20 auto_search

Parameter	Type	Description
skip_search_params	String	Hyperparameter parameters that need to be skipped
reward_attrs	Array of objects	Search metrics
search_params	Array of objects	Search parameters
algo_configs	Array of objects	Search algorithm configurations

Table 8-21 reward_attrs

Parameter	Type	Description
name	String	Metric name
mode	String	Search mode • **max**: A larger metric value is preferred. • **min**: A smaller metric value is preferred.
regex	String	Regular expression of a metric

Table 8-22 search_params

Parameter	Type	Description
name	String	Hyperparameter name
param_type	String	Parameter type • **continuous**: Parameter values are continuous. • **discrete**: Parameter values are discrete.
lower_bound	String	Lower bound of the hyperparameter
upper_bound	String	Upper bound of the hyperparameter
discrete_points_num	String	Number of discrete points of a hyperparameter with continuous values
discrete_values	Array of strings	Discrete hyperparameter values

Table 8-23 algo_configs

Parameter	Type	Description
name	String	Name of the search algorithm
params	Array of **AutoSearchAlgoConfigParameter** objects	Search algorithm parameters

Table 8-24 AutoSearchAlgoConfigParameter

Parameter	Type	Description
key	String	Parameter key
value	String	Parameter value
type	String	Parameter type

Table 8-25 Input

Parameter	Type	Description
name	String	Name of the data input channel
description	String	Description of the data input channel
local_dir	String	Local directory of the container to which the data input channel is mapped
remote	**InputDataInfo** object	Information of the data input
remote_constraint	Array of objects	Data input constraint

Table 8-26 InputDataInfo

Parameter	Type	Description
dataset	**dataset** object	Dataset as the data input
obs	**obs** object	OBS in which data input and output are stored

Table 8-27 dataset

Parameter	Type	Description
id	String	Dataset ID of a training job

Parameter	Type	Description
version_id	String	Dataset version ID of a training job
obs_url	String	OBS URL of the dataset for a training job, which is automatically parsed by ModelArts based on the dataset ID and dataset version IDs, for example, **/usr/data/**

Table 8-28 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-29 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location and dataset
attributes	String	Attributes when a dataset functions as the data input Options: • **data_format**: data format • **data_segmentation**: data segmentation • **dataset_type**: data labeling

Table 8-30 Output

Parameter	Type	Description
name	String	Name of the data output channel
description	String	Description of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output

Table 8-31 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-32 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-33 engine

Parameter	Type	Description
engine_id	String	Engine ID selected for a training job, which can be **engine_id**, **engine_name** and **engine_version**, or **image_url**
engine_name	String	Name of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
engine_version	String	Version of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
image_url	String	Custom image URL selected for a training job

Table 8-34 TaskResponse

Parameter	Type	Description
role	String	Role of a heterogeneous training job task Options: • **learner**: GPUs or CPUs are supported. • **worker**: CPUs are supported.
algorithm	**algorithm** object	Algorithm configurations in algorithm management

Parameter	Type	Description
task_resource	**FlavorResponse** object	Flavors for a training job or an algorithm

Table 8-35 algorithm

Parameter	Type	Description
code_dir	String	Absolute path of the directory where the algorithm boot file is stored
boot_file	String	Absolute path of the algorithm boot file
inputs	**inputs** object	Algorithm input channel
outputs	**outputs** object	Algorithm output channel
engine	**engine** object	Engine on which a heterogeneous job depends

Table 8-36 inputs

Parameter	Type	Description
name	String	Name of the data input channel
local_dir	String	Local path of the container to which the data input and output channels are mapped
remote	**remote** object	Actual data input, which can only be OBS for heterogeneous jobs

Table 8-37 remote

Parameter	Type	Description
obs	**obs** object	OBS in which data input and output are stored

Table 8-38 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-39 outputs

Parameter	Type	Description
name	String	Name of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output
mode	String	Data transmission mode, which defaults to **upload_periodically**
period	String	Data transmission period, which defaults to **30s**

Table 8-40 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-41 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-42 engine

Parameter	Type	Description
engine_id	String	Engine ID of a heterogeneous job, for example, **caffe-1.0.0-python2.7**

Parameter	Type	Description
engine_name	String	Engine name of a heterogeneous job, for example, **Caffe**
engine_version	String	Engine version of a heterogeneous job
v1_compatible	Boolean	Whether v1 is compatible
run_user	String	User UID for which the engine is started by default

Table 8-43 FlavorResponse

Parameter	Type	Description
flavor_id	String	ID of the resource flavor
flavor_name	String	Name of the resource flavor
max_num	Integer	Maximum number of nodes with the resource flavor
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details
attributes	Map<String,String>	Other flavor attributes

Table 8-44 billing

Parameter	Type	Description
code	String	Billing code

Parameter	Type	Description
unit_num	Integer	Number of billing units

Table 8-45 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.
cpu	**cpu** object	CPU specifications
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information

Table 8-46 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-47 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-48 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name
memory	String	Memory

Table 8-49 memory

Parameter	Type	Description
size	Integer	Memory size
unit	String	Number of memory units

Table 8-50 spec

Parameter	Type	Description
resource	**Resource** object	Resource flavors of a training job, which can either be **flavor_id** or **pool_id** and **flavor_id**
volumes	Array of objects	Volumes attached for a training job
log_export_path	**log_export_path** object	Export path of training job logs

Table 8-51 Resource

Parameter	Type	Description
policy	String	Resource flavor mode of a training job. Options: **regular**, **economic**, and **turbo**
flavor_id	String	Resource flavor ID of a training job
flavor_name	String	Read-only flavor name returned by ModelArts when **flavor_id** is specified

Parameter	Type	Description
node_count	Integer	Number of resource replicas selected for a training job Minimum value: **1**
pool_id	String	Resource pool ID selected for a training job
flavor_detail	**flavor_detail** object	Flavors for a training job or an algorithm

Table 8-52 flavor_detail

Parameter	Type	Description
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details

Table 8-53 billing

Parameter	Type	Description
code	String	Billing code
unit_num	Integer	Number of billing units

Table 8-54 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.

Parameter	Type	Description
cpu	**cpu** object	CPU specifications
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 8-55 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-56 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-57 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name

Parameter	Type	Description
memory	String	Memory

Table 8-58 memory

Parameter	Type	Description
size	Integer	Memory size
unit	String	Number of memory units

Table 8-59 disk

Parameter	Type	Description
size	String	Disk size
unit	String	Unit of the disk size, which is GB generally

Table 8-60 volumes

Parameter	Type	Description
nfs	**nfs** object	Disks attached in NFS mode

Table 8-61 nfs

Parameter	Type	Description
nfs_server_path	String	NFS server path
local_path	String	Path for attaching disks to the training container
read_only	Boolean	Whether the disks attached to the container in NFS mode are read-only

Table 8-62 log_export_path

Parameter	Type	Description
obs_url	String	OBS URL for storing training job logs
host_path	String	Path of the host where training job logs are stored

Table 8-63 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

Table 8-64 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.4 Obtaining the Details About a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
```



```
estimator = Estimator(session=session, job_id="618222c4-dc2f-4cfe-bc49-72b075b7552f")
job_info = estimator.get_job_info()
print(job_info)
```

- Method 2: Use the training job created in **Creating a Training Job**.

```
job_info = job_instance.get_job_info()
print(job_info)
```

Parameters

Table 8-65 Estimator request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

Table 8-66 get_job_info response parameters

Parameter	Type	Description
kind	String	Training job type, which defaults to **job**. Options: • **job**: training job • **hetero_job**: heterogeneous job • **autosearch_job**: auto search job • **mrs_job**: MRS job • **edge_job**: edge job
metadata	**JobMetadata** object	Metadata of a training job.
status	**Status** object	Status of a training job. When creating a training job, you do not need to set this parameter.

Parameter	Type	Description
algorithm	JobAlgorithmResponse object	Algorithm used by a training job. The following formats are supported: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
tasks	Array of TaskResponse objects	Tasks of a heterogeneous training job.
spec	spec object	Specifications of a training job.

Table 8-67 JobMetadata

Parameter	Type	Description
id	String	Training job ID, which is generated and returned by ModelArts after a training job is created.
name	String	Name of a training job. The value must contain 1 to 64 characters consisting of only digits, letters, underscores (_), and hyphens (-).
workspace_id	String	Workspace where a training job is deployed. Default value: **0**
description	String	Description of a training job, which defaults to **NULL**. The value must contain 0 to 256 characters.
create_time	Long	Time when a training job was created, in milliseconds. The value is generated and returned by ModelArts after a training job is created.
user_name	String	Username for creating a training job. The username is generated and returned by ModelArts after a training job is created.
annotations	Map<String, String>	Declaration template of a training job. For heterogeneous jobs, the default value of **job_template** is **Template RL**. For other jobs, the default value is **Template DL**.

Table 8-68 Status

Parameter	Type	Description
phase	String	Level-1 status of a training job. The value will remain unchanged. Options: **Creating**, **Pending**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, and **Abnormal**
secondary_phase	String	Level-2 status of a training job. The value can be changed. Options: **Creating**, **Queuing**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, **CreateFailed**, **TerminatedFailed**, **Unknown**, and **Lost**
duration	Long	Running duration of a training job, in milliseconds
node_count_metrics	Array<Array<Integer>>	Node count changes during the runtime of a training job
tasks	Array of strings	Task of a training job
start_time	String	Start time of a training job. The value is in timestamp format.
task_statuses	Array of objects	Status of a training job task

Table 8-69 task_statuses

Parameter	Type	Description
task	String	Task of a training job
exit_code	Integer	Exit code of a training job task
message	String	Error message of a training job task

Table 8-70 JobAlgorithmResponse

Parameter	Type	Description
id	String	Algorithm ID Options: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.

Parameter	Type	Description
name	String	Algorithm name
subscription_id	String	Subscription ID of the subscribed algorithm, which must be used with **item_version_id**
item_version_id	String	Version ID of the subscribed algorithm, which must be used with **subscription_id**
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used with **boot_file**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
boot_file	String	Boot file of a training job, which must be stored in the code directory, for example, **/usr/app/boot.py**. This parameter must be used with **code_dir**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
autosearch_config_path	String	YAML configuration path of an auto search job. An OBS URL is required.
autosearch_framework_path	String	Framework code directory of an auto search job. An OBS URL is required.
command	String	Boot command for starting the container of the custom image used for creating a training job. The value of this parameter can be the same as the **code_dir** value.
parameters	Array of **Parameter** objects	Running parameters of a training job.
policies	**policies** object	Policies supported by a training job.
inputs	Array of **Input** objects	Input of a training job.
outputs	Array of **Output** objects	Output of a training job.
engine	**engine** object	Engine of a training job. Leave this parameter blank if the job is created using **id** of the algorithm in algorithm management, or **subscription_id** and **item_version_id** of the subscribed algorithm.

Parameter	Type	Description
environments	Map<String,String>	Environment variables of a training job in the format of "key":"value". Leave this parameter blank.

Table 8-71 Parameter

Parameter	Type	Description
name	String	Parameter name
value	String	Parameter value
description	String	Parameter description
constraint	**constraint** object	Parameter constraint
i18n_description	**i18n_description** object	Internationalization description

Table 8-72 constraint

Parameter	Type	Description
type	String	Parameter type
editable	Boolean	Whether the parameter is editable
required	Boolean	Whether the parameter is mandatory
sensitive	Boolean	Whether the parameter is sensitive
valid_type	String	Valid type
valid_range	Array of strings	Valid range

Table 8-73 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 8-74 policies

Parameter	Type	Description
auto_search	**auto_search** object	Hyperparameter search configuration

Table 8-75 auto_search

Parameter	Type	Description
skip_search_params	String	Hyperparameter parameters that need to be skipped
reward_attrs	Array of objects	Search metrics
search_params	Array of objects	Search parameters
algo_configs	Array of objects	Search algorithm configurations

Table 8-76 reward_attrs

Parameter	Type	Description
name	String	Metric name
mode	String	Search mode • **max**: A larger metric value is preferred. • **min**: A smaller metric value is preferred.
regex	String	Regular expression of a metric

Table 8-77 search_params

Parameter	Type	Description
name	String	Hyperparameter name
param_type	String	Parameter type • **continuous**: Parameter values are continuous. • **discrete**: Parameter values are discrete.
lower_bound	String	Lower bound of the hyperparameter
upper_bound	String	Upper bound of the hyperparameter

Parameter	Type	Description
discrete_points_num	String	Number of discrete points of a hyperparameter with continuous values
discrete_values	Array of strings	Discrete hyperparameter values

Table 8-78 algo_configs

Parameter	Type	Description
name	String	Name of the search algorithm
params	Array of **AutoSearchAlgoConfigParameter** objects	Search algorithm parameters

Table 8-79 AutoSearchAlgoConfigParameter

Parameter	Type	Description
key	String	Parameter key
value	String	Parameter value
type	String	Parameter type

Table 8-80 Input

Parameter	Type	Description
name	String	Name of the data input channel
description	String	Description of the data input channel
local_dir	String	Local directory of the container to which the data input channel is mapped
remote	**InputDataInfo** object	Information of the data input
remote_constraint	Array of objects	Data input constraint

Table 8-81 InputDataInfo

Parameter	Type	Description
dataset	**dataset** object	Dataset as the data input
obs	**obs** object	OBS in which data input and output are stored

Table 8-82 dataset

Parameter	Type	Description
id	String	Dataset ID of a training job
version_id	String	Dataset version ID of a training job
obs_url	String	OBS URL of the dataset for a training job, which is automatically parsed by ModelArts based on the dataset ID and dataset version IDs, for example, **/usr/data/**

Table 8-83 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-84 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location and dataset
attributes	String	Attributes when a dataset functions as the data input Options: • **data_format**: data format • **data_segmentation**: data segmentation • **dataset_type**: data labeling

Table 8-85 Output

Parameter	Type	Description
name	String	Name of the data output channel
description	String	Description of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output

Table 8-86 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-87 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-88 engine

Parameter	Type	Description
engine_id	String	Engine ID selected for a training job, which can be **engine_id**, **engine_name** and **engine_version**, or **image_url**
engine_name	String	Name of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
engine_version	String	Version of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
image_url	String	Custom image URL selected for a training job

Table 8-89 TaskResponse

Parameter	Type	Description
role	String	Role of a heterogeneous training job task Options: • **learner**: GPUs or CPUs are supported. • **worker**: CPUs are supported.
algorithm	**algorithm** object	Algorithm configurations in algorithm management
task_resource	**FlavorResponse** object	Flavors for a training job or an algorithm

Table 8-90 algorithm

Parameter	Type	Description
code_dir	String	Absolute path of the directory where the algorithm boot file is stored
boot_file	String	Absolute path of the algorithm boot file
inputs	**inputs** object	Algorithm input channel
outputs	**outputs** object	Algorithm output channel
engine	**engine** object	Engine on which a heterogeneous job depends

Table 8-91 inputs

Parameter	Type	Description
name	String	Name of the data input channel
local_dir	String	Local path of the container to which the data input and output channels are mapped
remote	**remote** object	Actual data input, which can only be OBS for heterogeneous jobs

Table 8-92 remote

Parameter	Type	Description
obs	**obs** object	OBS in which data input and output are stored

Table 8-93 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-94 outputs

Parameter	Type	Description
name	String	Name of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output
mode	String	Data transmission mode, which defaults to **upload_periodically**
period	String	Data transmission period, which defaults to **30s**

Table 8-95 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-96 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-97 engine

Parameter	Type	Description
engine_id	String	Engine ID of a heterogeneous job, for example, **caffe-1.0.0-python2.7**
engine_name	String	Engine name of a heterogeneous job, for example, **Caffe**
engine_version	String	Engine version of a heterogeneous job
v1_compatible	Boolean	Whether v1 is compatible
run_user	String	User UID for which the engine is started by default

Table 8-98 FlavorResponse

Parameter	Type	Description
flavor_id	String	ID of the resource flavor
flavor_name	String	Name of the resource flavor
max_num	Integer	Maximum number of nodes with the resource flavor
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details
attributes	Map<String,String>	Other flavor attributes

Table 8-99 billing

Parameter	Type	Description
code	String	Billing code
unit_num	Integer	Number of billing units

Table 8-100 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.
cpu	**cpu** object	CPU specifications
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information

Table 8-101 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-102 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-103 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name
memory	String	Memory

Table 8-104 memory

Parameter	Type	Description
size	Integer	Memory size
unit	String	Number of memory units

Table 8-105 spec

Parameter	Type	Description
resource	**Resource** object	Resource flavors of a training job, which can either be **flavor_id** or **pool_id** and **flavor_id**
volumes	Array of objects	Volumes attached for a training job
log_export_path	**log_export_path** object	Export path of training job logs

Table 8-106 Resource

Parameter	Type	Description
policy	String	Resource flavor mode of a training job. Options: **regular**, **economic**, and **turbo**
flavor_id	String	Resource flavor ID of a training job
flavor_name	String	Read-only flavor name returned by ModelArts when **flavor_id** is specified
node_count	Integer	Number of resource replicas selected for a training job Minimum value: **1**
pool_id	String	Resource pool ID selected for a training job
flavor_detail	**flavor_detail** object	Flavors for a training job or an algorithm

Table 8-107 flavor_detail

Parameter	Type	Description
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details

Table 8-108 billing

Parameter	Type	Description
code	String	Billing code
unit_num	Integer	Number of billing units

Table 8-109 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.
cpu	**cpu** object	CPU specifications
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 8-110 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-111 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-112 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name
memory	String	Memory

Table 8-113 memory

Parameter	Type	Description
size	Integer	Memory size
unit	String	Number of memory units

Table 8-114 disk

Parameter	Type	Description
size	String	Disk size
unit	String	Unit of the disk size, which is GB generally

Table 8-115 volumes

Parameter	Type	Description
nfs	**nfs** object	Disks attached in NFS mode

Table 8-116 nfs

Parameter	Type	Description
nfs_server_path	String	NFS server path
local_path	String	Path for attaching disks to the training container
read_only	Boolean	Whether the disks attached to the container in NFS mode are read-only

Table 8-117 log_export_path

Parameter	Type	Description
obs_url	String	OBS URL for storing training job logs
host_path	String	Path of the host where training job logs are stored

Table 8-118 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.5 Modifying the Description of a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
estimator = Estimator(session=session, job_id="your job id")
estimator.update_job_configs(description="update job description")
```

- Method 2: Use the training job created in **Creating a Training Job**.

```
job_instance.update_job_configs(description="update job description fourth")
```

Parameters

Table 8-119 Estimator request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

Table 8-120 update_job_configs request parameters

Parameter	Mandatory	Type	Description
description	Yes	String	Description of the training job to be modified

There is no response for successfully calling an API.

Table 8-121 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.6 Deleting a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
Estimator.delete_job_by_id(session=session, job_id="your job id")
```

- Method 2: Use the training job created in **Creating a Training Job**.

