ModelArts

SDK Reference

Issue	01
Date	2022-09-23

HUAWEI TECHNOLOGIES CO., LTD.

Huawei Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	https://www.huawei.com
Email:	support@huawei.com

Contents

1 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
3.1 (Optional) Installing ModelArts SDKs Locally **Viewing Project ID** **3.3 Setting Domain Names and IP Addresses** **3.4 Configuring a Service Endpoint**	How to make preparations for secondary development using ModelArts SDK
Session Authentication	How to authenticate resources and initialize ModelArts SDK Client and OBS Client
5.1 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: **Training Management (Recommended)** **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, including Python 2.7, Python 3.6, and Python 3.7.

Application Scenarios

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

NOTICE

ModelArts SDKs cannot be used in training and inference environments. For example, SDK authentication cannot be used in inference scripts.

- 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 in 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 configured and used in local Windows or Linux. For details, see **3.1 (Optional) Installing ModelArts SDKs Locally**.

3 Preparations

3.1 (Optional) Installing ModelArts SDKs Locally

ModelArts SDKs can be configured and used in local Windows or Linux. To do so, perform the following operations:

1. Configure the runtime environment.
   - Download Python at **the Python official website** and install it. (Python 2.7 or later is recommended.) If the general package management tool pip is not installed during Python installation, install pip by following the operations provide at **the pip official website** after you install Python.
   - Configure the pip source.
2. **Downloading ModelArts SDKs**
3. **Installing ModelArts SDKs**

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.

Downloading ModelArts SDKs

The ModelArts SDK software package is stored in the public OBS bucket of the target site. The download address is as follows:

https://obs-sdk.obs.{*Region ID*}.{*Site domain name*}.com/modelarts-latest-py2.py3-none-any.whl

Contact the administrator to obtain *Region ID* and *Site domain name*.

For example, https://obs-sdk.obs.cn-central-231.xckpjs.com/modelarts-latest-py2.py3-none-any.whl

Installing ModelArts SDKs

1. Run **python --version** in the local environment to check whether Python has been installed.
```
C:\Users\xxx>python --version
Python *.*.*
```
2. Run **pip --version** to check whether the general package management tool pip is available.
```
C:\Users\xxx>pip --version
pip **.*.* from c:\users\xxx\appdata\local\programs\python\python**\lib\site-packages\pip (python *.*)
```
3. Install SDK.

   pip install *{Path for storing 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, 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.

3.2 Viewing the Project ID

When calling APIs, specify the project ID in certain URLs. Therefore, the project ID must be obtained. To obtain a project ID, perform the following operations:

1. Log in to the console.
2. In the upper right corner, click your account avatar icon and choose **My Settings** from the drop-down list.
3. On the **My Settings** page, go to the **Project List** tab page, which is displayed by default. View the project ID and name in the project list.

3.3 Setting Domain Names and IP Addresses

If a computer is used for configuration, set the domain names and IP addresses of IAM, OBS, and ModelArts in **C:\Windows\System32\drivers\etc\hosts**. Contact technical support for the domain names and IP addresses. The following shows an example configuration:

```
xxx.xxx.xxx.xxx      modelarts.xxx.xxx.com
xxx.xxx.xxx.xxx      iam.xxx.xxx.com
xxx.xxx.xxx.xxx      obs.xxxx.xxx.com
xxx.xxx.xxx.xxx      swr.xxx.xxx.com
```

3.4 Configuring a Service Endpoint

Using the SDK locally requires to call IAM, OBS, and ModelArts. Therefore, the endpoints of these services are required. To obtain the endpoints, configure as follows:

```
from modelarts.session import Session
Session.set_endpoint(iam_endpoint="***", obs_endpoint="***", modelarts_endpoint="***", region_name="***")
```

Table 3-1 Parameters

Parameter	Description
iam_endpoint	IAM endpoint
obs_endpoint	OBS endpoint
modelarts_endpoint	ModelArts endpoint
region_name	Region name

4 Session Authentication

4.1 Overview of Session Authentication

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

ModelArts notebook development environments do not require session authentication and can be directly used. The sample code is as follows:

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

When using the ModelArts SDK in other development environments, select one of the following authentication methods for session authentication:

- **4.2 Authentication Using the Username and Password**: Available for **OBS Management**, **Training Management**, **Model Management**, and **Service Management**.
- **4.3 AK/SK-based Authentication**: Available for **OBS Management**, **Training Management**, **Model Management**, and **Service Management**.