```
job_instance.delete_job()
```

Parameters

Table 8-122 delete_job_by_id request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

There is no response for successfully calling an API.

Table 8-123 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.7 Terminating a Training Job

Terminate a training job. Only jobs in the creating, awaiting, or running state can be terminated.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.
```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
info = Estimator.control_job_by_id(session=session, job_id="your job id")
print(info)
```
- Method 2: Use the training job created in **Creating a Training Job**.
```
job_instance.control_job()
```

Parameters

Table 8-124 control_job_by_id request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

Table 8-125 Response parameters

Parameter	Type	Description
kind	String	Training job type, which defaults to **job**. Options: • **job**: training job • **hetero_job**: heterogeneous job • **autosearch_job**: auto search job • **mrs_job**: MRS job • **edge_job**: edge job
metadata	**JobMetadata** object	Metadata of a training job.
status	**Status** object	Status of a training job. When creating a training job, you do not need to set this parameter.
algorithm	**JobAlgorithmResponse** object	Algorithm used by a training job. The following formats are supported: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
tasks	Array of **TaskResponse** objects	Tasks of a heterogeneous training job.
spec	**spec** object	Specifications of a training job.

Table 8-126 JobMetadata

Parameter	Type	Description
id	String	Training job ID, which is generated and returned by ModelArts after a training job is created.
name	String	Name of a training job. The value must contain 1 to 64 characters consisting of only digits, letters, underscores (_), and hyphens (-).
workspace_id	String	Workspace where a training job is deployed. Default value: **0**
description	String	Description of a training job, which defaults to **NULL**. The value must contain 0 to 256 characters.

Parameter	Type	Description
create_time	Long	Time when a training job was created, in milliseconds. The value is generated and returned by ModelArts after a training job is created.
user_name	String	Username for creating a training job. The username is generated and returned by ModelArts after a training job is created.
annotations	Map<String,String>	Declaration template of a training job. For heterogeneous jobs, the default value of **job_template** is **Template RL**. For other jobs, the default value is **Template DL**.

Table 8-127 Status

Parameter	Type	Description
phase	String	Level-1 status of a training job. The value will remain unchanged. Options: **Creating**, **Pending**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, and **Abnormal**
secondary_phase	String	Level-2 status of a training job. The value can be changed. Options: **Creating**, **Queuing**, **Running**, **Failed**, **Completed**, **Terminating**, **Terminated**, **CreateFailed**, **TerminatedFailed**, **Unknown**, and **Lost**
duration	Long	Running duration of a training job, in milliseconds
node_count_metrics	Array<Array<Integer>>	Node count changes during the runtime of a training job
tasks	Array of strings	Task of a training job
start_time	String	Start time of a training job. The value is in timestamp format.
task_statuses	Array of objects	Status of a training job task

Table 8-128 task_statuses

Parameter	Type	Description
task	String	Task of a training job
exit_code	Integer	Exit code of a training job task

Parameter	Type	Description
message	String	Error message of a training job task

Table 8-129 JobAlgorithmResponse

Parameter	Type	Description
id	String	Algorithm ID Options: • **id**: Only the algorithm ID is used. • **subscription_id** and **item_version_id**: The subscription ID and version ID of the algorithm are used. • **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
name	String	Algorithm name
subscription_id	String	Subscription ID of the subscribed algorithm, which must be used with **item_version_id**
item_version_id	String	Version ID of the subscribed algorithm, which must be used with **subscription_id**
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used with **boot_file**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
boot_file	String	Boot file of a training job, which must be stored in the code directory, for example, **/usr/app/boot.py**. This parameter must be used with **code_dir**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.
autosearch_config_path	String	YAML configuration path of an auto search job. An OBS URL is required.
autosearch_framework_path	String	Framework code directory of an auto search job. An OBS URL is required.
command	String	Boot command for starting the container of the custom image used for creating a training job. The value of this parameter can be the same as the **code_dir** value.

Parameter	Type	Description
parameters	Array of **Parameter** objects	Running parameters of a training job.
policies	**policies** object	Policies supported by a training job.
inputs	Array of **Input** objects	Input of a training job.
outputs	Array of **Output** objects	Output of a training job.
engine	**engine** object	Engine of a training job. Leave this parameter blank if the job is created using **id** of the algorithm in algorithm management, or **subscription_id** and **item_version_id** of the subscribed algorithm.
environments	Map<String,String>	Environment variables of a training job in the format of "key":"value". Leave this parameter blank.

Table 8-130 Parameter

Parameter	Type	Description
name	String	Parameter name
value	String	Parameter value
description	String	Parameter description
constraint	**constraint** object	Parameter constraint
i18n_description	**i18n_description** object	Internationalization description

Table 8-131 constraint

Parameter	Type	Description
type	String	Parameter type
editable	Boolean	Whether the parameter is editable
required	Boolean	Whether the parameter is mandatory
sensitive	Boolean	Whether the parameter is sensitive

Parameter	Type	Description
valid_type	String	Valid type
valid_range	Array of strings	Valid range

Table 8-132 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 8-133 policies

Parameter	Type	Description
auto_search	auto_search object	Hyperparameter search configuration

Table 8-134 auto_search

Parameter	Type	Description
skip_search_params	String	Hyperparameter parameters that need to be skipped
reward_attrs	Array of objects	Search metrics
search_params	Array of objects	Search parameters
algo_configs	Array of objects	Search algorithm configurations

Table 8-135 reward_attrs

Parameter	Type	Description
name	String	Metric name
mode	String	Search mode • **max**: A larger metric value is preferred. • **min**: A smaller metric value is preferred.

Parameter	Type	Description
regex	String	Regular expression of a metric

Table 8-136 search_params

Parameter	Type	Description
name	String	Hyperparameter name
param_type	String	Parameter type • **continuous**: Parameter values are continuous. • **discrete**: Parameter values are discrete.
lower_bound	String	Lower bound of the hyperparameter
upper_bound	String	Upper bound of the hyperparameter
discrete_points_num	String	Number of discrete points of a hyperparameter with continuous values
discrete_values	Array of strings	Discrete hyperparameter values

Table 8-137 algo_configs

Parameter	Type	Description
name	String	Name of the search algorithm
params	Array of **AutoSearchAlgoConfig-Parameter** objects	Search algorithm parameters

Table 8-138 AutoSearchAlgoConfigParameter

Parameter	Type	Description
key	String	Parameter key
value	String	Parameter value
type	String	Parameter type

Table 8-139 Input

Parameter	Type	Description
name	String	Name of the data input channel
description	String	Description of the data input channel
local_dir	String	Local directory of the container to which the data input channel is mapped
remote	**InputDataInfo** object	Information of the data input
remote_constraint	Array of objects	Data input constraint

Table 8-140 InputDataInfo

Parameter	Type	Description
dataset	**dataset** object	Dataset as the data input
obs	**obs** object	OBS in which data input and output are stored

Table 8-141 dataset

Parameter	Type	Description
id	String	Dataset ID of a training job
version_id	String	Dataset version ID of a training job
obs_url	String	OBS URL of the dataset for a training job, which is automatically parsed by ModelArts based on the dataset ID and dataset version IDs, for example, **/usr/data/**

Table 8-142 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-143 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location and dataset
attributes	String	Attributes when a dataset functions as the data input Options: • **data_format**: data format • **data_segmentation**: data segmentation • **dataset_type**: data labeling

Table 8-144 Output

Parameter	Type	Description
name	String	Name of the data output channel
description	String	Description of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output

Table 8-145 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-146 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-147 engine

Parameter	Type	Description
engine_id	String	Engine ID selected for a training job, which can be **engine_id**, **engine_name** and **engine_version**, or **image_url**
engine_name	String	Name of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
engine_version	String	Version of the engine selected for a training job. Leave this parameter blank if **engine_id** is specified.
image_url	String	Custom image URL selected for a training job

Table 8-148 TaskResponse

Parameter	Type	Description
role	String	Role of a heterogeneous training job task Options: • **learner**: GPUs or CPUs are supported. • **worker**: CPUs are supported.
algorithm	**algorithm** object	Algorithm configurations in algorithm management
task_resource	**FlavorResponse** object	Flavors for a training job or an algorithm

Table 8-149 algorithm

Parameter	Type	Description
code_dir	String	Absolute path of the directory where the algorithm boot file is stored
boot_file	String	Absolute path of the algorithm boot file
inputs	**inputs** object	Algorithm input channel
outputs	**outputs** object	Algorithm output channel
engine	**engine** object	Engine on which a heterogeneous job depends

Table 8-150 inputs

Parameter	Type	Description
name	String	Name of the data input channel
local_dir	String	Local path of the container to which the data input and output channels are mapped
remote	**remote** object	Actual data input, which can only be OBS for heterogeneous jobs

Table 8-151 remote

Parameter	Type	Description
obs	**obs** object	OBS in which data input and output are stored

Table 8-152 obs

Parameter	Type	Description
obs_url	String	OBS URL of the dataset for a training job, for example, **/usr/data/**

Table 8-153 outputs

Parameter	Type	Description
name	String	Name of the data output channel
local_dir	String	Local directory of the container to which the data output channel is mapped
remote	**remote** object	Information of the data output
mode	String	Data transmission mode, which defaults to **upload_periodically**
period	String	Data transmission period, which defaults to **30s**

Table 8-154 remote

Parameter	Type	Description
obs	**obs** object	OBS to which data is exported

Table 8-155 obs

Parameter	Type	Description
obs_url	String	OBS URL to which data is exported

Table 8-156 engine

Parameter	Type	Description
engine_id	String	Engine ID of a heterogeneous job, for example, **caffe-1.0.0-python2.7**
engine_name	String	Engine name of a heterogeneous job, for example, **Caffe**
engine_version	String	Engine version of a heterogeneous job
v1_compatible	Boolean	Whether v1 is compatible
run_user	String	User UID for which the engine is started by default

Table 8-157 FlavorResponse

Parameter	Type	Description
flavor_id	String	ID of the resource flavor
flavor_name	String	Name of the resource flavor
max_num	Integer	Maximum number of nodes with the resource flavor
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**

Parameter	Type	Description
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details
attributes	Map<String,String>	Other flavor attributes

Table 8-158 billing

Parameter	Type	Description
code	String	Billing code
unit_num	Integer	Number of billing units

Table 8-159 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.
cpu	**cpu** object	CPU specifications
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information

Table 8-160 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-161 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-162 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name
memory	String	Memory

Table 8-163 memory

Parameter	Type	Description
size	Integer	Memory size
unit	String	Number of memory units

Table 8-164 spec

Parameter	Type	Description
resource	**Resource** object	Resource flavors of a training job, which can either be **flavor_id** or **pool_id** and **flavor_id**
volumes	Array of objects	Volumes attached for a training job
log_export_path	**log_export_path** object	Export path of training job logs

Table 8-165 Resource

Parameter	Type	Description
policy	String	Resource flavor mode of a training job. Options: **regular**, **economic**, and **turbo**

Parameter	Type	Description
flavor_id	String	Resource flavor ID of a training job
flavor_name	String	Read-only flavor name returned by ModelArts when **flavor_id** is specified
node_count	Integer	Number of resource replicas selected for a training job Minimum value: **1**
pool_id	String	Resource pool ID selected for a training job
flavor_detail	**flavor_detail** object	Flavors for a training job or an algorithm

Table 8-166 flavor_detail

Parameter	Type	Description
flavor_type	String	Resource flavor type. Options: • **CPU** • **GPU** • **Ascend**
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details

Table 8-167 billing

Parameter	Type	Description
code	String	Billing code
unit_num	Integer	Number of billing units

Table 8-168 flavor_info

Parameter	Type	Description
max_num	Integer	Maximum number of nodes that can be selected. Value **1** indicates that the distributed mode is not supported.
cpu	**cpu** object	CPU specifications

Parameter	Type	Description
gpu	**gpu** object	GPU specifications
npu	**npu** object	Ascend specifications
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 8-169 cpu

Parameter	Type	Description
arch	String	CPU architecture
core_num	Integer	Number of cores

Table 8-170 gpu

Parameter	Type	Description
unit_num	Integer	Number of GPUs
product_name	String	Product name
memory	String	Memory

Table 8-171 npu

Parameter	Type	Description
unit_num	String	Number of NPUs
product_name	String	Product name
memory	String	Memory

Table 8-172 memory

Parameter	Type	Description
size	Integer	Memory size

Parameter	Type	Description
unit	String	Number of memory units

Table 8-173 disk

Parameter	Type	Description
size	String	Disk size
unit	String	Unit of the disk size, which is GB generally

Table 8-174 volumes

Parameter	Type	Description
nfs	**nfs** object	Disks attached in NFS mode

Table 8-175 nfs

Parameter	Type	Description
nfs_server_path	String	NFS server path
local_path	String	Path for attaching disks to the training container
read_only	Boolean	Whether the disks attached to the container in NFS mode are read-only

Table 8-176 log_export_path

Parameter	Type	Description
obs_url	String	OBS URL for storing training job logs
host_path	String	Path of the host where training job logs are stored

Table 8-177 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.

Parameter	Type	Description
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.8 Obtaining Training Logs

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
estimator = Estimator(session=session, job_id="your job id")
info = estimator.get_job_log()
print(info)
```

- Method 2: Use the training job created in **Creating a Training Job**.

```
log = job_instance.get_job_log(task_id="worker-0")
print(log)
```

Parameters

Table 8-178 Parameters for initializing the Estimator

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

Table 8-179 get_job_log request parameters

Parameter	Mandatory	Type	Description
task_id	No	String	ID of a worker node for obtaining logs. It defaults to **worker-0**. If **train_instance_count** is set to **2** when you create a training job, the value of this parameter can be **worker-0** or **worker-1**.

Table 8-180 Response parameters

Parameter	Type	Description
content	String	Log content • If the size of the log file does not exceed the limit allowed (n MB), all logs are returned. • If the size of the log file exceeds the limit allowed (n MB), the latest n MB logs are returned.
current_size	Integer	Size of the returned log file, in bytes. The maximum value is 5 MB.
full_size	Integer	Size of a complete log file, in bytes.

Table 8-181 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.1.9 Obtaining the Runtime Metrics of a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
estimator = Estimator(session=session, job_id="your job id")
info = estimator.get_job_metrics()
print(info)
```

- Method 2: Use the training job created in **Creating a Training Job**.

```
info = job_instance.get_job_metrics(task_id="worker-0")
print(info)
```

Parameters

Table 8-182 Parameters for initializing the Estimator

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can obtain **job_id** using the training job created in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Obtaining Training Jobs**.

Table 8-183 get_job_log request parameters

Parameter	Mandatory	Type	Description
task_id	No	String	ID of a worker node for obtaining logs. It defaults to **worker-0**. If **train_instance_count** is set to **2** when you create a training job, the value of this parameter can be **worker-0** or **worker-1**.

Table 8-184 Response parameters

Parameter	Type	Description
metrics	Array of objects	Runtime metrics

Table 8-185 metrics

Parameter	Type	Description
metric	String	Runtime metric. The value can be cpuUsage (CPU usage), memUsage (physical memory usage), gpuUtil (GPU usage), gpuMemUsage (GPU memory usage), npuUtil (NPU usage), or npuMemUsage (NPU memory usage).
value	Array of numbers	Value of a runtime metric. An average value is collected every minute.

Table 8-186 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.2 APIs for Resources and Engine Specifications

8.2.1 Obtaining Resource Flavors

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
info = Estimator.get_train_instance_types(session=session)
print(info)
```


Parameters

Table 8-187 get_train_instance_types parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 8-188 Successful response parameters

Type	Description
List	List of resource flavors

Table 8-189 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

8.2.2 Obtaining Engine Types

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimatorV2 import Estimator
session = Session()
info = Estimator.get_framework_list(session=session)
print(info)
```


Parameters

Table 8-190 get_train_instance_types parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 8-191 Successful response parameters of **get_framework_list**

Type	Description
List	List of engine types. For details, see **Table 3**.

Table 8-192 framework_list parameters

Parameter	Type	Description
framework_type	String	Engine type
framework_version	String	Engine version

Table 8-193 Response for the failure to call a training API

Parameter	Type	Description
error_msg	String	Error message when calling an API failed. This parameter is unavailable if an API is successfully called.
error_code	String	Error code when calling an API failed. For details, see **Error Codes**. This parameter is unavailable if an API is successfully called.
error_solution	String	Solution to an API calling failure. This parameter is unavailable if an API is successfully called.

9 Training Management (Old Version)

9.1 Training Jobs

9.1.1 Creating a Training Job

For training on the training platform, if the training fails, you can view the detailed log information on the platform or by calling the API in **Querying Training Job Logs**.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Example 1: Create a training job using the data stored on OBS.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/bucket/output/',                              # Training output directory
                    train_instance_type='modelarts.vm.cpu.2u',                  # Training environment flavor
                    train_instance_count=1,                                     # Number of training nodes
                    job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_instance = estimator.fit(inputs='/bucket/data/train/', wait=False, job_name='my_training_job')
```

- Example 2: Create a training job using a dataset.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
```



```
                modelarts_session=session,
                framework_type='PyTorch',                                   # AI engine name
                framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                code_dir='/bucket/src/',                                    # Training script directory
                boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                log_url='/bucket/log/',                                     # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                              # Training output directory
                train_instance_type='modelarts.vm.cpu.2u',                  # Training environment flavor
                train_instance_count=1,                                     # Number of training nodes
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_instance = estimator.fit(dataset_id='4AZNvFkN7KYr5EdhFkH',
dataset_version_id='UOF9BIeSGArwVt0oI6T', wait=False, job_name='my_training_job')
```

- Example 3: Create a training job using a custom image.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                modelarts_session=session,
                log_url='/bucket/log/',                                     # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                              # Training output directory
                train_instance_type='modelarts.vm.cpu.2u',                  # Training environment flavor
                train_instance_count=1,                                     # Number of training nodes
                user_command='bash -x /home/work/run_train.sh python /home/work/user-job-
dir/app/mnist/mnist_softmax.py --data_url /home/work/user-job-dir/app/
mnist_data',                                                # Boot command of the custom image
                user_image_url='100.125.5.235:20202/jobmng/cpu-base:1.0',   # Address for
downloading the custom image
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_instance = estimator.fit(inputs='/bucket/data/train/', wait=False, job_name='my_training_job')
```

- Example 4: Submit a training job in a dedicated resource pool.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                modelarts_session=session,
                framework_type='PyTorch',                                   # AI engine name
                framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                code_dir='/bucket/src/',                                    # Training script directory
                boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                log_url='/bucket/log/',                                     # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                              # Training output directory
            pool_id="your pool id",                                 # Dedicated resource pool ID
                train_instance_type='your instance type',                   # Training environment flavor. If
the value is None, the default flavor of the dedicated resource pool will be used.
                train_instance_count=1,                                     # Number of training nodes
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_instance = estimator.fit(inputs='/bucket/data/train/', wait=False, job_name='my_training_job')
```


Parameters

Table 9-1 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
train_instance_count	Yes	Int	Number of compute nodes in a training job
code_dir	No	String	Code directory of a training job, for example, **/bucket/src/**. Leave this parameter blank when **model_name** is set.
boot_file	No	String	Boot file of a training job, which must be stored in the code directory. For example, **/bucket/src/boot.py**. Leave this parameter blank when **model_name** is set.
model_name	No	String	Name of the built-in algorithm used by a training job. If you have configured **model_name**, you do not need to configure **app_url**, **boot_file_url**, **framework_type**, and **framework_version**. You can obtain this value by calling the API described in **Querying a Built-in Algorithm**.
output_path	Yes	String	Output path of a training job
hyperparameters	No	JSON Array	Running parameters of a training job. It is a collection of label-value pairs of the string type. This parameter is a container environment variable when a job uses a custom image.
log_url	No	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/log/**
train_instance_type	Yes	String	Resource flavor selected for a training job. If you choose to train on the training platform, obtain the value by calling the API described in **Querying the List of Resource Flavors**.

Parameter	Mandatory	Type	Description
framework_type	No	String	Engine selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
framework_version	No	String	Engine version selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
job_description	No	String	Description of a training job
user_image_url	No	String	SWR URL of the custom image used by a training job. Example value: **100.125.5.235:20202/jobmng/custom-cpu-base:1.0**
user_command	No	String	Boot command used to start the container of the custom image of a training job. The format is **bash /home/work/run_train.sh python /home/work/user-job-dir/app/train.py {python_file_parameter}**.
pool_id	No	String	ID of the resource pool for a training job. To obtain the ID, do as follows: Log in to the ModelArts management console, choose **Dedicated Resource Pools** in the navigation pane on the left, and view the resource pool ID in the dedicated resource pool list.

Table 9-2 fit request parameters

Parameter	Mandatory	Type	Description
inputs	Yes	String	Data storage location of a training job. **inputs** cannot be used with **dataset_id** and **dataset_version_id**, or with **data_source** at the same time. However, one of the parameters must exist. Only this parameter is supported in local training.

Parameter	Mandatory	Type	Description
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.
wait	No	Boolean	Whether to wait for the completion of a training job. Default value: **False**
job_name	No	String	Name of a training job. Enter 1 to 64 characters. Only the following characters are allowed: a-z, A-Z, 0-9, hyphens (-), and underscores (_). If this parameter is left blank, a job name is generated randomly.

Table 9-3 Parameters in the successful response to training

Parameter	Type	Description
TrainingJob	Object	Training object. This object contains attributes such as **job_id** and **version_id**, and operations on a training job, such as querying, modifying, or deleting the training job. For example, you can use **job_instance.job_id** to obtain the ID of a training job.

9.1.2 Debugging a Training Job

Before creating a real-time training job, create a local training job for debugging.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Step 1: Create a local training job. If **train_instance_type** is set to **local**, a local training job is created, which can be used to debug code and parameters.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
from modelarts.environment import Environment
from modelarts.environment.conda_env import CondaDependencies
```

```
session = Session()
env = Environment("tensorflow_mlp_mnist")
cd = CondaDependencies.create(pip_packages=["tensorflow==1.13.1", "requests"],
conda_packages=["python=3.6.2"])
env.conda = cd
src_local_path = "/home/ma-user/work/tensorflow_mlp_mnist_local_mode/train/"
train_file = "tensorflow_mlp_mnist.py"
estimator = Estimator(modelarts_session=session,
                      code_dir=src_local_path,           # Path of the local training script
                      boot_file=train_file,              # Path of the local training boot script
                      train_instance_type='local',       # Local training
                      train_instance_count=1,            # Number of training nodes
                      environment=env)                   # Environment for running the training script
job_instance = estimator.fit(wait=False, job_name='my_training_job')
```

- Step 2: After the local training job is complete, create a real-time training job. If **train_instance_type** is set to a training environment flavor, a real-time training job is created.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
from modelarts.environment import Environment
from modelarts.environment.conda_env import CondaDependencies

session = Session()
env = Environment("tensorflow_mlp_mnist")
cd = CondaDependencies.create(pip_packages=["tensorflow==1.13.1", "requests"],
conda_packages=["python=3.6.2"])
env.conda = cd
src_local_path = "/home/ma-user/work/tensorflow_mlp_mnist_local_mode/train/"
train_file = "tensorflow_mlp_mnist.py"
estimator = Estimator(modelarts_session=session,
                      code_dir=src_local_path,                    # Path of the training script
                       boot_file=train_file,                      # Path of the training boot script
                      train_instance_type='modelarts.vm.cpu.2u',  # Real-time training
                      train_instance_count=1,                     # Number of training nodes
                      environment=env)                            # Environment for running the training script
job_instance = estimator.fit(wait=False, job_name='my_training_job')
```

Parameters

Table 9-4 Environment parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Environment name
conda	No	CondaDependencies	Conda environment. For details, see **Table 9-5**.

Table 9-5 CondaDependencies parameters

Parameter	Mandatory	Type	Description
channels	No	List	Source for downloading the Python package

Parameter	Mandatory	Type	Description
pip_packages	No	List	Python package required by the Conda virtual environment, such as TensorFlow and Pillow
conda_packages	No	List	Conda package required by the Conda virtual environment, for example, a specified Python version

Table 9-6 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
train_instance_count	Yes	Int	Number of compute nodes in a training job.
code_dir	No	String	Code directory of a training job, for example, **/bucket/src/**. Leave this parameter blank if **model_name** is set.
boot_file	No	String	Boot file of a training job, which needs to be stored in the code directory, for example, **/bucket/src/boot.py**. Leave this parameter blank if **model_name** is set.
model_name	No	String	Name of the built-in algorithm used by a training job. If you have configured **model_name**, you do not need to configure **app_url**, **boot_file_url**, **framework_type**, and **framework_version**. You can obtain this value by calling the API described in **Querying a Built-in Algorithm**.
output_path	Yes	String	Output path of a training job.
hyperparameters	No	JSON array	Running parameters of label-value pairs of the string type for a training job. This parameter is a container environment variable if a job uses a custom image.
log_url	No	String	OBS URL of training job logs. By default, this parameter is left blank. An example value is **/usr/log/**.

Parameter	Mandatory	Type	Description
train_instance_type	Yes	String	Resource flavor selected for a training job. If you choose to train on the training platform, obtain the value by calling the API described in **Querying the List of Resource Flavors**.
framework_type	No	String	Engine selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank if **model_name** is set.
framework_version	No	String	Engine version selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank if **model_name** is set.
job_description	No	String	Description of a training job.
user_image_url	No	String	SWR URL of the custom image used by a training job. An example value is **100.125.5.235:20202/jobmng/custom-cpu-base:1.0**.
user_command	No	String	Boot command used to start the container of the custom image used by a training job. The format is **bash /home/work/run_train.sh python /home/work/user-job-dir/app/train.py {python_file_parameter}**.
pool_id	No	String	ID of the resource pool for a training job. To obtain the ID, do as follows: Log in to the ModelArts console, choose **Dedicated Resource Pools** in the navigation pane, and view the resource pool ID in the dedicated resource pool list.

Table 9-7 fit request parameters

Parameter	Mandatory	Type	Description
inputs	Yes	String	Data storage location of a training job. **inputs** cannot be used with **dataset_id** and **dataset_version_id**, or with **data_source** at the same time. However, one of these parameters must be set. Only this parameter is supported in local training.
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.
wait	No	Boolean	Whether to wait for the completion of a training job. It defaults to **False**.
job_name	No	String	Name of a training job. Enter 1 to 64 characters. Only the following characters are allowed: a-z, A-Z, 0-9, hyphens (-), and underscores (_). If this parameter is left blank, a job name is generated randomly.

9.1.3 Querying the List of Training Jobs

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
job_list_info = Estimator.get_job_list(modelarts_session=session, status=8, per_page=10, page=1,
order="asc", search_content="job")
print(job_list_info)
```

Parameters

Table 9-8 get_job_list request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
status	No	Integer	Job status to be queried. By default, jobs of all statuses are queried. For example, to view jobs that fail to be created, set this parameter to **3**, **5**, **6**, or **13**. For details about the job statuses, see **Job Statuses**.
per_page	No	Integer	Number of jobs displayed on each page. The value range is [1, 1000]. Default value: **10**
page	No	Integer	Index of the page to be queried. Default value: **1**
sortBy/ sort_by	No	String	When AK/SK-based authentication is used, the parameter name is **sortBy**. When account-based authentication is used, the parameter name is **sort_by**. The parameter specifies the sorting mode of the query. The value can be **job_name**, **job_desc**, **status**, **duration**, **engine_type**, or **create_time**. Default value: **job_name**
order	No	String	The options are as follows: • **asc**: ascending order • **desc**: descending order. The default value is **desc**.
search_content	No	String	Search content, for example, a training job name. The value consists of 0 to 100 characters. By default, this parameter is left blank.

Table 9-9 get_job_list response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
job_total_count	Integer	Total number of created jobs
job_count_limit	Integer	Number of training jobs that can be created
is_success	Boolean	Whether the API call succeeds
quotas	Integer	Maximum number of training jobs
jobs	JSON Array	Attributes of a training job. For details, see **Table 9-10**.

Table 9-10 jobs parameters

Parameter	Type	Description
job_id	Long	Training job ID
job_name	String	Training job name
version_id	Long	Version ID of a training job
status	Byte	Status of a training job. For details about the job statuses, see **Job Statuses**.
create_time	Long	Timestamp when a training job is created
duration	Long	Training job running duration, in milliseconds
job_desc	String	Description of a training job
version_count	Long	Number of versions of a training job

9.1.4 Querying the Details About a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id** and **version_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
```



```
session = Session()
estimator = Estimator(modelarts_session=session, job_id="182626", version_id="278813")
job_info = estimator.get_job_info()
print(job_info)
```

- Method 2: Use the training job created in **Creating a Training Job**.
  ```
  job_info = job_instance.get_job_info()
  print(job_info)
  ```
- Method 3: Use the training job version object returned in **Querying the List of Training Job Versions**.
  ```
  job_info = job_version_instance_list[0].get_job_info()
  print(job_info)
  ```

Parameters

Table 9-11 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
version_id	Yes	String	Version ID of a training job. You can query **version_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.version_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-12 get_job_info response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds
job_id	Long	Training job ID

Parameter	Type	Description
job_name	String	Training job name
job_desc	String	Description of a training job
version_id	Long	Version ID of a training job
version_name	String	Version name of a training job
pre_version_id	Long	Name of the previous version of a training job
engine_type	Short	Engine type of a training job. The mapping between **engine_type** and **engine_name** is as follows: • **engine_type**: **1**, **engine_name**: **TensorFlow** • **engine_type**: **2**, **engine_name**: **MXNet** • **engine_type**: **3**, **engine_name**: **Ray** • **engine_type**: **4**, **engine_name**: **Caffe** • **engine_type**: **5**, **engine_name**: **Spark_MLlib** • **engine_type**: **9**, **engine_name**: **XGBoost-Sklearn** • **engine_type**: **10**, **engine_name**: **PyTorch** • **engine_type**: **12**, **engine_name**: **Horovod**
engine_name	String	Name of the engine selected for a training job. Currently, the following engines are supported: • Caffe • Horovod • MXNet • PyTorch • Ray • Spark_MLlib • TensorFlow • XGBoost-Sklearn
engine_id	Long	ID of the engine selected for a training job
engine_version	String	Version of the engine selected for a training job
status	Integer	Status of a training job. For details about the job statuses, see **Job Statuses**.
app_url	String	Code directory of a training job
boot_file_url	String	Boot file of a training job
create_time	Long	Time when a training job is created

Parameter	Type	Description
parameter	JSON Array	Running parameters of a training job. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image.
duration	Long	Training job running duration, in milliseconds
spec_id	Long	ID of the resource specifications selected for a training job
core	String	Number of cores of the resource specifications
cpu	String	CPU memory of the resource specifications
gpu_num	Integer	Number of GPUs of the resource specifications
gpu_type	String	GPU type of the resource specifications
worker_server_num	Integer	Number of workers in a training job
data_url	String	Dataset of a training job
train_url	String	OBS path to the training job output file
dataset_version_id	String	Dataset version ID of a training job
dataset_id	String	Dataset ID of a training job
data_source	JSON Array	Datasets of a training job
model_id	Long	Model ID of a training job
model_metric_list	JSON Array	Model metrics of a training job
system_metric_list	JSON Array	System monitoring metrics of a training job
user_image_url	String	SWR URL of the custom image used by a training job
user_command	String	Boot command used to start the container of the custom image of a training job

Table 9-13 data_source parameters

Parameter	Type	Description
dataset_id	String	Dataset ID of a training job

Parameter	Type	Description
dataset_version	String	Dataset version ID of a training job
type	String	Dataset type **obs**: Data from OBS is used. **dataset**: Data from a specified dataset is used.
data_url	String	OBS bucket path

Table 9-14 model_metric_list parameters

Parameter	Type	Description
metric	JSON Array	Validation metrics of a class of a training job
total_metric	JSON Array	All validation metrics of a training job

Table 9-15 system_metric_list parameters

Parameter	Type	Description
cpuUsage	JSON Array	CPU usage of a training job
memUsage	JSON Array	Memory usage of a training job
gpuUtil	JSON Array	GPU usage of a training job

Table 9-16 metric parameters

Parameter	Type	Description
metric_values	JSON Array	Validation metrics of a class of a training job
reserved_data	JSON Array	Reserved parameter
metric_meta	JSON Array	A class of a training job, including the class ID and name

Table 9-17 metric_values parameters

Parameter	Type	Description
recall	JSON Array	Recall of a class of a training job
precision	JSON Array	Precision of a class of a training job
accuracy	JSON Array	Accuracy of a class of a training job

Table 9-18 total_metric parameters

Parameter	Type	Description
total_metric_meta	JSON Array	Reserved parameter
total_reserved_data	JSON Array	Reserved parameter
total_metric_values	JSON Array	All validation metrics of a training job

Table 9-19 total_metric_values parameters

Parameter	Type	Description
f1_score	Float	F1 score of a training job
recall	Float	Total recall of a training job
precision	Float	Total precision of a training job
accuracy	Float	Total accuracy of a training job

9.1.5 Modifying the Description of a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Modify the description of a training job based on the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
```



```
estimator = Estimator(modelarts_session=session, job_id="182626")
job_description = estimator.update_job_description(description='update description')
```

- Method 2: Modify the description of the training job created in **Creating a Training Job**.

```
job_description = job_instance.update_job_description(description='update description')
```

- Method 3: Modify the description of a training job version object returned in **Querying the List of Training Job Versions**.

```
job_description = job_version_instance_list[0].update_job_description(description='update description')
```

Parameters

Table 9-20 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-21 update_job_description request parameters

Parameter	Mandatory	Type	Description
description	Yes	String	Description of the training job to be modified

Table 9-22 update_job_description response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.1.6 Obtaining the Name of a Training Job Log File

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Obtain the file based on the specified **job_id** and **version_id**.
```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(modelarts_session=session, job_id="182626", version_id="278813")
job_log_list = estimator.get_job_log_file_list()
```
- Method 2: Obtain the file based on the training job created in **Creating a Training Job**.
```
job_log_list = job_instance.get_job_log_file_list()
```
- Method 3: Obtain the file based on the training job version object returned in **Querying the List of Training Job Versions**.
```
job_log_list = job_version_instance_list[0].get_job_log_file_list()
```

Parameters

Table 9-23 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
version_id	Yes	String	Version ID of a training job. You can query **version_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.version_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-24 get_job_log_file_list response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
log_file_list	List	Log file name of a training job. A single-node job has only one log file, and a distributed job has multiple log files.
is_success	Boolean	Whether the API call succeeds

9.1.7 Querying Training Job Logs

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id** and **version_id**.
```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(modelarts_session=session, job_id="182626", version_id="278813")
job_log = estimator.get_job_log(log_file='job-job-0713-191758.0')
print(job_log)
```
- Method 2: Use the training job created in **Creating a Training Job**.
```
job_log = job_instance.get_job_log(log_file='job-job-0713-191758.0')
print(job_log)
```
- Method 3: Use the training job version object returned in **Querying the List of Training Job Versions**.
```
job_log = job_version_instance_list[0].get_job_log(log_file='job-job-0713-191758.0')
print(job_log)
```

Parameters

Table 9-25 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Parameter	Mandatory	Type	Description
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
version_id	Yes	String	Version ID of a training job. You can query **version_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.version_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-26 get_job_log request parameters

Parameter	Mandatory	Type	Description
log_file	Yes	String	Name of a training job log file
start_byte	No	Long	Start position for obtaining the log. The default value is **0**. The value range is [-1, +∞]. If the value is **-1**, the log with the latest offset is obtained.
offset	No	Long	Length of the obtained log. The default value is **2048**. The value range is [-2048, 2048].

Table 9-27 get_job_log response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
content	String	Content of the requested log
lines	Integer	Number of lines in the log
start_line	String	Start position of the obtained log

Parameter	Type	Description
end_line	String	End position of the obtained log

9.1.8 Deleting a Training Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

Method 1: Delete a training job based on the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
Estimator.delete_job_by_id(modelarts_session=session, job_id="155500")
```

Method 2: Delete the training job created in **Creating a Training Job**.

```
status = job_instance.delete_job()
```

Parameters

Table 9-28 delete_job_by_id request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-29 delete_job_by_id response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.2 Training Job Versions

9.2.1 Creating a Training Job Version

A training job must exist before you create a version for it. You can create a training job version based on **Creating a Training Job** or **job_id** and **version_id** of the object returned by **Querying the List of Training Job Versions**.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Example 1: Create a training job version using the data stored on OBS.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/bucket/output/',                              # Training output directory
                    train_instance_type='modelarts.vm.gpu.p100',                # Training environment flavor
                    train_instance_count=1)
job_version_instance = estimator.create_job_version(job_id='182626', pre_version_id=278813, inputs='/
bucket/data/train/', wait=False, job_desc='create a job version')
```

- Example 2: Create a training job version using a dataset.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
```



```
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                          # Training output directory
                train_instance_type='modelarts.vm.gpu.p100',            # Training environment flavor
                train_instance_count=1,                                 # Number of training nodes
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_version_instance = estimator.create_job_version(job_id='182626', pre_version_id=278813, inputs='/
bucket/data/train/', wait=False, job_desc='create a job version')
```

- Example 3: Create a training job version using a custom image.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                modelarts_session=session,
                log_url='/bucket/log/',                                 # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                          # Training output directory
                train_instance_type='modelarts.vm.gpu.p100',            # Training environment flavor
                train_instance_count=1,                                 # Number of training nodes
                user_command='bash -x /home/work/run_train.sh python /home/work/user-job-
dir/app/mnist/mnist_softmax.py --data_url /home/work/user-job-dir/app/
mnist_data',                                                            # Boot command of the custom image
                user_image_url='100.125.5.235:20202/jobmng/cpu-base:1.0',     # Address for
downloading the custom image
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_version_instance = estimator.create_job_version(job_id='182626', pre_version_id=278813, inputs='/
bucket/data/train/', wait=False, job_desc='create a job version')
```

- Example 4: Create a training job version using a built-in algorithm.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                modelarts_session=session,
                model_name='Faster_RCNN_ResNet_v1_50',                  # Name of the built-in
algorithm
                log_url='/bucket/log/',                                 # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                          # Training output directory
                train_instance_type='modelarts.vm.gpu.p100',            # Training environment flavor
                train_instance_count=1,                                 # Number of training nodes
                job_description='pytorch-sentiment with ModelArts SDK')     # Training job description
job_version_instance = estimator.create_job_version(job_id='182626', pre_version_id=278813, inputs='/
bucket/data/train/', wait=False, job_desc='create a job version')
```


Parameters

Table 9-30 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
train_instance_count	Yes	Long	Number of workers in a training job
code_dir	No	String	Code directory of a training job, for example, **/bucket/src/**. Leave this parameter blank when **model_name** is set.
boot_file	No	String	Boot file of a training job, which must be stored in the code directory. For example, **/bucket/src/boot.py**. Leave this parameter blank when **model_name** is set.
model_name	No	Long	Name of the built-in algorithm used by a training job. If you have configured **model_name**, you do not need to configure **app_url**, **boot_file_url**, **framework_type**, and **framework_version**. You can obtain this value by calling the API described in **Querying a Built-in Algorithm**.
output_path	Yes	String	Output path of a training job
hyperparameters	No	JSON Array	Running parameters of a training job. It is a collection of label-value pairs of the string type. This parameter is a container environment variable when a job uses a custom image.
log_url	No	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/log/**
train_instance_type	Yes	Long	Resource flavor selected for a training job. If you choose to train on the training platform, obtain the value by calling the API described in **Querying the List of Resource Flavors**.

Parameter	Mandatory	Type	Description
framework_type	No	String	Engine selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
framework_version	No	String	Engine version selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
user_image_url	No	String	SWR URL of the custom image used by a training job. Example value: **100.125.5.235:20202/jobmng/custom-cpu-base:1.0**
user_command	No	String	Boot command used to start the container of the custom image of a training job. The format is **bash /home/work/run_train.sh python /home/work/user-job-dir/app/train.py {python_file_parameter}**.

Table 9-31 create_job_version request parameters

Parameter	Mandatory	Type	Description
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
pre_version_id	Yes	Long	ID of the previous version of a training job. You can query **pre_version_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.version_id**, or from the response obtained in **Querying the List of Training Jobs**.

Parameter	Mandatory	Type	Description
inputs	Yes	String	Data storage location of a training job. **inputs** cannot be used with **dataset_id** and **dataset_version_id**, or with **data_source** at the same time. However, one of the parameters must exist. Only this parameter is supported in local training.
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.
wait	No	Boolean	Whether to wait for the completion of creating a training job version. Default value: **False**
job_desc	No	String	Description of a training job

Table 9-32 create_job_version response parameters

Parameter	Type	Description
TrainingJob	Object	Training object. This object contains attributes such as **job_id** and **version_id**, and operations on a training job, such as querying, modifying, or deleting the training job. For example, you can use **job_version_instance.job_id** to obtain the ID of a training job.

9.2.2 Querying the List of Training Job Versions

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(session, job_id="182626")
```



```
job_version_instance_list = estimator.get_job_version_object_list()
print(job_version_instance_list)
```

Parameters

Table 9-33 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-34 get_job_version_object_list request parameters

Parameter	Mandatory	Type	Description
is_show	No	Boolean	Whether to print the training job version details. Default value: **True**

A training object list is returned in the successful response to **get_job_version_object_list**. For details, see **Table 9-35**.