4.2 Authentication Using the Username and Password

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

Sample Code

- Authentication using an account

  Set **username** to your account name.

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

4.3 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 as follows:

- For details about how to obtain **access_key** and **secret_key**, see "Obtaining an Access Key".
- **project_id** indicates the project ID. For details about how to obtain the value, see **3.2 Viewing the Project ID**.
- **region_name** indicates the region ID. Obtain the region ID from the administrator.

5 OBS Management

5.1 Overview of OBS Management

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

- **5.2 Uploading a File to OBS**
- **5.3 Uploading a Folder to OBS**
- **5.4 Downloading a File from OBS**
- **5.5 Downloading a Folder from OBS**

5.2 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 **4 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 5-1 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 5-2 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.

5.3 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 **4 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 5-3 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 5-4 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.

5.4 Downloading a File from OBS

NOTE

If the size of a file in a folder exceeds 5 GB, the file cannot be downloaded in this mode.

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 **4 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 5-5 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 5-6 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.

5.5 Downloading a Folder from OBS

NOTE

If the size of a file in a folder exceeds 5 GB, the file cannot be downloaded in this mode. However, other files whose size is less than 5 GB in the folder can be downloaded.

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 **4 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/")
```

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

Parameters

Table 5-7 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 5-8 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 Training Management

6.1 Training Jobs

6.1.1 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**.

- 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"}]
parameters = [{"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"},
                      job_description='This is a 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"}]
parameters = [{"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"},
                      job_description='This is a 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"}]
parameters = [{"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"},
                      job_description='This is a image net train job')
job_instance = estimator.fit(inputs=[InputData(obs_path="obs://bucket_name/input/", 
name="data_url")],
                             job_name="job_name_3")
```

Parameters

Table 6-1 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 6-2**.
outputs	No	Array of **OutputData** objects	Training output path. For details, see **Table 6-3**.
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 **6.2.1 Obtaining Resource Flavors**.
train_instance_count	Yes	Int	Number of compute nodes in a training job
framework_type	No	String	Engine type selected for a training job. For details, see **6.2.2 Obtaining Engine Types**.
framework_version	No	String	Engine version selected for a training job. For details, see **6.2.2 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/**
job_description	No	String	Description of a training job

Parameter	Mandatory	Type	Description

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
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 6-2 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 6-3 Parameters for initializing **OutputData**

Parameter	Mandatory	Type	Description
obs_path	Yes	String	OBS path to which data is exported

Parameter	Mandatory	Type	Description
name	Yes	String	Name of the data output channel

Table 6-4 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.
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 6-5 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 6-6 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 6-7 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" in *ModelArts API Reference*. 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.

6.1.2 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 6-8 get_job_list request parameters

Parameter	Mandatory	Type	Description
session	Yes	Object	Session object. For details about the initialization method, see **Session Authentication**.
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 6-9 filters

Parameter	Mandatory	Type	Description
key	No	String	Key of the grouping condition.

Parameter	Mandatory	Type	Description
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 6-10 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 6-11 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
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.
tasks	Array of **TaskResponse** objects	Tasks of a heterogeneous training job.
spec	**spec** object	Specifications of a training job.

Table 6-12 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 6-13 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 6-14 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 6-15 JobAlgorithmResponse

Parameter	Type	Description
id	String	Algorithm ID Options: ● **id**: Only the algorithm ID is used. ● **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
name	String	Algorithm name
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used together 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 together 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.

Parameter	Type	Description
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 6-16 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 6-17 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 6-18 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 6-19 policies

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

Table 6-20 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 6-21 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.