Table 9-35 TrainingJob object description

Parameter	Type	Description
TrainingJob	Object	Training object. This object contains attributes such as **job_id** and **version_id**, and operations on a training job, such as querying, modifying, or deleting the training job. For example, you can use **job_version_instance.job_id** to obtain the ID of a training job.

9.2.3 Querying the Details About a Training Job Version

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **job_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(session, job_id="182626")
job_version_info = estimator.get_job_version_info()
print(job_version_info)
```

- Method 2: Use the training job version created in **Creating a Training Job Version**.

```
job_version_info = job_version_instance.get_job_version_info()
print(job_version_info)
```

Parameters

Table 9-36 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.

Table 9-37 get_job_version_info response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

Parameter	Type	Description
job_id	Long	Training job ID
job_name	String	Training job name
job_desc	String	Description of a training job
version_count	Long	Number of versions of a training job
versions	JSON Array	Version parameters of a training job

Table 9-38 versions parameters

Parameter	Type	Description
version_id	Long	Version ID of a training job
version_name	String	Version name of a training job
pre_version_id	Long	ID of the previous version of a training job
engine_type	Long	Engine type of a training job
engine_id	Long	ID of the engine selected for a training job
engine_version	String	Version of the engine selected for a training job
status	Integer	Status of a training job
app_url	String	Code directory of a training job
boot_file_url	String	Boot file of a training job
create_time	Long	Time when a training job is created
parameter	JSON Array	Running parameters of a training job. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image.
duration	Long	Training job running duration, in milliseconds
spec_id	Long	ID of the resource specifications selected for a training job
core	String	Number of cores of the resource specifications
cpu	String	CPU memory of the resource specifications
gpu_num	Integer	Number of GPUs of the resource specifications

Parameter	Type	Description
gpu_type	String	GPU type of the resource specifications
worker_server_num	Integer	Number of workers in a training job
data_url	String	Dataset of a training job
train_url	String	OBS path to the training job output file
log_url	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/log/**
dataset_version_id	String	Dataset version ID of a training job
dataset_id	String	Dataset ID of a training job
data_source	JSON Array	Datasets of a training job
model_id	String	Model ID of a training job
model_metric_list	JSON Array	Model metrics of a training job
system_metric_list	JSON Array	System monitoring metrics of a training job
user_image_url	String	SWR URL of the custom image used by a training job
user_command	String	Boot command used to start the container of the custom image of a training job

Table 9-39 data_source parameters

Parameter	Type	Description
dataset_id	String	Dataset ID of a training job
dataset_version	String	Dataset version ID of a training job
type	String	Dataset type **obs**: Data from OBS is used. **dataset**: Data from a specified dataset is used.
data_url	String	OBS bucket path

Table 9-40 model_metric_list parameters

Parameter	Type	Description
metric	JSON Array	Validation metrics of a class of a training job
total_metric	JSON Array	All validation metrics of a training job

Table 9-41 system_metric_list parameters

Parameter	Type	Description
cpuUsage	JSON Array	CPU usage of a training job
memUsage	JSON Array	Memory usage of a training job
gpuUtil	JSON Array	GPU usage of a training job

Table 9-42 metric parameters

Parameter	Type	Description
metric_values	JSON Array	Validation metrics of a class of a training job
reserved_data	JSON Array	Reserved parameter
metric_meta	JSON Array	A class of a training job, including the class ID and name

Table 9-43 metric_values parameters

Parameter	Type	Description
recall	JSON Array	Recall of a class of a training job
precision	JSON Array	Precision of a class of a training job
accuracy	JSON Array	Accuracy of a class of a training job

Table 9-44 total_metric parameters

Parameter	Type	Description
total_metric_meta	JSON Array	Reserved parameter
total_reserved_data	JSON Array	Reserved parameter
total_metric_values	JSON Array	All validation metrics of a training job

Table 9-45 total_metric_values parameters

Parameter	Type	Description
f1_score	Float	F1 score of a training job
recall	Float	Total recall of a training job
precision	Float	Total precision of a training job
accuracy	Float	Total accuracy of a training job

9.2.4 Stopping a Training Job Version

You can stop a training job version that is being created only when the job is running.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Stop a training job version based on the specified **job_id** and **version_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(session, job_id="182626", version_id="278813")
status = estimator.stop_job_version()
```

- Method 2: Stop the training job version created in **Creating a Training Job Version**.

```
status = job_version_instance.stop_job_version()
```

- Method 3: Stop the training job version object returned in **Querying the List of Training Job Versions**.

```
status = job_version_instance_list[0].stop_job_version()
```

Parameters

Table 9-46 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
version_id	Yes	String	Version ID of a training job. You can query **version_id** from the response obtained in **Querying the List of Training Job Versions**.

Table 9-47 stop_job_version response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.2.5 Deleting a Training Job Version

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Delete a training job version based on the specified **job_id** and **version_id**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
```



```
estimator = Estimator(session, job_id="182626", version_id="278813")
status = estimator.delete_job_version()
```

- Method 2: Delete the training job version created in **Creating a Training Job Version**.

```
status = job_version_instance.delete_job_version()
```

- Method 3: Delete the training job version object returned in **Querying the List of Training Job Versions**.

```
status = job_version_instance_list[0].delete_job_version()
```

Parameters

Table 9-48 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
job_id	Yes	String	ID of a training job. You can query **job_id** using the training job object generated in **Creating a Training Job**, for example, **job_instance.job_id**, or from the response obtained in **Querying the List of Training Jobs**.
version_id	Yes	String	Version ID of a training job. You can query **version_id** from the response obtained in **Querying the List of Training Job Versions**.

Table 9-49 delete_job_version response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.3 Training Job Parameter Configuration

9.3.1 Creating a Training Job Configuration

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Example 1: Create a training job parameter configuration using the data stored on OBS.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/bucket/output/',                              # Training output directory
                    train_instance_type='modelarts.vm.gpu.p100',                # Training environment
flavor
                    train_instance_count=1)                                     # Number of training nodes
job_config_instance = estimator.create_job_configs(config_name='my_job_config', inputs='/bucket/
data/train/', config_desc='my job config')
```

- Example 2: Create a training job parameter configuration using a dataset.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/bucket/output/',                              # Training output directory
                    train_instance_type='modelarts.vm.gpu.p100',                # Training environment
flavor
                    train_instance_count=1)                                     # Number of training nodes
job_config_instance = estimator.create_job_configs(config_name='my_job_config',
dataset_id='4AZNvFkN7KYr5EdhFkH', dataset_version_id='UOF9BIeSGArwVt0oI6T', config_desc='my
job config')
```

Parameters

Table 9-50 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
train_instance_count	Yes	Long	Number of workers in a training job
code_dir	No	String	Code directory of a training job, for example, **/bucket/src/**. Leave this parameter blank when **model_name** is set.
boot_file	No	String	Boot file of a training job, which needs to be stored in the code directory. For example, **/bucket/src/boot.py**. Leave this parameter blank when **model_name** is set.
model_name	No	Long	Name of the built-in algorithm used by a training job. If you have configured **model_name**, you do not need to configure **app_url**, **boot_file_url**, **framework_type**, and **framework_version**.
output_path	Yes	String	Output path of a training job
hyperparameters	No	JSON Array	Running parameters of a training job. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image.
log_url	No	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/log/**
train_instance_type	Yes	Long	Resource flavor selected for a training job. If you choose to train on the training platform, obtain the value by calling the API described in **Querying the List of Resource Flavors**.
framework_type	No	String	Engine selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.

Parameter	Mandatory	Type	Description
framework_version	No	String	Engine version selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
job_description	No	String	Description of a training job
user_image_url	No	String	SWR URL of the custom image used by a training job. Example value: **100.125.5.235:20202/jobmng/custom-cpu-base:1.0**
user_command	No	String	Boot command used to start the container of the custom image of a training job. The format is **bash /home/work/run_train.sh python /home/work/user-job-dir/app/train.py {python_file_parameter}**.

Table 9-51 create_job_configs request parameters

Parameter	Mandatory	Type	Description
config_name	No	String	Name of a training job parameter configuration. The value is a string of 1 to 20 characters consisting of only digits, letters, underscores (_), and hyphens (-). By default, if this parameter is left blank, the value is dynamically generated by date.
config_desc	No	String	Description of a training job parameter configuration. The value is a string of 0 to 256 characters. By default, this parameter is left blank.
inputs	No	String	OBS storage path of a training job
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.

Table 9-52 create_job_configs response parameters

Parameter	Type	Description
TrainingJob	Object	Training object. This object contains attributes such as **config_name**, and operations on a training job parameter configuration, such as querying or deleting the training job parameter configuration. For example, you can use **job_config_instance.config_name** to obtain the name of a training job parameter configuration.

9.3.2 Querying the List of Training Job Parameter Configuration Objects

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
job_config_instance_list = Estimator.get_job_configs_object_list(modelarts_session=session, is_show=True,
per_page=10, page=1, sort_by="create_time", order="asc", search_content="configname")
print(job_config_instance_list)
```

Parameters

Table 9-53 get_job_configs_object_list request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
per_page	No	Integer	Number of job parameters displayed on each page. The value range is [1, 1000]. Default value: **10**
page	No	Integer	Index of the page to be queried. Default value: **1**

Parameter	Mandatory	Type	Description
sortBy/ sort_by	No	String	When AK/SK-based authentication is used, the parameter name is **sortBy**. When the username and password are used for authentication, the parameter name is **sort_by**. The parameter specifies the sorting mode of the query. The value can be **job_name**, **job_desc**, **status**, **duration**, **engine_type**, or **create_time**. Default value: **job_name**
order	No	String	Sorting order. The options are as follows: • **asc**: ascending order. It is the default value. • **desc**: descending order
search_content	No	String	Search content, for example, a parameter name. By default, this parameter is left blank.
is_show	No	Boolean	Whether to print the training job parameter configuration list. Default value: **True**

A training object list is returned in the successful response to **get_job_configs_object_list**. For details, see **Table 9-54**.

Table 9-54 TrainingJob object description

Parameter	Type	Description
TrainingJob	Object	Training object. This object contains attributes such as **config_name**, and operations on a training job parameter configuration, such as querying or deleting the training job parameter configuration. For example, you can use **job_config_instance.config_name** to obtain the name of a training job parameter configuration.

9.3.3 Querying the List of Training Job Configurations

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
```



```
session = Session()
job_paras_list = Estimator.get_job_configs_list(modelarts_session=session, per_page=10, page=1,
sort_by="create_time", order="asc", search_content="configname")
print(job_paras_list)
```

Parameters

Table 9-55 get_job_configs_list request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
per_page	No	Integer	Number of job parameters displayed on each page. The value range is [1, 1000]. Default value: **10**
page	No	Integer	Index of the page to be queried. Default value: **1**
sortBy/sort_by	No	String	When AK/SK-based authentication is used, the parameter name is **sortBy**. When account-based authentication is used, the parameter name is **sort_by**. The parameter specifies the sorting mode of the query. The value can be **job_name**, **job_desc**, **status**, **duration**, **engine_type**, or **create_time**. Default value: **job_name**
order	No	String	Sorting order. The options are as follows: • **asc**: ascending order. It is the default value. • **desc**: descending order
search_content	No	String	Search content, for example, a parameter name. By default, this parameter is left blank.

Table 9-56 get_job_configs_list response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
config_total_count	Integer	Total number of the queried training job configurations
configs	JSON Array	**configs** parameters
is_success	Boolean	Whether the API call succeeds

Table 9-57 configs parameters

Parameter	Type	Description
config_name	String	Name of a training job parameter configuration
config_desc	String	Description of a training job parameter configuration
create_time	Long	Time when a training job is created
engine_type	Short	Engine type of a training job
engine_name	String	Name of the engine selected for a training job
engine_id	Long	ID of the engine selected for a training job
engine_version	String	Version of the engine selected for a training job

9.3.4 Querying the Details About a Training Job Configuration

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **config_name**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(modelarts_session=session, config_name="my_job_config")
job_paras_info = estimator.get_job_configs_info()
print(job_paras_info)
```

- Method 2: Use the object returned in **Creating a Training Job Configuration**.

```
job_paras_info = job_config_instance.get_job_configs_info()
print(job_paras_info)
```

- Method 3: Use the object returned in **Querying the List of Training Job Parameter Configuration Objects**.

```
job_paras_info = job_config_instance_list[0].get_job_configs_info()
print(job_paras_info)
```

Parameters

Table 9-58 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
config_name	Yes	String	Name of a training job parameter configuration

Table 9-59 get_job_configs_info response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
config_name	String	Name of a training job parameter configuration
config_desc	String	Description of a training job parameter configuration
worker_server_num	Integer	Number of workers in a training job
app_url	String	Code directory of a training job
boot_file_url	String	Boot file of a training job
model_id	Long	Model ID of a training job
parameter	JSON Array	Running parameters of a training job. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image.
spec_id	Long	ID of the resource specifications selected for a training job
data_url	String	Dataset of a training job

Parameter	Type	Description
dataset_id	String	Dataset ID of a training job
dataset_version_id	String	Dataset version ID of a training job
engine_type	Short	Engine type of a training job
engine_name	String	Name of the engine selected for a training job
engine_id	Long	ID of the engine selected for a training job
engine_version	String	Version of the engine selected for a training job
train_url	String	OBS URL of the output file of a training job. By default, this parameter is left blank. Example value: **/usr/train/**
log_url	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/train/**
user_image_url	String	SWR URL of the custom image used by a training job
user_command	String	Boot command used to start the container of the custom image of a training job
is_success	Boolean	Whether the API call succeeds

9.3.5 Modifying a Training Job Configuration

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Example 1: Modify a training job parameter configuration using the data stored on OBS.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                    modelarts_session=session,
                    framework_type='PyTorch',                                   # AI engine name
                    framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                    code_dir='/bucket/src/',                                    # Training script directory
                    boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                    log_url='/bucket/log/',                                     # Training log directory
                    hyperparameters=[
                                {"label":"classes",
                                 "value": "10"},
                                {"label":"lr",
                                 "value": "0.001"}
                                ],
                    output_path='/bucket/output/',                              # Training output directory
```

```
                train_instance_type='modelarts.vm.gpu.p100',                # Training environment flavor
                train_instance_count=1)                                     # Number of training nodes
update_info = estimator.update_job_configs(config_name='my_job_config', inputs='/bucket/dataset/', config_desc='update')
```

- Example 2: Modify a training job parameter configuration using a dataset.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(
                modelarts_session=session,
                framework_type='PyTorch',                                   # AI engine name
                framework_version='PyTorch-1.0.0-python3.6',                # AI engine version
                code_dir='/bucket/src/',                                    # Training script directory
                boot_file='/bucket/src/pytorch_sentiment.py',               # Training boot script directory
                log_url='/bucket/log/',                                     # Training log directory
                hyperparameters=[
                        {"label":"classes",
                         "value": "10"},
                        {"label":"lr",
                         "value": "0.001"}
                        ],
                output_path='/bucket/output/',                              # Training output directory
                train_instance_type='modelarts.vm.gpu.p100',                # Training environment flavor
                train_instance_count=1)                                     # Number of training nodes
update_info = estimator.update_job_configs(config_name='my_job_config',
dataset_id='4AZNvFkN7KYr5EdhFkH', dataset_version_id='UOF9BIeSGArwVt0oI6T',
config_desc='update')
```

Parameters

Table 9-60 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
train_instance_count	Yes	Long	Number of workers in a training job
code_dir	No	String	Code directory of a training job, for example, **/bucket/src/**. Leave this parameter blank when **model_name** is set.
boot_file	No	String	Boot file of a training job, which needs to be stored in the code directory. For example, **/bucket/src/boot.py**. Leave this parameter blank when **model_name** is set.
model_name	No	Long	Name of the built-in algorithm used by a training job. If you have configured **model_name**, you do not need to configure **app_url**, **boot_file_url**, **framework_type**, and **framework_version**.
output_path	Yes	String	Output path of a training job

Parameter	Mandatory	Type	Description
hyperparameters	No	JSON Array	Running parameters of a training job. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image.
log_url	No	String	OBS URL of the logs of a training job. By default, this parameter is left blank. Example value: **/usr/log/**
train_instance_type	Yes	Long	Resource flavor selected for a training job. If you choose to train on the training platform, obtain the value by calling the API described in **Querying the List of Resource Flavors**.
framework_type	No	String	Engine selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
framework_version	No	String	Engine version selected for a training job. Obtain the value by calling the API described in **Querying the List of Engine Types**. Leave this parameter blank when **model_name** is set.
job_description	No	String	Description of a training job
user_image_url	No	String	SWR URL of the custom image used by a training job. Example value: **100.125.5.235:20202/jobmng/custom-cpu-base:1.0**
user_command	No	String	Boot command used to start the container of the custom image of a training job. The format is **bash /home/work/run_train.sh python /home/work/user-job-dir/app/train.py {python_file_parameter}**.