Parameter	Type	Description
regex	String	Regular expression of a metric

Table 6-22 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 6-23 algo_configs

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

Table 6-24 AutoSearchAlgoConfigParameter

Parameter	Type	Description
key	String	Parameter key
value	String	Parameter value

Parameter	Type	Description
type	String	Parameter type

Table 6-25 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 6-26 InputDataInfo

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

Table 6-27 obs

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

Table 6-28 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location.

Table 6-29 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 6-30 remote

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

Table 6-31 obs

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

Table 6-32 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**

Parameter	Type	Description
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 6-33 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 6-34 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 6-35 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 6-36 remote

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

Table 6-37 obs

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

Table 6-38 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 6-39 remote

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

Table 6-40 obs

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

Table 6-41 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 6-42 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

Parameter	Type	Description
flavor_type	String	Resource flavor type. Options: ● **CPU** ● **GPU**
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 6-43 billing

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

Table 6-44 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
memory	**memory** object	Memory information

Table 6-45 cpu

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

Table 6-46 gpu

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

Table 6-47 memory

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

Table 6-48 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 6-49 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 6-50 flavor_detail

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

Table 6-51 billing

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

Table 6-52 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
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 6-53 cpu

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

Table 6-54 gpu

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

Table 6-55 memory

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

Table 6-56 disk

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

Table 6-57 volumes

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

Table 6-58 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 6-59 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 6-60 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" in *ModelArts API Reference*. 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 6-61 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" in *ModelArts API Reference*. 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.

6.1.3 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 6-62 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 6-63 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.
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.

Parameter	Type	Description
spec	**spec** object	Specifications of a training job.

Table 6-64 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 6-65 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**

Parameter	Type	Description
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 6-66 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 6-67 JobAlgorithmResponse

Parameter	Type	Description
id	String	Algorithm ID Options: ● **id**: Only the algorithm ID is used. ● **code_dir** and **boot_file**: The code directory and boot file of the training job are used.
name	String	Algorithm name
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used together with **boot_file**. Leave this parameter blank if **id**, or **subscription_id** and **item_version_id** are specified.

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

Table 6-68 Parameter

Parameter	Type	Description
name	String	Parameter name
value	String	Parameter value

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

Table 6-69 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 6-70 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 6-71 policies

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

Table 6-72 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 6-73 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 6-74 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

Parameter	Type	Description
discrete_values	Array of strings	Discrete hyperparameter values

Table 6-75 algo_configs

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

Table 6-76 AutoSearchAlgoConfigParameter

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

Table 6-77 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 6-78 InputDataInfo

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

Table 6-79 obs

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

Table 6-80 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location

Table 6-81 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 6-82 remote

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

Table 6-83 obs

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

Table 6-84 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 6-85 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 6-86 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 6-87 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 6-88 remote

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

Table 6-89 obs

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

Table 6-90 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 6-91 remote

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

Table 6-92 obs

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

Table 6-93 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 6-94 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**
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 6-95 billing

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

Table 6-96 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
memory	**memory** object	Memory information

Table 6-97 cpu

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

Table 6-98 gpu

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

Table 6-99 memory

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

Table 6-100 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 6-101 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 6-102 flavor_detail

Parameter	Type	Description
flavor_type	String	Resource flavor type. Options: ● **CPU** ● **GPU**

Parameter	Type	Description
billing	**billing** object	Billing information of a resource flavor
flavor_info	**flavor_info** object	Resource flavor details

Table 6-103 billing

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

Table 6-104 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
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 6-105 cpu

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

Table 6-106 gpu

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

Table 6-107 memory

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

Table 6-108 disk

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

Table 6-109 volumes

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

Table 6-110 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 6-111 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 6-112 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" in *ModelArts API Reference*. 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.

6.1.4 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 6-113 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 6-114 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 6-115 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" in *ModelArts API Reference*. 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.

6.1.5 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 6-116 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 6-117 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" in *ModelArts API Reference*. 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.

6.1.6 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 6-118 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 6-119 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.

Parameter	Type	Description
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. ● **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 6-120 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 6-121 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 6-122 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 6-123 JobAlgorithmResponse

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

Parameter	Type	Description
code_dir	String	Code directory of a training job, for example, **/usr/app/**. This parameter must be used together 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 together 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.
environments	Map<String,String>	Environment variables of a training job in the format of "key":"value". Leave this parameter blank.