Table 9-61 update_job_configs request parameters

Parameter	Mandatory	Type	Description
config_name	Yes	String	Name of a training job parameter configuration. The value is a string of 1 to 20 characters consisting of only digits, letters, underscores (_), and hyphens (-). By default, if this parameter is left blank, the value is dynamically generated by date.
config_desc	No	String	Description of a training job parameter configuration. The value is a string of 0 to 256 characters. By default, this parameter is left blank.
inputs	No	String	OBS storage path of a training job
dataset_id	No	String	Dataset ID of a training job. This parameter must be used with **dataset_version_id**, but cannot be used with **inputs**.
dataset_version_id	No	String	Dataset version ID of a training job. This parameter must be used with **dataset_id**, but cannot be used with **inputs**.
data_source	No	JSON Array	Dataset of a training job. This parameter cannot be used with **inputs**, **dataset_id**, or **dataset_version_id**.

Table 9-62 data_source parameters

Parameter	Mandatory	Type	Description
dataset_id	No	String	Dataset ID of a training job
dataset_version	No	String	Dataset version ID of a training job
type	Yes	String	Dataset type. The value can be **obs** or **dataset**.
data_url	No	String	OBS bucket path. This parameter cannot be used with **dataset_id** or **dataset_version**.

Table 9-63 update_job_configs response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.3.6 Deleting a Training Job Configuration

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Delete a training job parameter configuration based on the specified **config_name**.
```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
estimator = Estimator(modelarts_session=session, config_name="my_job_config")
status = estimator.delete_job_configs()
```
- Method 2: Delete the training job parameter configuration created in **Creating a Training Job Configuration**.
```
status = job_config_instance.delete_job_configs()
```
- Method 3: Delete the training job parameter configuration object returned in **Querying the List of Training Job Parameter Configuration Objects**.
```
status = job_config_instance_list[0].delete_job_configs()
```

Parameters

Table 9-64 Estimator request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
config_name	Yes	String	Name of a training job parameter configuration

Table 9-65 delete_job_configs response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.4 Visualization Jobs

9.4.1 Creating a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session)
job_visualization_instance = job.create_visualization_job(train_url='/bucket/train/',
job_name='visualization_job', job_desc='my visualization job')
```

Parameters

Table 9-66 create_visualization_job request parameters

Parameter	Mandatory	Type	Description
job_name	No	String	Name of a visualization job. The value is a string of 1 to 20 characters consisting of only digits, letters, underscores (_), and hyphens (-).
job_desc	No	String	Description of a visualization job. The value is a string of 0 to 256 characters. By default, this parameter is left blank.

Parameter	Mandatory	Type	Description
train_url	Yes	String	OBS path to the visualization file. The visualization file is provided for the visualization job to read and display, and is usually located in the training output path. The visualization file is generated by the **tf.summary** or **tensorboardx.SummaryWriter** module in the training code, and the file name usually starts with **events.out.tfevents**.

Table 9-67 create_visualization_job response parameters

Parameter	Type	Description
VisualizationJob	Object	Visualization job object. This object contains attributes such as **visualization_id**, **create_time**, **job_name**, and **status**, and operations on a visualization job, such as querying, modifying, stopping, restarting, or deleting the visualization job.

Table 9-68 VisualizationJob parameters

Parameter	Type	Description
create_time	Long	Time when a visualization job is created
job_name	String	Name of a visualization job
status	Byte	Status of a visualization job. For details about the job statuses, see **Job Statuses**.
job_id	String	ID of a visualization job
is_success	Boolean	Whether the API call succeeds

9.4.2 Querying the List of Visualization Job Objects

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job_visualization_instance_list = VisualizationJob.get_visualization_job_object_list(modelarts_session=session,
```

```
is_show=True, status=8, per_page=10, page=1, sort_by="create_time", order="asc", search_content="job")
print(job_visualization_instance_list)
```

Parameters

Table 9-69 get_visualization_job_object_list request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
status	No	String	Status of a visualization job. For details about the job statuses, see **Job Statuses**.
per_page	No	Integer	Number of jobs displayed on each page. The value range is [1, 100]. Default value: **10**
page	No	Integer	Index of the page to be queried. Default value: **1**
sortBy/sort_by	No	String	When AK/SK-based authentication is used, the parameter name is **sortBy**. When the username and password are used for authentication, the parameter name is **sort_by**. The parameter specifies the sorting mode of the query. The value can be **job_name**, **job_desc**, **status**, **duration**, **create_time**, or **log_dir**. Default value: **job_name**
order	No	String	Sorting order. The options are as follows: • **asc**: ascending order. It is the default value. • **desc**: descending order
search_content	No	String	Search content, for example, a visualization job name. The value consists of 0 to 100 characters. By default, this parameter is left blank.
is_show	No	Boolean	Whether to print the visualization job list. Default value: **True**

Table 9-70 get_visualization_job_object_list response parameters

Parameter	Type	Description
VisualizationJob	Object	Visualization job object. This object contains attributes such as **visualization_id**, **create_time**, **job_name**, and **status**, and operations on a visualization job, such as querying, modifying, stopping, restarting, or deleting the visualization job.

Table 9-71 VisualizationJob parameters

Parameter	Type	Description
create_time	Long	Time when a visualization job is created
job_name	String	Name of a visualization job
status	Byte	Status of a visualization job. For details about the job statuses, see **Job Statuses**.
job_id	String	ID of a visualization job
is_success	Boolean	Whether the API call succeeds

9.4.3 Querying the List of Visualization Jobs

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job_list = VisualizationJob.get_visualization_job_list(modelarts_session=session, status=8, per_page=10,
page=1, sort_by="create_time", order="asc", search_content="job")
print(job_list)
```

Parameters

Table 9-72 get_visualization_job_list request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Parameter	Mandatory	Type	Description
status	No	String	Status of a visualization job. For details about the job statuses, see **Job Statuses**.
per_page	No	Integer	Number of jobs displayed on each page. The value range is [1, 100]. Default value: **10**
page	No	Integer	Index of the page to be queried. Default value: **1**
sortBy/sort_by	No	String	When AK/SK-based authentication is used, the parameter name is **sortBy**. When the username and password are used for authentication, the parameter name is **sort_by**. The parameter specifies the sorting mode of the query. The value can be **job_name**, **job_desc**, **status**, **duration**, **create_time**, or **log_dir**. Default value: **job_name**
order	No	String	Sorting order. The options are as follows: • **asc**: ascending order. It is the default value. • **desc**: descending order
search_content	No	String	Search content, for example, a visualization job name. The value consists of 0 to 100 characters. By default, this parameter is left blank.

Table 9-73 get_visualization_job_list response parameters

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
job_total_count	Integer	Total number of the queried visualization jobs
job_count_limit	Integer	Number of visualization jobs that can be created

Parameter	Type	Description
jobs	JSON Array	Visualization job attributes. For details, see **Table 9-74**.

Table 9-74 jobs parameters

Parameter	Type	Description
job_id	Integer	ID of a visualization job
job_name	String	Name of a visualization job
status	Integer	Status of a visualization job. For details about the job statuses, see **Job Statuses**.
create_time	Long	Time when a visualization job is created
duration	Long	Running duration of a visualization job, in milliseconds
job_desc	String	Description of a visualization job
service_url	String	Endpoint of a visualization job
train_url	String	Path to visualization job logs

9.4.4 Querying the Details About a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Use the specified **visualization_id**.
```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session, visualization_id='8992')
job_info = job.get_visualization_job_info()
print(job_info)
```
- Method 2: Use the visualization job created in **Creating a Visualization Job**.
```
job_info = job_visualization_instance.get_visualization_job_info()
print(job_info)
```
- Method 3: Use the visualization job object returned in **Querying the List of Visualization Job Objects**.
```
job_info = job_visualization_instance_list[0].get_visualization_job_info()
print(job_info)
```

Parameters

Table 9-75 VisualizationJob request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
visualization_id	Yes	String	ID of a visualization job

Table 9-76 get_visualization_job_info response parameters

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
job_name	String	Name of a visualization job
service_url	String	Endpoint of a visualization job
is_success	Boolean	Whether the API call succeeds
duration	Long	Running duration of a visualization job
create_time	Long	Time when a visualization job is created
train_url	String	OBS path to the visualization job output file
job_id	Long	ID of a visualization job
job_desc	String	Description of a visualization job
resource_id	String	Resource ID of a visualization job
status	Integer	Status of a visualization job. For details about the job statuses, see **Job Statuses**.

9.4.5 Modifying the Description of a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Modify the description of a visualization job based on the specified **visualization_id**.

```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session, visualization_id='8992')
job_description = job.update_visualization_job(job_desc='update visualization job')
```

- Method 2: Modify the description of the visualization job created in **Creating a Visualization Job**.

```
job_description = job_visualization_instance.update_visualization_job(job_desc='update visualization job')
```

- Method 3: Modify the description of a visualization job object returned in **Querying the List of Visualization Job Objects**.

```
job_description = job_visualization_instance_list[0].update_visualization_job(job_desc='update visualization job')
```

Parameters

Table 9-77 VisualizationJob request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
visualization_id	Yes	String	ID of a visualization job

Table 9-78 update_visualization_job request parameters

Parameter	Mandatory	Type	Description
job_desc	Yes	String	Description of a visualization job. The value is a string of 0 to 256 characters.

Table 9-79 update_visualization_job response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.4.6 Stopping a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Stop a visualization job based on the specified **visualization_id**.
```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session, visualization_id='8992')
status = job.stop_visualization_job()
```
- Method 2: Stop the visualization job created in **Creating a Visualization Job**.
```
status = job_visualization_instance.stop_visualization_job()
```
- Method 3: Stop the visualization job object returned in **Querying the List of Visualization Job Objects**.
```
status = job_visualization_instance_list[0].stop_visualization_job()
```

Parameters

Table 9-80 VisualizationJob request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
visualization_id	Yes	String	ID of a visualization job

Table 9-81 stop_visualization_job response parameters

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.4.7 Restarting a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Restart a visualization job based on the specified **visualization_id**.
```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session, visualization_id='8992')
resp = job.restart_visualization_job()
```
- Method 2: Restart the visualization job created in **Creating a Visualization Job**.
```
status = job_visualization_instance.restart_visualization_job()
```
- Method 3: Restart the visualization job object returned in **Querying the List of Visualization Job Objects**.
```
status = job_visualization_instance_list[0].restart_visualization_job()
```

Parameters

Table 9-82 VisualizationJob request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
visualization_id	Yes	String	ID of a visualization job

Table 9-83 restart_visualization_job response parameters

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.4.8 Deleting a Visualization Job

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Delete a visualization job based on the specified **visualization_id**.
```
from modelarts.session import Session
from modelarts.estimator import VisualizationJob
session = Session()
job = VisualizationJob(modelarts_session=session, visualization_id='8992')
status = job.delete_visualization_job()
```
- Method 2: Delete the visualization job created in **Creating a Visualization Job**.
```
status = job_visualization_instance.delete_visualization_job()
```
- Method 3: Delete the visualization job object returned in **Querying the List of Visualization Job Objects**.
```
status = job_visualization_instance_list[0].delete_visualization_job()
```

Parameters

Table 9-84 VisualizationJob request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
visualization_id	Yes	String	ID of a visualization job

Table 9-85 delete_visualization_job response parameters

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.
is_success	Boolean	Whether the API call succeeds

9.5 Resource and Engine Specifications

9.5.1 Querying a Built-in Algorithm

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
algo_info = Estimator.get_built_in_algorithms(modelarts_session=session)
print(algo_info)
```

Parameters

Table 9-86 get_built_in_algorithms request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 9-87 get_built_in_algorithms response parameters

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

Parameter	Type	Description
error_code	String	Error code when the API fails to be called. For details, see **Error Codes**. This parameter is not included when the API call succeeds.
model_total_count	Integer	Number of models
models	JSON Array	Parameter list of a model
is_success	Boolean	Whether the API call succeeds

Table 9-88 models parameters

Parameter	Type	Description
model_id	Integer	Model ID
model_name	String	Model name
model_usage	Integer	Model usage. The options are as follows: • **1**: image classification • **2**: object class and location • **3**: image semantic segmentation • **4**: natural language processing
model_precision	String	Model precision
model_size	Long	Model size, in bytes
model_train_dataset	String	Model training dataset
model_dataset_format	String	Dataset format required by a model
model_description_url	String	URL of the model description
parameter	JSON Array	Running parameters of a model. It is a collection of label-value pairs. This parameter is a container environment variable when a job uses a custom image. For details, see the sample request.
create_time	Long	Time when a model is created
engine_id	Long	Engine ID of a model

Parameter	Type	Description
engine_name	String	Engine name of a model
engine_version	String	Engine version of a model

9.5.2 Querying the List of Resource Flavors

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
algo_info = Estimator.get_train_instance_types(modelarts_session=session)
print(algo_info)
```

Parameters

Table 9-89 Request parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 9-90 Successful response parameters

Type	Description
List	List of resource flavor attributes

Table 9-91 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.

Parameter	Type	Description
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

9.5.3 Querying the List of Engine Types

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.estimator import Estimator
session = Session()
engine_list = Estimator.get_framework_list(modelarts_session=session)
print(engine_list)
```

Parameters

Table 9-92 get_framework_list parameters

Parameter	Mandatory	Type	Description
modelarts_session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 9-93 Successful response parameters of **get_framework_list**

Type	Description
List	List of engine flavor attributes. For details, see **Table 9-94**.

Table 9-94 framework_list parameters

Parameter	Type	Description
framework_type	String	Engine type
framework_version	String	Engine version

Table 9-95 Failed response parameters

Parameter	Type	Description
error_code	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

9.6 Job Statuses

Table 9-96 describes the job statuses.

Table 9-96 Job statuses

Status Value	Description
0	JOBSTAT_UNKNOWN: Unknown status.
1	JOBSTAT_INIT: The job is being initialized.
2	JOBSTAT_IMAGE_CREATING: The job image is being created.
3	JOBSTAT_IMAGE_FAILED: Failed to create the job image.
4	JOBSTAT_SUBMIT_TRYING: The job is being submitted.
5	JOBSTAT_SUBMIT_FAILED: Failed to submit the job.
6	JOBSTAT_DELETE_FAILED: Failed to delete the job.
7	JOBSTAT_WAITING: The job is queuing.
8	JOBSTAT_RUNNING: The job is running.
9	JOBSTAT_KILLING: The job is being canceled.
10	JOBSTAT_COMPLETED: The job has been completed.
11	JOBSTAT_FAILED: Failed to run the job.
12	JOBSTAT_KILLED: Job canceled successfully.
13	JOBSTAT_CANCELED: Job canceled.
14	JOBSTAT_LOST: Job lost.
15	JOBSTAT_SCALING: The job is being scaled.
16	JOBSTAT_SUBMIT_MODEL_FAILED: Failed to submit the model.
17	JOBSTAT_DEPLOY_SERVICE_FAILED: Failed to deploy the service.

Status Value	Description
18	JOBSTAT_CHECK_INIT: The job review is being initialized.
19	JOBSTAT_CHECK_RUNNING: The job is being reviewed.
20	JOBSTAT_CHECK_RUNNING_COMPLETED: The approval job is completed.
21	JOBSTAT_CHECK_FAILED: Failed to review the job.
22	MOUNT_FAILED: Failed to mount.

10 Model Management

10.1 Debugging a Model

After the training is complete, create a local model, debug the model locally, and deploy the model on ModelArts.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

Step 1 Save the custom inference file and model configuration file to the directory storing the model file generated during training. If the model generated during training is stored in **/home/ma-user/work/tensorflow_mlp_mnist_local_mode/train/model/**, the inference file **customize_service.py** and model configuration file **config.json** are also stored in this directory.

Step 2 Create a Conda virtual environment for running models.

```
from modelarts.environment import Environment
from modelarts.environment.conda_env import CondaDependencies

env = Environment("tensorflow_mlp_mnist")
cd = CondaDependencies.create(pip_packages=["tensorflow==1.13.1", "Pillow>=8.0.1"],
                              conda_packages=["python=3.6.2"])
env.conda = cd
```

Step 3 Create a local model.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
src_local_path = "/home/ma-user/work/tensorflow_mlp_mnist_local_mode/train/"
model = Model(session,
              publish=False,
              source_location_type="LOCAL_SOURCE",         # Type of the model file location
              source_location=src_local_path + 'model',    # Location of the model file
              environment=env,
              model_version='1.0.1',
              model_type='TensorFlow',                     # AI framework used by the model
              model_algorithm="image_classification",
              model_name="tensorflow_mlp_mnist_local_infer")
```


After a local model is created, you can deploy it as a local service.

Step 4 Call the API to publish the model.

```
model.publish_model(obs_location=obs_location)
```

After the **obs_location** parameter is specified, the local model file is uploaded to this directory. This parameter can be omitted. See the following example:

```
model.publish_model()
```

The model file is uploaded to the directory whose name ends with the current timestamp in the default OBS bucket. The directory is displayed after the command is executed:

```
Successfully upload file /home/ma-user/work/tensorflow_mlp_mnist_local_mode/train/model to OBS modelarts-cn-north-4-08aae033/model-0107-224502
```

----End

Parameters

Table 10-1 Parameters for creating a model

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
model_name	No	String	Name of a model that consists of 1 to 64 characters and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed. If this parameter is not specified, the system automatically generates a model name.
model_version	Yes	String	Model version in the format of *Digit.Digit.Digit*. The value range of the digits is [1, 99]. The version number cannot start with 0, for example, **01.01.01**.
publish	No	Bool	Whether to publish a model. The options are as follows: • **True**: Publish the model. (Default value) • **False**: Do not publish the model. Create a local model, which can be used to debug related code.
source_location_type	No	String	Model location type. The options are as follows: • **OBS_SOURCE**: OBS path. (Default value) • **LOCAL_SOURCE**: local path.

Parameter	Mandatory	Type	Description
source_location	Yes	String	Path (parent directory) of the model file • If **source_location_type** is set to **OBS_SOURCE**, the model file path is an OBS path in the format of **/obs_bucketname/.../model_file_parent_dir/**. • If **source_location_type** is set to **LOCAL_SOURCE**, the model file path is a local path in the format of **/local_path/.../model_file_parent_dir/**.
environment	No	Environment instance	Environment required for normal model running, such as the Python or TensorFlow version For details, see **Table 10-2**.
source_job_id	No	String	ID of the source training job. If the model is generated from a training job, specify this parameter for source tracing. If the model is imported from a third-party meta model, leave this parameter blank. By default, this parameter is left blank.
source_job_version	No	String	Version of the source training job. If the model is generated from a training job, specify this parameter for source tracing. If the model is imported from a third-party meta model, leave this parameter blank. By default, this parameter is left blank.
source_type	No	String	Model source type. The value can only be **auto**, which indicates an ExeML model (model download is not allowed). If the model is deployed via a training job, leave this parameter blank. By default, this parameter is left blank.
model_type	Yes	String	Model type. The value can be **TensorFlow**, **MXNet**, **Spark_MLlib**, **Scikit_Learn**, **XGBoost**, **MindSpore**, **Image**, or **PyTorch**.
model_algorithm	No	String	Model algorithm. If the algorithm has been configured in the model configuration file, this parameter can be left blank. Possible options are **predict_analysis**, **object_detection**, and **image_classification**.
description	No	String	Model description, which contains a maximum of 100 characters and cannot contain the following special characters: !<>=&'"

Parameter	Mandatory	Type	Description
execution_code	No	String	OBS path to the execution script. The inference script must be stored in the **model** directory in the path where the model is located. For details, see the **source_location** parameter. The script name is fixed to **customize_service.py**.
input_params	No	**params** array	List of input parameters for model inference. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the input parameters from the **apis** field in the configuration file.
output_params	No	**params** array	List of output parameters for model inference. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the output parameters from the **apis** field in the configuration file.
dependencies	No	**dependency** array	Dependency package required for running the code and model. By default, this parameter is left blank. If the **dependencies** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the dependencies to be installed from the **dependencies** field in the configuration file.
apis	No	String	List of inference APIs provided by a model. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the configured inference API information from the **apis** field in the configuration file.

Table 10-2 Environment parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Environment name
conda	No	CondaDependencies	Conda environment. For details, see **Table 10-3**.

Table 10-3 CondaDependencies parameters

Parameter	Mandatory	Type	Description
channels	No	List	Source for downloading the Python package
pip_packages	No	List	Python package required by the Conda virtual environment, such as TensorFlow and Pillow
conda_packages	No	List	Conda package required by the Conda virtual environment, for example, a specified Python version

Table 10-4 params parameters

Parameter	Mandatory	Type	Description
url	Yes	String	Request path of a model inference API
param_name	Yes	String	Parameter name, which contains a maximum of 64 characters
param_type	Yes	String	Basic parameter types of JSON schema, including **string**, **object**, **array**, **boolean**, **number**, and **integer**
min	No	Double	This parameter is optional when **param_type** is set to **int** or **float**. By default, this parameter is left blank.
max	No	Double	This parameter is optional when **param_type** is set to **int** or **float**. By default, this parameter is left blank.
param_desc	No	String	Parameter description, which contains a maximum of 100 characters. By default, this parameter is left blank.

Table 10-5 dependency parameters

Parameter	Mandatory	Type	Description
installer	Yes	String	Installation mode, which can only be **pip**
packages	Yes	**package** array	Collection of dependency packages

Table 10-6 package parameters

Parameter	Mandatory	Type	Description
package_name	Yes	String	Name of a dependency package
package_version	No	String	Version of a dependency package
restraint	No	String	Version filtering condition. This parameter is mandatory only when **package_version** is available. Options: • **EXACT**: a specified version • **ATLEAST**: not earlier than the specified version • **ATMOST**: not later than the specified version

Table 10-7 Parameters for creating a model

Parameter	Mandatory	Type	Description
model	Yes	Model object	Model object, which can be any of the APIs described in this chapter

10.2 Importing a Model

Importing a model includes:

- Initialize the existing model and create a model object based on the model ID.
- Create a model. For details about the attributes of the created model, see **Obtaining Details About a Model**.

Sample Model File

The following uses **PyTorch** as an example to describe how to edit a model file. For details about the PyTorch model package structure, see **Introduction to Model Package Specifications**.

```
OBS bucket or directory name
├── resnet
│   ├── model Mandatory: Fixed subdirectory name. The subdirectory is used to store model-related files.
│   │   ├──<<Custom Python package>> (Optional) Custom Python package, which can be directly referenced in model inference code
│   │   ├──mnist_mlp.pt (Mandatory) PyTorch model file, which contains variable and weight information and is saved as state_dict
│   │   ├──config.json Mandatory: Model configuration file. The file name is fixed to config.json. Only one model configuration file is allowed.
│   │   ├──customize_service.py Mandatory: Model inference code. The file name is fixed to customize_service.py. Only one model inference file is allowed. The files on which customize_service.py depends can be directly stored in the model directory.
```

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig, Params, Dependencies, Packages

session = Session()
```

- Method 1: Initialize an existing model.
  ```
  model_instance = Model(session, model_id="your_model_id")
  ```
- Method 2: Create a model.
  - Use a preset image and specify an OBS path to create a model.
    ```
    model_location = "/your_obs_bucket/model_path"            # Change to the OBS path to the model file
    execution_code = "/your_obs_bucket/model_path/customize_service.py"
    runtime = "python3.7"

    model_instance = Model(
                        session,
                        model_name="input_model_name",    #  (Optional) Model name
                        model_version="1.0.0",            # (Optional) Model version
                        source_location=model_location,   # OBS path to the model file, for example, /your_obs_bucket/model_path
                        model_type="PyTorch",             # Model type
                        execution_code=execution_code,    # (Optional) OBS path to the execution script, for example, /your_obs_bucket/model_path/customize_service.py
                        runtime = runtime                 # (Optional) Supported runtime environment
                       )
    ```

    NOTE

    dependencies will overwrite the data in **config.json** in the preceding example. You do not need to use **dependencies**. The following section describes the **dependencies** formats.

    - Format of the **dependencies** parameter group

      SDKs define the **dependencies** parameter group. **dependencies** is in list format, and those of the tuple objects in the list are Dependencies.

      The code is as follows:


```
dependencies = []
dependency1 = Dependencies(
                        installer="pip",                    # Installation mode. pip is supported.
    packages=packages                 # Collection of dependency packages. For details, see
packages.
)
dependencies.append(dependency1)
```

- Format of the **package** parameter group

  SDKs define the **packages** parameter group. **packages** is in list format, and those of the tuple objects in the list are Packages.

  The code is as follows:

```
packages = []
package1 = Packages(
                    package_name="package_name",              # Package name
                    package_version="version",                # Package version
    restraint="EXACT"
)
packages.append(package1)
```

  NOTE

  The following is an example of creating a **dependencies** parameter group:

```
dependencies = []
packages = [{
    "package_name": "numpy",
    "package_version": "1.15.0",
    "restraint": "EXACT"
    }, {
        "package_name": "h5py",
        "package_version": "2.8.0",
        "restraint": "EXACT"
    }]
dependency = Dependencies(installer="pip", packages=packages)
dependencies.append(dependency)
```

– Use a custom image to create a model.

  This method applies if the script of the inference service has been built in the custom image and the service is automatically started when the image is started.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
image_path = "custom_image_path"
model_instance = Model(
                    session,
                    model_name="your_model_name",              # Model name
                    model_version="0.1.0",                  # Model version
                    source_location=image_path,                # Model file path
                    model_type="Image"                      # Model type
                  )
```


Parameters

Table 10-8 Parameters for initializing a model

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
model_id	Yes	String	Model ID

Table 10-9 Parameters for creating a model

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
model_name	No	String	Name of a model that consists of 1 to 64 characters and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed. If this parameter is not specified, the system automatically generates a model name.
model_version	Yes	String	Model version in the format of *Digit.Digit.Digit*. The value range of the digits is [1, 99]. The version number cannot start with 0, for example, **01.01.01**.
publish	No	Bool	Whether to publish a model. The options are as follows: • **True**: Publish the model. (Default value) • **False**: Do not publish the model. Create a local model, which can be used to debug related code.
source_location_type	No	String	Model location type. The options are as follows: • **OBS_SOURCE**: OBS path. (Default value) • **LOCAL_SOURCE**: local path.

Parameter	Mandatory	Type	Description
source_location	Yes	String	Path (parent directory) of the model file • If **source_location_type** is set to **OBS_SOURCE**, the model file path is an OBS path in the format of **/obs_bucketname/.../model_file_parent_dir/**. • If **source_location_type** is set to **LOCAL_SOURCE**, the model file path is a local path in the format of **/local_path/.../model_file_parent_dir/**.
environment	No	Environment instance	Environment required for normal model running, such as the Python or TensorFlow version For details about the example environment, see **Sample Code**.
source_job_id	No	String	ID of the source training job. If the model is generated from a training job, specify this parameter for source tracing. If the model is imported from a third-party meta model, leave this parameter blank. By default, this parameter is left blank.
source_job_version	No	String	Version of the source training job. If the model is generated from a training job, specify this parameter for source tracing. If the model is imported from a third-party meta model, leave this parameter blank. By default, this parameter is left blank.
source_type	No	String	Model source type. The value can only be **auto**, which indicates an ExeML model (model download is not allowed). If the model is deployed via a training job, leave this parameter blank. By default, this parameter is left blank.
model_type	Yes	String	Model type. The value can be **TensorFlow**, **MXNet**, **Spark_MLlib**, **Scikit_Learn**, **XGBoost**, **MindSpore**, **Image**, or **PyTorch**.
model_algorithm	No	String	Model algorithm. If the algorithm has been configured in the model configuration file, this parameter can be left blank. For example, **predict_analysis**, **object_detection**, or **image_classification**.

Parameter	Mandatory	Type	Description
description	No	String	Model description, which contains a maximum of 100 characters and cannot contain the following special characters: !<>=&'"
execution_code	No	String	OBS path to the script to be executed. If **customize_service.py** is not output by the model, configure this parameter to specify the path. The inference script must be stored in the **model** directory in the path where the model is located. For details, see the **source_location** parameter. The script name is fixed to **customize_service.py**.
runtime	No	String	Supported runtime environment. This parameter is mandatory if **model_type** is used. For details, see **Supported AI engines and their runtime**.
input_params	No	**params** array	List of input parameters for model inference. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the input parameters from the **apis** field in the configuration file.
output_params	No	**params** array	List of output parameters for model inference. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the output parameters from the **apis** field in the configuration file.
dependencies	No	**dependency** array	Dependency package required for running the code and model. By default, this parameter is left blank. If the **dependencies** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the dependencies to be installed from the **dependencies** field in the configuration file.

Parameter	Mandatory	Type	Description
apis	No	String	List of inference APIs provided by a model. By default, this parameter is left blank. If the **apis** information has been configured in the model configuration file, you do not need to set this parameter. The backend automatically reads the configured inference API information from the **apis** field in the configuration file.

Table 10-10 params parameters

Parameter	Mandatory	Type	Description
url	Yes	String	Request path of a model inference API
param_name	Yes	String	Parameter name, which contains a maximum of 64 characters
param_type	Yes	String	Basic parameter types of JSON schema, including **string**, **object**, **array**, **boolean**, **number**, and **integer**
min	No	Double	This parameter is optional when **param_type** is set to **int** or **float**. By default, this parameter is left blank.
max	No	Double	This parameter is optional when **param_type** is set to **int** or **float**. By default, this parameter is left blank.
param_desc	No	String	Parameter description, which contains a maximum of 100 characters. By default, this parameter is left blank.

Table 10-11 dependency parameters

Parameter	Mandatory	Type	Description
installer	Yes	String	Installation mode. Only **pip** is supported.
packages	Yes	**package** array	Collection of dependency packages

Table 10-12 package parameters

Parameter	Mandatory	Type	Description
package_name	Yes	String	Name of a dependency package
package_version	No	String	Version of a dependency package
restraint	No	String	Version filtering condition. This parameter is mandatory only when **package_version** exists. Possible values are as follows: • **EXACT**: the specified version • **ATLEAST**: not earlier than the specified version • **ATMOST**: not later than the specified version

Table 10-13 create_model response parameters

Parameter	Mandatory	Type	Description
model_instance	Yes	Model object	Model object, which can be any of the APIs described in this chapter

NOTE

Example of creating a model in a handwritten digit recognition project using MXNet:

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_instance = Model(session,
                       model_name="digit_recognition",
                       model_version="1.0.0",
                       source_location=model_location,
                       model_type="MXNet",
                       model_algorithm="image_classification"
                       )
```

10.3 Obtaining Models

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Scenario 1**: Obtain all models of a user.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_list = Model.get_model_list(session)
```

- **Scenario 2**: Obtain the models of a user based on search criteria.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_list = Model.get_model_list(session, model_status="published", model_name="digit",
order="desc")
```

Parameters

Table 10-14 Query parameters

Parameter	Mandatory	Type	Description
model_name	No	String	Model name. Fuzzy match is supported.
model_version	No	String	Model version
model_status	No	String	Model status. The value can be **publishing**, **published**, or **failed**. You can obtain jobs based on their statuses.
description	No	String	Description. Fuzzy match is supported.
offset	No	Integer	Index of the page to be queried. Default value: **0**
limit	No	Integer	Maximum number of records returned on each page. Default value: **280**
sort_by	No	String	Sorting mode. The value can be **create_at**, **model_version**, or **model_size**. Default value: **create_at**
order	No	String	Sorting order. The value can be **asc** or **desc**, indicating the ascending or descending order. Default value: **desc**
workspace_id	No	String	Workspace ID. Default value: **0**

Table 10-15 get_model_list parameters

Parameter	Type	Description
total_count	Integer	Total number of models that meet the search criteria when no paging is implemented

Parameter	Type	Description
count	Integer	Number of models
models	**model** array	Model metadata

Table 10-16 model parameters

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version
model_type	String	Model type. The value can be **TensorFlow**, **MXNet**, **Spark_MLlib**, **Scikit_Learn**, **XGBoost**, **MindSpore**, **Image**, or **PyTorch**.
model_size	Long	Model size, in bytes
tenant	String	Tenant to whom a model belongs
project	String	Project to which a model belongs
owner	String	User to whom a model belongs
create_at	Long	Time when a model is created, in milliseconds calculated from 1970.1.1 0:0:0 UTC
description	String	Model description
source_type	String	Model source type. This parameter is valid only when the model is deployed through ExeML. The value is **auto**.

10.4 Obtaining Model Objects

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Scenario 1**: Obtain all model objects of a user.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
```



```
model_object_list = Model.get_model_object_list(session)
print(model_object_list)
```

- **Scenario 2**: Obtain the model objects of a user based on search criteria.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_object_list = Model.get_model_object_list(session, model_status="published",
model_name="digit", order="desc")
print(model_object_list)
```

Parameters

- You can use this API to obtain the model list. The size of the list is equal to the number of models that have been deployed by the current user. Each element in the list is a model object. The object attributes are the same as those in **Obtaining Details About a Model**. For example, in **model_list = [model_instance1, model_instance2, model_instance3 ...]**, each **model_instance** in the list is a model API that can be called.
- The model list can be obtained based on the query parameters. **Table 10-17** describes the query parameters.
- When the model list is queried, details about the models are returned. See **Table 10-18** and **Table 10-19**.
- A maximum of 150 model objects can be obtained.

Table 10-17 Query parameters

Parameter	Mandatory	Type	Description
model_name	No	String	Model name. Fuzzy match is supported.
model_version	No	String	Model version
model_status	No	String	Model status. The value can be **publishing**, **published**, or **failed**. You can obtain jobs based on their statuses.
description	No	String	Description. Fuzzy match is supported.
offset	No	Integer	Index of the page to be queried. Default value: **0**
limit	No	Integer	Maximum number of records returned on each page. Default value: **280**
sort_by	No	String	Sorting mode. The value can be **create_at**, **model_version**, or **model_size**. Default value: **create_at**
order	No	String	Sorting order. The value can be **asc** or **desc**, indicating the ascending or descending order. Default value: **desc**

Parameter	Mandatory	Type	Description
workspace_id	No	String	Workspace ID. Default value: **0**

Table 10-18 get_model_list parameters

Parameter	Type	Description
total_count	Integer	Total number of models that meet the search criteria when no paging is implemented
count	Integer	Number of models
models	**model** array	Model metadata

Table 10-19 model parameters

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version
model_type	String	Model type. The value can be **TensorFlow**, **MXNet**, **Spark_MLlib**, **Scikit_Learn**, **XGBoost**, **MindSpore**, **Image**, or **PyTorch**.
model_size	Long	Model size, in bytes
tenant	String	Tenant to whom a model belongs
project	String	Project to which a model belongs
owner	String	User to which a model belongs
create_at	Long	Time when a model is created, in milliseconds calculated from 1970.1.1 0:0:0 UTC
description	String	Model description
source_type	String	Model source type. This parameter is valid only when the model is deployed by an ExeML project. The value is **auto**.

10.5 Obtaining Details About a Model

You can use the API to obtain the information about a model object.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Obtain details about a model based on the model object created in **Importing a Model**.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_instance = Model(session, model_id="your_model_id")
model_info = model_instance.get_model_info()
print(model_info)
```

- **Method 2**: Obtain details about a model based on the model object returned in **Obtaining Model Objects**.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_object_list = Model.get_model_object_list(session)
model_instance = model_object_list[0]
model_info = model_instance.get_model_info()
print(model_info)
```

Parameters

Table 10-20 get_model_info response parameters

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version
tenant	String	Tenant
project	String	Project
owner	String	User
create_at	Long	Time when a model is created, in milliseconds calculated from 1970.1.1 0:0:0 UTC
source_location	String	OBS path where a model resides

Parameter	Type	Description
source_job_id	String	ID of the source training job
source_job_version	String	Version of the source training job
source_type	String	Type of a model source • If a model is deployed by an ExeML project, the value is **auto**. • If a model is deployed by a training job or OBS model file, this parameter is left blank.
model_type	String	Model type. The value can be **TensorFlow**, **MXNet**, **Spark_MLlib**, **Scikit_Learn**, **XGBoost**, **MindSpore**, **Image**, or **PyTorch**.
model_size	Long	Model size, in bytes
model_status	String	Model status. The value can be **publishing**, **published**, or **failed**.
description	String	Model description
execution_code	String	OBS path for storing the execution code. The name of the execution code file is fixed to **customize_service.py**.
schema_doc	String	Download address of the model schema file
image_address	String	Execution image path of a model. Before the image is built, that is, before a model has been published as a service, this parameter is left blank.
input_params	**params** array	Collection of input parameters of a model. By default, this parameter is left blank.
output_params	**params** array	Collection of output parameters of a model. By default, this parameter is left blank.
dependencies	**dependency** array	Package required for running the code and model
model_metrics	String	Model evaluation parameter. This parameter is returned only when **source_job_id** and **source_job_version** are assigned values and the corresponding training job has evaluation results.
apis	String	All **apis** input and output parameters of the model

Table 10-21 params parameters

Parameter	Type	Description
url	String	API URL
param_name	String	Parameter name, which contains a maximum of 64 characters
param_type	String	Parameter type. The value can be **int**, **string**, **float**, **timestamp**, **date**, or **file**.
min	Number	When **param_type** is set to **int** or **float** and **min** is set during model creation, the value will be returned. By default, this parameter is left blank.
max	Number	When **param_type** is set to **int** or **float** and **max** is set during model creation, the value will be returned. By default, this parameter is left blank.
param_desc	String	Parameter description, which contains a maximum of 100 characters. By default, this parameter is left blank.

Table 10-22 dependency parameters

Parameter	Type	Description
installer	String	Installer
packages	**package** array	Collection of dependency packages

Table 10-23 package parameters

Parameter	Type	Description
package_name	String	Name of a dependency package
package_version	String	Version of a dependency package
restraint	String	Version filtering criterion. The options are as follows: • **EXACT**: the specified version • **ATLEAST**: not earlier than the specified version • **ATMOST**: not later than the specified version

Table 10-24 metric parameters

Parameter	Mandatory	Type	Description
f1	Yes	Double	Mean
recall	Yes	Double	Recall
precision	Yes	Double	Precision
accuracy	Yes	Double	Accuracy

10.6 Deleting a Model

You can use the API to delete a model object.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Delete the model object created in **Importing a Model** or **Debugging a Model**.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_instance = Model(session, model_id="your_model_id")
model_instance.delete_model()
```

- **Method 2**: Delete the model object returned in **Obtaining Model Objects**.

```
from modelarts.session import Session
from modelarts.model import Model

session = Session()
model_object_list = Model.get_model_object_list(session)
model_instance = model_object_list[0]
model_instance.delete_model()
```


11 Service Management

11.1 Service Management Overview

Service management indicates deploying a model that has been successfully created as a real-time or local service. This feature provides functions such as real-time prediction, local prediction, service details query, and service log query.

The real-time services include **predictor** and **transformer**, both of which provide the functions described in the following sections. This chapter uses **predictor** as an example.

NOTE

The sample code in this chapter is implemented in ModelArts notebook instances. If the code is used in other development environments, the session needs to be authenticated. For details about session authentication, see **Session Authentication**.

11.2 Deploying a Local Service for Debugging

Debug a service locally for a real-time service. This does not require in-cloud resources. To do so, import a model or debug a model by following the operations provided in **Importing a Model** or **Debugging a Model**, and deploy a predictor locally for local inference.

NOTE

The local service predictor can be deployed only on the Linux platform. Use ModelArts notebook to deploy local services.

- **Local service predictor and real-time service predictor**
  - Deploying a local service predictor is to deploy the model file to a local environment. The environment specifications depend on the local host. For example, you can deploy predictor in a notebook instance of the **modelarts.vm.cpu.2u** flavor.
  - Deploying the predictor in **Deploying a Real-Time Service** is to deploy the model file stored in OBS to the container provided by the **Service Deployment** module. The environment specifications (such as CPU and GPU specifications) are determined by **configs parameters of predictor**.

  - To deploy the predictor in **Deploying a Real-Time Service**, you must create a container based on the AI engine, which is time-consuming. Deploying a local service predictor takes a maximum of 10 seconds. Local service predictors can be used to test models but are not recommended for industrial applications of models.
- In this version, the following AI engines can be used to deploy a local service predictor: **XGBoost**, **Scikit_Learn**, **PyTorch**, **TensorFlow**, and **Spark_MLlib**. For details about the version, see **Supported AI engines and their runtime**.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

Sample code of local TensorFlow 1.8 inference

Configure **tensorflow_model_server** in the environment. You can call the SDK API to quickly configure it. For details, see the following sample code.

- In the CPU-based environment, call **Model.configure_tf_infer_environ(device_type="CPU")** to complete the configuration. You only need to configure the item and run it once in the environment.
- In the GPU-based environment, call **Model.configure_tf_infer_environ(device_type="GPU")** to complete the configuration. You only need to configure the item and run it once in the environment.

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig

session = Session()
# GPU-based environment inference configuration
Model.configure_tf_infer_environ(device_type="GPU")
# CPU-based environment inference configuration
#Model.configure_tf_infer_environ(device_type="CPU")

model_instance = Model(
                  session,
                  model_name="input_model_name",            # Model name
                  model_version="1.0.0",                  # Model version
                  source_location=model_location,           # Model file path
                  model_type="MXNet",                     # Model type
                  model_algorithm="image_classification",    # Model algorithm
                  execution_code="OBS_PATH",
                   input_params=input_params,               # For details, see the input_params format
description.
                  output_params=output_params,             # For details, see the output_params format
description.
                   dependencies=dependencies,              # For details, see the dependencies format
description.
                  apis=apis)

configs = [ServiceConfig(model_id=model_instance.get_model_id(), weight="100", instance_count=1,
                     specification="local")]
predictor_instance = model_instance.deploy_predictor(configs=configs)
if predictor_instance is not None:
    predict_result = predictor_instance.predict(data="your_raw_data_or_data_path",
data_type="your_data_type")     # Local inference and prediction. data can be raw data or a file path, and
data_type can be JSON, files, or images.
    print(predict_result)
```


Parameters

Table 11-1 Parameters for deploying a local service predictor

Parameter	Mandatory	Type	Description
service_name	No	String	Name of a service that consists of 1 to 64 characters and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.
configs	Yes	JSON Array	Local service configurations

Table 11-2 configs parameters of **predictor**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID. Obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.
weight	Yes	Integer	Traffic weight allocated to a model. When a local service predictor is deployed, set this parameter to **100**.
specification	Yes	String	When a local service is deployed, set this parameter to **local**.
instance_count	Yes	Integer	Number of instances deployed in a model. The maximum number of instances is 5. When a local service predictor is deployed, set this parameter to **1**.
envs	No	Map<String, String>	(Optional) Environment variable key-value pair required for running a model. By default, this parameter is left blank.

Table 11-3 Parameters returned for deploying a local service predictor

Parameter	Mandatory	Type	Description
predictor	Yes	Predictor object	Predictor object, which contains only the attributes in **Testing an Inference Service**

11.3 Deploying a Real-Time Service

Real-time service deployment covers the following aspects:

- Initialize a real-time service.
- Deploy a real-time service predictor.
- Deploy a batch service transformer.

The service object predictor is returned after deployment. The attributes of the service object include all functions described in this chapter.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- Method 1: Initialize the predictor that has been deployed as a real-time service.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
```

- Method 2: Deploy a real-time service predictor.
  - Deploy the service in a public resource pool.

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig, TransformerConfig, Schedule

session = Session()
model_instance = Model(session, model_id='your_model_id')
vpc_id = None                                   # (Optional) ID of the VPC where the real-time service instance is deployed. This parameter is left blank by default.
subnet_network_id = None                        # (Optional) Subnet ID. This parameter is left blank by default.
security_group_id = None                        # (Optional) Security group. This parameter is left blank by default.
configs = [ServiceConfig(model_id=model_instance.model_id,
                         weight="100",
                         instance_count=1,
                         specification="modelarts.vm.cpu.2u")]  # For details, see specification.
predictor_instance = model_instance.deploy_predictor(
        service_name="service_predictor_name",
        infer_type="real-time",
        vpc_id=vpc_id,
        subnet_network_id=subnet_network_id,
        security_group_id=security_group_id,
        configs=configs,                        # predictor configuration parameter. For details, see configs.
        schedule = [Schedule(op_type='stop', time_unit='HOURS', duration=1)]       # (Optional) Specify the runtime duration for a real-time service.
)
```

    The **model_id** parameter specifies the model that is to be deployed as a real-time service. Obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.

  - Deploy the service in a dedicated resource pool.

```
from modelarts.config.model_config import ServiceConfig
```



```
configs = [ServiceConfig(model_id=model_instance.model_id, weight="100", instance_count=1,
                         specification="modelarts.vm.cpu.2u")]
predictor_instance = model_instance.deploy_predictor(
                                              service_name="your_service_name",
                                              infer_type="real-time",
                                              configs=configs,
                                              cluster_id="your dedicated pool id"
                                             )
```

configs is defined by **ServiceConfig** in the SDK. The type of **configs** is list, and the tuple object in the list is **ServiceConfig**. The code is as follows:

```
configs = []
envs = {"model_name":"mxnet-model-1", "load_epoch":"0"}

service_config1 = ServiceConfig(
        model_id="model_id1",                # model_id1 and model_id2 must be the IDs of different versions of the same model.
        weight="70",
        specification="modelarts.vm.cpu.2u",  # For details, see specification.
        instance_count=2,
        envs=envs)                           # (Optional) Configure the environment variable, for example, envs = {"model_name":"mxnet-model-1", "load_epoch":"0"}.
service_config2 = ServiceConfig(
        model_id='model_id2',
        weight="30",
        specification="modelarts.vm.cpu.2u",  # For details, see specification.
        instance_count=2,
        envs=envs)                           # (Optional) Configure the environment variable, for example, envs = {"model_name":"mxnet-model-1", "load_epoch":"0"}.
configs.append(service_config1)
configs.append(service_config2)
```

- Method 3: Deploy a batch service transformer.

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import TransformerConfig

session = Session()
model_instance = Model(session, model_id='your_model_id')
vpc_id = None                                # (Optional) ID of the VPC where the batch service instance is deployed. This parameter is left blank by default.
subnet_network_id = None                     # (Optional) Subnet ID. This parameter is left blank by default.
security_group_id = None                     # (Optional) Security group. This parameter is left blank by default.

transformer = model_instance.deploy_transformer(
        service_name="service_transformer_name",
        infer_type="batch",
        vpc_id=vpc_id,
        subnet_network_id=subnet_network_id,
        security_group_id=security_group_id,
        configs=configs                      # transformer configuration parameter. For details, see configs.
)
```

configs is defined by **TransformerConfig** in the SDK. The type of **configs** is list, and the tuple object in the list is **TransformerConfig**. The code is as follows:

```
configs = []
mapping_rule = None                          # (Optional) Mapping between input parameters and CSV data
mapping_type= "file"                         # File or CSV
envs = {"model_name":"mxnet-model-1", "load_epoch":"0"}

transformer_config1 = TransformerConfig(
          model_id="model_id",
          specification="modelarts.vm.cpu.2u",   # For details, see specification.
          instance_count=2,
          src_path="/shp-cn4/sdk-demo/",         # OBS path to the input of the batch task, for
```



```
example, /your_obs_bucket/src_path
        dest_path="/shp-cn4/data-out/",       # OBS path to the output of the batch task, for
example, /your_obs_bucket/dest_path
        req_uri="/",
        mapping_type=mapping_type,
        mapping_rule=mapping_rule,
        envs=envs)                            # (Optional) Configure the environment variable, for example,
envs = {"model_name":"mxnet-model-1", "load_epoch":"0"}.
configs.append(transformer_config1)
```

Parameters

Table 11-4 Parameters

Parameter	Mandatory	Type	Description
service_id	Yes	String	Service ID, which can be obtained from the real-time service on the ModelArts management console
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.

Table 11-5 Parameters for deploying the predictor and transformer

Parameter	Mandatory	Type	Description
service_name	No	String	Name of a service that consists of 1 to 64 characters and must start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.
description	No	String	Service description, which contains a maximum of 100 characters. By default, this parameter is left blank.
infer_type	No	String	Inference mode. The value can be **real-time** or **batch**. The default value is **real-time**. • **real-time**: real-time service. A model is deployed as a web service and provides real-time test UI and monitoring capabilities. The service keeps running. • **batch**: batch service. A batch service can perform inference on batch data and automatically stops after data processing is completed.

Parameter	Mandatory	Type	Description
vpc_id	No	String	ID of the VPC to which a real-time service instance is deployed. By default, this parameter is left blank. In this case, ModelArts allocates a dedicated VPC to each user, and users are isolated from each other. To access other service components in the VPC of the service instance, set this parameter to the ID of the corresponding VPC. Once a VPC is configured, it cannot be modified. When **vpc_id** and **cluster_id** are configured, only the dedicated cluster parameter takes effect.
subnet_network_id	No	String	ID of a subnet. By default, this parameter is left blank. This parameter is mandatory when **vpc_id** is configured. Enter the network ID displayed in the subnet details on the VPC management console. A subnet provides dedicated network resources that are isolated from other networks.
security_group_id	No	String	Security group. By default, this parameter is left blank. This parameter is mandatory when **vpc_id** is configured. A security group is a virtual firewall that provides secure network access control policies for service instances. A security group must contain at least one inbound rule to permit the requests whose protocol is TCP, source address is **0.0.0.0/0**, and port number is **8080**.
configs	Yes	**configs** parameters of **predictor** and **transformer**	Model running configurations • When **infer_type** is set to **batch**, only one model can be configured. • When **infer_type** is set to **real-time**, you can configure multiple models and assign traffic weights based on service requirements. The version numbers of the models must be different.
schedule	No	**schedule** array	Service scheduling configuration, which can be configured only for real-time services. By default, this parameter is not used. Services run for a long time. For details, see **Table 11-9**.

Parameter	Mandatory	Type	Description
cluster_id	No	String	ID of an old-version dedicated resource pool, which is left blank by default. If this parameter is configured, the service will be deployed in the specified old-version dedicated resource pool.
pool_name	No	String	Name of a new-version dedicated resource pool.

Table 11-6 configs parameters of **predictor**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID. Obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.

weight	Yes	Integer	Weight of traffic allocated to a model. This parameter is mandatory only when **infer_type** is set to **real-time**. The sum of multiple weights must be equal to 100. If multiple model versions are configured in a real-time service and different traffic weights are set, ModelArts continuously accesses the prediction API of the service and forwards prediction requests to the model instances of the corresponding versions based on the weights.
			{
			"service_name": "mnist",
			"description": "mnist service",
			"infer_type": "real-time",
			"config": [
			{
			"model_id": "xxxmodel-idxxx",
			"weight": "70",
			"specification": "modelarts.vm.cpu.2u",
			"instance_count": 1,
			"envs":
			{
			"model_name": "mxnet-model-1",
			"load_epoch": "0"
			}
			},
			{
			"model_id": "xxxxxx",
			"weight": "30",
			"specification": "modelarts.vm.cpu.2u",
			"instance_count": 1
			}
			]
			}

Parameter	Mandatory	Type	Description
specification	Yes	String	Resource specifications. The options are **modelarts.vm.cpu.2u**, **modelarts.vm.gpu.p4** (permission required), and **modelarts.vm.ai1.a310** (permission required). For the options that require a permission, create a service ticket on Huawei Cloud. Then, ModelArts O&M personnel will add the permissions for you.
instance_count	Yes	Integer	Number of instances deployed in a model. The maximum number of instances is 5. To use more instances, submit a service ticket.
envs	No	Map<String, String>	(Optional) Environment variable key-value pair required for running a model. By default, this parameter is left blank.

Table 11-7 configs parameters of **transformer**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID
specification	Yes	String	Resource flavor. Currently, **modelarts.vm.cpu.2u** and **modelarts.vm.gpu.p4** are available.
instance_count	Yes	Integer	Number of instances deployed in a model. The value range during the closed beta test is [1, 2].
envs	No	Map<String, String>	(Optional) Environment variable key-value pair required for running a model. By default, this parameter is left blank.
src_path	Yes	String	OBS path of the input data of a batch job
dest_path	Yes	String	OBS path of the output data of a batch job
req_uri	Yes	String	Inference API called in a batch task, that is, the RESTful API exposed in the model image. You must select an API URL from the **config.json** file of the model for inference. If a built-in inference image of ModelArts is used, the API is displayed as **/**.

Parameter	Mandatory	Type	Description

mapping_type	Yes	String	Mapping type of the input data. The value can be **file** or **csv**.
			• If you select **file**, each inference request corresponds to a file in the input data path. When this mode is used, **req_uri** of a model can have only one input parameter and the type of this parameter is **file**.
			• If you select **csv**, each inference request corresponds to a row of data in the CSV file. When this mode is used, the files in the input data path can only be in CSV format and **mapping_rule** needs to be configured to map the index of each parameter in the inference request body to the CSV file.
			The following shows how to create a batch service whose **mapping_type** is set to **file**:
			{
			"service_name": "batchservicetest",
			"description": "",
			"infer_type": "batch",
			"config": [{
			"model_id": "598b913a-af3e-41ba-a1b5-bf065320f1e2",
			"specification": "modelarts.vm.cpu.2u",
			"instance_count": 1,
			"src_path": "https://infers-data.obs.xxx.com/xgboosterdata/",
			"dest_path": "https://infers-data.obs.xxx.com/output/",
			"req_uri": "/",
			"mapping_type": "file"
			}]
			}
			The following shows how to create a batch service whose **mapping_type** is set to **csv**:
			{
			"service_name": "batchservicetest",
			"description": "",
			"infer_type": "batch",
			"config": [{
			"model_id": "598b913a-af3e-41ba-a1b5-bf065320f1e2",
			"specification": "modelarts.vm.cpu.2u",
			"instance_count": 1,
			"src_path": "https://infers-data.obs.xxx.com/xgboosterdata/",
			"dest_path": "https://infers-data.obs.xxx.com/output/",
			"req_uri": "/",
			"mapping_type": "csv",
			"mapping_rule": {
			"type": "object",
			"properties": {
			"data": {
			"type": "object",
			"properties": {
			"req_data": {
			"type": "array",
			"items": [{
			"type": "object",
			"properties": {

Parameter	Mandatory	Type	Description

			<pre>"input5": {
			"type": "number",
			"index": 0
			},
			"input4": {
			"type": "number",
			"index": 1
			},
			"input3": {
			"type": "number",
			"index": 2
			},
			"input2": {
			"type": "number",
			"index": 3
			},
			"input1": {
			"type": "number",
			"index": 4
			}
			}
			}]
			}
			}
			}
			}
			}]
			}</pre>

mapping_rule	No	Map	Mapping between input parameters and CSV data. This parameter is mandatory only when **mapping_type** is set to **csv**. The mapping rule is similar to the input parameter definition in the **config.json** model configuration file. You only need to configure the index parameters under each parameter of the string, number, integer, or boolean type, and the value of this parameter to the values of the index parameters in the CSV file to send an inference request. Use commas (,) to separate multiple pieces of CSV data. The values of the index parameters start from **0**. If the value of the index parameter is **-1**, ignore this parameter. For details, see the **sample code of deploying transformer**.
			The format of the inference request body described in **mapping_rule** is as follows:
			<pre>{
			"data": {
			"req_data": [{
			"input1": 1,
			"input2": 2,
			"input3": 3,
			"input4": 4,
			"input5": 5
			}]
			}
			}</pre>

Table 11-8 Parameters in the response to the request for deploying **predictor** and **transformer**

Parameter	Mandatory	Type	Description
predictor	Yes	Predictor object	Predictor object. Its attributes include all functions described in this chapter.

Table 11-9 schedule parameters

Parameter	Mandatory	Type	Description
op_type	Yes	String	Scheduling type. Currently, only the value **stop** is supported.
time_unit	Yes	String	Scheduling time unit. The options are as follows: • DAYS • HOURS • MINUTES
duration	Yes	Integer	Value that maps to the time unit. For example, if the task stops after two hours, set **time_unit** to **HOURS** and **duration** to **2**.

NOTE

- Example of deploying a real-time **predictor** instance in the handwritten digit recognition project implemented by MXNet:

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig, TransformerConfig

model_instance = Model(session, model_id = "you_model_id")
configs = []
config1 = ServiceConfig(model_id="you_model_id",
                        weight="100",
                        instance_count=1,
                        specification="modelarts.vm.cpu.2u",
                        envs={"input_data_name":"images",
                              "input_data_shape":"0,1,28,28",
                              "output_data_shape":"0,10"})
configs.append(config1)
predictor = model_instance.deploy_predictor(service_name="DigitRecognition", configs=configs)
```

- Example of deploying a **transformer** instance (batch processing) in a handwritten digit recognition project implemented by MXNet:

```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig, TransformerConfig

model_instance = Model(session, model_id = "your_model_id")
configs = []
config1 = TransformerConfig(model_id="your_model_id",
                            specification="modelarts.vm.cpu.2u",
                            instance_count=1,

envs={"input_data_name":"images","input_data_shape":"0,1,28,28","output_data_shape":"0,10"},
                            src_path="/w0403/testdigitrecognition/inferimages/",
                            dest_path="/w0403/testdigitrecognition/" ,
                            req_uri = "/",
                            mapping_type = "file")
configs.append(config1)
predictor = model_instance.deploy_transformer(service_name="DigitRecognition",
infer_type="batch", configs=configs)
```

11.4 Obtaining Details About a Service

You can use the API to obtain details about a service object.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Obtain details about a service object created in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
predictor_info = predictor_instance.get_service_info()
print(predictor_info)
```

- **Method 2**: Obtain details about a service based on the service object returned in **Obtaining Service Objects**.

```
from modelarts.session import Session
from modelarts.model import Predictor
```



```
session = Session()
predictor_object_list = Predictor.get_service_object_list(session)
predictor_instance = predictor_object_list[0]
predictor_info = predictor_instance.get_service_info()
print(predictor_info)
```

Parameters

Table 11-10 get_service_info response parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name
description	String	Service description
tenant	String	Tenant to whom a service belongs
project	String	Project to which a service belongs
owner	String	User to whom a service belongs
publish_at	Number	Latest service publishing time, in milliseconds calculated from 1970.1.1 0:0:0 UTC
infer_type	String	Inference mode. The value can be **real-time** or **batch**.
vpc_id	String	ID of the VPC to which a service instance belongs. This parameter is returned when the network configuration is customized.
subnet_network_id	String	ID of the subnet where a service instance resides. This parameter is returned when the network configuration is customized.
security_group_id	String	Security group to which a service instance belongs. This parameter is returned when the network configuration is customized.
status	String	Service status. The value can be **running**, **deploying**, **concerning**, **failed**, **stopped**, or **finished**.
error_msg	String	Error message. When **status** is **failed**, the deployment failure cause is returned.
config	**config** array corresponding to **infer_type**	**config** array corresponding to **infer_type** Service configurations (If a service is shared, only **model_id**, **model_name**, and **model_version** are returned.)
access_address	String	Access address of an inference request. This parameter is returned when **infer_type** is set to **real-time**.

Parameter	Type	Description
invocation_times	Number	Total number of service calls
failed_times	Number	Number of failed service calls
is_shared	Boolean	Whether a service is subscribed
shared_count	Number	Number of subscriptions
progress	Integer	Deployment progress. This parameter is returned when **status** is **deploying**.

Table 11-11 config parameters corresponding to **real-time**

Parameter	Type	Description
model_id	String	Model ID. You can obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.
model_name	String	Model name
model_version	String	Model version
source_type	String	Model source. This parameter is returned when a model is created by an ExeML project. The value is **auto**.
status	String	Running status of a model instance. Possible values are as follows: • **ready**: ready (All instances have been started.) • **concerning**: partially ready (Some instances are started but some are not.) • **notReady**: not ready (All instances are not started.)
weight	Integer	Traffic weight allocated to a model
specification	String	Resource flavor. The value can be **modelarts.vm.cpu.2u**, **modelarts.vm.gpu.p4**, or **modelarts.vm.ai1.a310**.
envs	Map<String, String>	Environment variable key-value pair required for running a model
instance_count	Integer	Number of instances deployed in a model
scaling	Boolean	Whether auto scaling is enabled

Table 11-12 config parameters corresponding to **batch**

Parameter	Type	Description
model_id	String	Model ID. You can obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.
model_name	String	Model name
model_version	String	Model version
specification	String	Resource flavor. The value can be **modelarts.vm.cpu.2u** or **modelarts.vm.gpu.p4**.
envs	Map<String, String>	Environment variable key-value pair required for running a model
instance_count	Integer	Number of instances deployed in a model
src_path	String	OBS path of the input data of a batch job
dest_path	String	OBS path of the output data of a batch job
req_uri	String	Inference path of a batch job
mapping_type	String	Mapping type of the input data. The value can be **file** or **csv**.
mapping_rule	Map	Mapping between input parameters and CSV data. This parameter is returned only when **mapping_type** is set to **csv**.

11.5 Testing an Inference Service

A real-time service supports files, images, and JSON data for test. Deploy a real-time service predictor for the inference test.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

Scenario: Perform an inference test using the predictor in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
predict_result = predictor_instance.predict(data=data_path, data_type=data_type)
print(predict_result)
```


Parameters

Table 11-13 Parameters

Parameter	Mandatory	Type	Description
data_type	Yes	String	The following types are supported: files, images, and JSON.

data	Yes	String	• For files or images, this parameter indicates the local path, for example:
			`data = "/home/ma-user/work/test.jpg"`
			• For JSON data, this parameter indicates the local path, for example:
			`data = "/home/ma-user/work/test.json"`
			It can also indicate a variable of the **dict** type, for example:
			`data = {`
			`"is_training": "False",`
			`"observations": [[1,2,3,4]],`
			`"default_policy/eps:0" : "0.0"`
			`}`

path	No	String	Internal inference path, which defaults to "/"

Table 11-14 predict response parameters

Parameter	Description
Response body	Output parameters and values. The platform only forwards the output parameters and values, but does not recognize them.

11.6 Obtaining Services

Obtain the service list of a user.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Scenario 1**: Obtain all services of a user.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_list = Predictor.get_service_list(session)
print(predictor_list)
```

- **Scenario 2**: Obtain the services of a user based on search criteria.

```
from modelarts.session import Session
from modelarts.model import Predictor
```



```
session = Session()
predictor_list = Predictor.get_service_list(session, service_name="digit", order="asc", offset="0",
infer_type="real-time")
print(predictor_list)
```

Parameters

Table 11-15 Query parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
service_id	No	String	Service ID. By default, the service ID is not filtered.
service_name	No	String	Service name. By default, the service name is not filtered.
infer_type	No	String	Inference mode. The value can be **real-time** or **batch**. By default, this parameter is left blank.
offset	No	Integer	Start page of the paging list. Default value: **0**
limit	No	Integer	Maximum number of records returned on each page. Default value: **1000**

Parameter	Mandatory	Type	Description
service_status	No	String	Service status. By default, the service status is not filtered. The service list can be queried based on the service status. Possible values are as follows: • **running**: The service is running properly and is being billed. • **deploying**: The service is being deployed or scheduling resources are being deployed. • **concerning**: An alarm is generated, indicating that the backend instance is abnormal and may be billed. For example, in the case of multiple instances, some instances are normal, but some are not. A normal instance is billed but is in the **concerning** status. • **failed**: The service fails to be deployed. For details about the failure cause, see the event and log. • **stopped**: The service has been stopped. • **finished**: This status is displayed only for the batch service, indicating that the service running is completed.
sort_by	No	String	Sorting mode. The value can be **publish_at** or **service_name**. Default value: **publish_at**
order	No	String	Sorting order. The value can be **asc** or **desc**, indicating the ascending or descending order. Default value: **desc**
model_id	No	String	Model ID. By default, the model ID is not filtered.

Table 11-16 get_service_list response parameters

Parameter	Type	Description
total_count	Integer	Total number of services that meet the search criteria when no paging is implemented
count	Integer	Number of services in the query result. If **offset** and **limit** are not set, the values of **count** and **total** are the same.

Parameter	Type	Description
services	**service** array	Collection of the queried services

Table 11-17 service parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name
description	String	Service description
tenant	String	Tenant to whom a service belongs
project	String	Project to which a service belongs
owner	String	User to whom a service belongs
publish_at	Number	Latest service publishing time, in milliseconds calculated from 1970.1.1 0:0:0 UTC
infer_type	String	Inference mode. The value can be **real-time** or **batch**.
status	String	Service status. The value can be **running**, **deploying**, **concerning**, **failed**, **stopped**, or **finished**.
progress	Integer	Deployment progress. This parameter is returned when **status** is **deploying**.
invocation_times	Number	Total number of service calls
failed_times	Number	Number of failed service calls
is_shared	Boolean	Whether a service is subscribed
shared_count	Number	Number of subscriptions

11.7 Obtaining Service Objects

You can use the API to obtain the service object list of a user.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Scenario 1**: Obtain all service objects of a user.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_list_object_resp = Predictor.get_service_object_list(session)
print(predictor_list_object_resp)
```

- **Scenario 2**: Obtain the service objects of a user based on search criteria.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_object_list = Predictor.get_service_object_list(session, service_name="digit", order="asc",
offset="0", infer_type="real-time")
print(predictor_object_list)
```

Parameters

- You can use the API to obtain the service list. The list size is equal to the number of services deployed by the user. Each element in the list is a predictor object. The object attributes are the same as those in service initialization.

  For example, in **service_list_resp = [service_instance1, service_instance2, service_instance3 ...]**, each **service_instance** in the list is a service API that can be called in the service management section.
- The service list can be queried based on the query parameters. **Table 11-18** describes the query parameters.
- When the model list is queried, details about the services are returned. See **Table 11-19** and **Table 11-20**.

Table 11-18 Query parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
is_show	No	Boolean	Whether to print service object information. Default value: **True**
service_id	No	String	Service ID. By default, the service ID is not filtered.
service_name	No	String	Service name. By default, the service name is not filtered.
infer_type	No	String	Inference mode. The value can be **real-time** or **batch**. By default, this parameter is left blank.
offset	No	Integer	Start page of the paging list. Default value: **0**
limit	No	Integer	Maximum number of records returned on each page. Default value: **1000**

Parameter	Mandatory	Type	Description
sort_by	No	String	Sorting mode. The value can be **publish_at** or **service_name**. Default value: **publish_at**
order	No	String	Sorting order. The value can be **asc** or **desc**, indicating the ascending or descending order. Default value: **desc**
model_id	No	String	Model ID. By default, the model ID is not filtered.

Table 11-19 get_service_list response parameters

Parameter	Type	Description
total_count	Integer	Total number of services that meet the search criteria when no paging is implemented
count	Integer	Number of services in the query result. If **offset** and **limit** are not set, the values of **count** and **total** are the same.
services	**service** array	Collection of the queried services

Table 11-20 service parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name
description	String	Service description
tenant	String	Tenant to whom a service belongs
project	String	Project to which a service belongs
owner	String	User to whom a service belongs
publish_at	Number	Latest service publishing time, in milliseconds calculated from 1970.1.1 0:0:0 UTC
infer_type	String	Inference mode. The value can be **real-time** or **batch**.
status	String	Service status. The value can be **running**, **deploying**, **concerning**, **failed**, **stopped**, or **finished**.

Parameter	Type	Description
progress	Integer	Deployment progress. This parameter is returned when **status** is **deploying**.
invocation_times	Number	Total number of service calls
failed_times	Number	Number of failed service calls
is_shared	Boolean	Whether a service is subscribed
shared_count	Number	Number of subscriptions

11.8 Updating Service Configurations

You can use the API to update the configurations of a service object.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Update the configurations of a service based on the service object created in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor
from modelarts.config.model_config import ServiceConfig

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
configs = [ServiceConfig(weight="100", instance_count=1,
specification="modelarts.vm.cpu.2u",model_id="your_model_id")]
service_config = predictor_instance.update_service_config(description="description",
                                                          status="running",
                                                          configs=configs)
```

- **Method 2**: Update the configurations of a service based on the service object returned in **Obtaining Service Objects**.

```
from modelarts.session import Session
from modelarts.model import Predictor
from modelarts.config.model_config import ServiceConfig

session = Session()
predictor_object_list = Predictor.get_service_object_list(session)
predictor_instance = predictor_object_list[0]
configs = [ServiceConfig(weight="100", instance_count=1,
specification="modelarts.vm.cpu.2u",model_id="your_model_id")]
predictor_config = predictor_instance.update_service_config(description="description",
                                                            status="running",
                                                            configs=configs)
```


Parameters

Table 11-21 Parameters for deploying **predictor**

Parameter	Mandatory	Type	Description
description	No	String	Service description, which contains a maximum of 100 characters. If this parameter is not set, the service description is not updated.
status	No	String	Service status. The value can be **running** or **stopped**. If this parameter is not set, the service status is not changed. **status** and **configs** cannot be modified at the same time. If both parameters exist, modify only the **status** parameter.
configs	No	**predictor configs** and **transformer configs**	Service configurations. If this parameter is not set, the service is not updated. For details about how to generate **configs**, see **Deploying a Real-Time Service**.

NOTE

The restrictions on updating service configurations are as follows:

- The specified **status** cannot be the same as the current service status.
- If the service status is **deploying**, **stopping**, or **deleting**, **status** cannot be set to **running** or **configs** is not allowed to configure.
- If the service status is **waiting**, **status** cannot be set to **running**.
- If the service status is **concerning**, **status** cannot be set to **running**.

Table 11-22 configs parameters of **predictor**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID. You can obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.

Parameter	Mandatory	Type	Description
weight	Yes	Integer	Weight of traffic allocated to a model. This parameter is mandatory only when **infer_type** is set to **real-time**. The sum of multiple weights must be equal to 100. If multiple model versions are configured in a real-time service and different traffic weights are set, ModelArts continuously accesses the prediction API of the service and forwards prediction requests to the model instances of the corresponding versions based on the weights.
specification	Yes	String	Resource flavor. Currently, **modelarts.vm.cpu.2u**, **modelarts.vm.gpu.p4** (you must apply for it), and **modelarts.vm.ai1.a310** (you must apply for it) are available. To use a flavor that requires permission, submit a service ticket on HUAWEI CLOUD and ModelArts O&M engineers will grant you the permission.
instance_count	Yes	Integer	Number of instances deployed in a model. The maximum number of instances is 5. To use more instances, submit a service ticket.
envs	No	Map<String, String>	(Optional) Environment variable key-value pair required for running a model. By default, this parameter is left blank.

Table 11-23 configs parameters of **transformer**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID. You can obtain the value by calling the API described in **Obtaining Models** or from the ModelArts management console.
specification	Yes	String	Resource flavor. Currently, **modelarts.vm.cpu.2u** and **modelarts.vm.gpu.p4** are available.
instance_count	Yes	Integer	Number of instances deployed in a model. The maximum number of instances is 5. To use more instances, submit a service ticket.
envs	No	Map<String, String>	(Optional) Environment variable key-value pair required for running a model. By default, this parameter is left blank.

Parameter	Mandatory	Type	Description
src_path	Yes	String	OBS path of the input data of a batch job
dest_path	Yes	String	OBS path of the output data of a batch job
req_uri	Yes	String	Inference API called in batch tasks. You must select an API URL from the **config.json** file of the model for inference.
mapping_type	Yes	String	Mapping type of the input data. The value can be **file** or **csv**. ● If you select **file**, each inference request corresponds to a file in the input data path. When this mode is used, **req_uri** of a model can have only one input parameter and the type of this parameter is **file**. ● If you select **csv**, each inference request corresponds to a row of data in the CSV file. When this mode is used, the files in the input data path can only be in CSV format and **mapping_rule** needs to be configured to map the index of each parameter in the inference request body to the CSV file.
mapping_rule	No	Map	Mapping between input parameters and CSV data. This parameter is mandatory only when **mapping_type** is set to **csv**. The mapping rule is similar to the definition of the input parameters in the **config.json** file. You only need to configure the index parameters under each parameter of the string, number, integer, or boolean type, and the value of this parameter to the values of the index parameters in the CSV file to send an inference request. Use commas (,) to separate multiple pieces of CSV data. The values of the index parameters start from **0**. If the value of the index parameter is **-1**, ignore this parameter.

Table 11-24 update_service_config response parameters

Parameter	Mandatory	Type	Description
error_code	Yes	String	Error code when the API call fails. This parameter is not included when the API call succeeds.
error_msg	Yes	String	Error message when the API call fails. This parameter is not included when the API call succeeds.

11.9 Obtaining Service Monitoring Information

You can use the API to obtain the monitoring information about a service.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Obtain the monitoring information of a service based on the service object created in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
predictor_monitor = predictor_instance.get_service_monitor()
print(predictor_monitor)
```

- **Method 2**: Obtain the monitoring information of a service based on the service object returned in **Obtaining Service Objects**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_object_list = Predictor.get_service_object_list(session)
predictor_instance = predictor_object_list[0]
predictor_monitor = predictor_instance.get_service_monitor()
print(predictor_monitor)
```

Parameters

Table 11-25 get_service_monitor response parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name

Parameter	Type	Description
monitors	**monitor** array corresponding to **infer_type** of a service	Monitoring details

Table 11-26 monitor parameters corresponding to **real-time**

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version
invocation_times	Number	Total number of model instance calls
failed_times	Number	Number of failed model instance calls
cpu_core_usage	Float	Number of used CPUs
cpu_core_total	Float	Total number of CPUs
cpu_memory_usage	Integer	Used memory, in MB
cpu_memory_total	Integer	Total memory, in MB
gpu_usage	Float	Number of used GPUs
gpu_total	Float	Total number of GPUs

11.10 Obtaining Service Logs

You can use the API to obtain the logs of a service object.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Obtain the logs of a service based on the service object created in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor
```



```
session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
predictor_log = predictor_instance.get_service_logs()
print(predictor_log)
```

- **Method 2**: Obtain the logs of a service based on the service object returned in **Obtaining Service Objects**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_object_list = Predictor.get_service_object_list(session)
predictor_instance = predictor_object_list[0]
predictor_log = predictor_instance.get_service_logs()
print(predictor_log)
```

Parameters

Table 11-27 get_service_logs response parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name
logs	**log** array	Service update logs

Table 11-28 log parameters

Parameter	Type	Description
update_time	Long	Time when a service is updated, in milliseconds calculated from 1970.1.1 0:0:0 UTC
result	String	Update result. The value can be **SUCCESS**, **FAIL**, or **RUNNING**.
config	**config** array	Updated service configurations. This parameter is returned when **infer_type** is set to **real-time**.

Table 11-29 config parameters

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version
weight	Integer	Traffic weight allocated to a model

Parameter	Type	Description
specification	String	Resource flavor
instance_count	Integer	Number of instances deployed in a model
envs	Map<String, String>	Environment variable key-value pair required for running a model

Table 11-30 result parameters

Parameter	Type	Description
node_name	String	Name of an edge node
operation	String	Operation type. The value can be **deploy** or **delete**.
result	Boolean	Operation result. **true** indicates a successful operation, and **false** indicates a failed operation.

11.11 Delete a Service

You can delete a service in either of the following ways:

- Delete the service created in **Deploying a Real-Time Service**.
- Delete the service object returned in **Obtaining Service Objects**.

Sample Code

In ModelArts notebook, you do not need to enter authentication parameters for session authentication. For details about session authentication of other development environments, see **Session Authentication**.

- **Method 1**: Delete a service based on the service object created in **Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_instance = Predictor(session, service_id="your_service_id")
predictor_instance.delete_service()
```

- **Method 2**: Delete a service based on the service object returned in **Obtaining Service Objects**.

```
from modelarts.session import Session
from modelarts.model import Predictor

session = Session()
predictor_object_list = Predictor.get_service_object_list(session)
```



```
predictor_instance = predictor_object_list[0]
predictor_instance.delete_service()
```


12 Change History

Released On	Description
2023-11-17	Optimized **(Optional) Session Authentication**.
2023-09-27	Optimized **SDK Overview**. Added **Getting Started**.
2023-02-23	Added **Using the SDK to Debug a Single-Node Training Job** and **Using the SDK to Debug a Multi-Node Distributed Training Job**.
2022-11-24	Optimized **SDK Overview** and **Creating a Training Job**.
2022-11-01	Added **Transferring Files (Recommended)**.
2022-10-28	Optimized **Importing a Model** and **Deploying a Real-Time Service**.
2022-03-29	Added the training management (recommended) SDK. **Creating a Training Job**
2021-11-18	Optimized **(Optional) Installing the ModelArts SDK Locally**.
2021-01-15	Added the SDK reference. • **Debugging a Training Job** • **Debugging a Model**
2020-04-10	Added the following sections: • **OBS Management**: Added OBS-related operation guide.

Released On	Description
2019-08-20	This is the second official release. Added the following sections: • **Training Management**: Added the local training function. • **Model Management**: Added the function of obtaining the model object list. • **Service Management**: Added local service deployment and local inference. Modified the following sections: • **Session Authentication**: Optimized session authentication.
2019-05-09	This is the first official release.