Table 6-124 Parameter

Parameter	Type	Description
name	String	Parameter name

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

Table 6-125 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 6-126 i18n_description

Parameter	Type	Description
language	String	Internationalization language
description	String	Description

Table 6-127 policies

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

Table 6-128 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 6-129 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 6-130 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 6-131 algo_configs

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

Table 6-132 AutoSearchAlgoConfigParameter

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

Table 6-133 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 6-134 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 6-135 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 6-136 obs

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

Table 6-137 remote_constraint

Parameter	Type	Description
data_type	String	Data input type, including the data storage location

Table 6-138 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 6-139 remote

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

Table 6-140 obs

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

Table 6-141 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 6-142 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 6-143 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 6-144 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 6-145 remote

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

Table 6-146 obs

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

Table 6-147 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 6-148 remote

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

Table 6-149 obs

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

Table 6-150 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 6-151 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
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 6-152 billing

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

Table 6-153 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
memory	**memory** object	Memory information

Table 6-154 cpu

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

Table 6-155 gpu

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

Table 6-156 memory

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

Table 6-157 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 6-158 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 6-159 flavor_detail

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

Table 6-160 billing

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

Table 6-161 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
memory	**memory** object	Memory information
disk	**disk** object	Disk information

Table 6-162 cpu

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

Table 6-163 gpu

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

Table 6-164 memory

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

Table 6-165 disk

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

Table 6-166 volumes

Paramete r	Type	Description
nfs	**nfs** object	Disks attached in NFS mode

Table 6-167 nfs

Paramete r	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 6-168 log_export_path

Paramete r	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 6-169 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" in *ModelArts API Reference*. 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.

6.1.7 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 6-170 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 6-171 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 6-172 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 6-173 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" in *ModelArts API Reference*. 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.

6.1.8 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 6-174 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 6-175 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 6-176 Response parameters

Parameter	Type	Description
metrics	Array of objects	Runtime metrics

Table 6-177 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 6-178 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" in *ModelArts API Reference*. 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.

6.2 APIs for Resources and Engine Specifications

6.2.1 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 6-179 get_train_instance_types parameters

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

Table 6-180 Successful response parameters

Type	Description
List	List of resource flavors

Table 6-181 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" in *ModelArts API Reference*. 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.

6.2.2 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 6-182 **get_train_instance_types** parameters

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

Table 6-183 Successful response parameters of **get_framework_list**

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

Table 6-184 **framework_list** parameters

Parameter	Type	Description
framework_type	String	Engine type
framework_version	String	Engine version

Table 6-185 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" in *ModelArts API Reference*. 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.

7 Model Management

7.1 Importing a Model

The model import function covers the following aspects:

- 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 **7.4 Querying the Details About a Model**.

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="input your model id")
```
- Method 2: Create a model.
```
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)
```

– Definition formats of **input_params** and **output_params** parameter groups used in method 2

  The SDK provides the definition of **input_params** and **output_params** parameter groups. The types of **input_params** and **output_params** are list, and those of the tuple objects in the list are Params.

  The following uses **input_params** as an example:

```
input_params = []                                    # The type of input_params is list. Multiple
objects of the Params type can be stored.
input_params1 = Params(
                  url='url',                    # URL
                  param_name='param_name',             # Parameter name
                  param_type='param_type',             # Parameter type
                  min='min',
                  max='max',
                  param_desc='param_desc')
input_params.append(input_params)
```

– Definition formats of the **dependencies** parameter group used in method 2

  The SDK provides the definition of the **dependencies** parameter group. The type of **dependencies** is list, 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 about the definition format, see the definition of packages.
dependencies.append(dependency1)
```

– Definition formats of the **package** parameter group used in method 2

  The SDK provides the definition of the **packages** parameter group. The type of **packages** is list, 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="restraint")
packages.append(package1)
```

Parameters

Table 7-1 Parameters for initializing a model

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

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID

Table 7-2 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. The value is a string of 1 to 64 characters consisting of only letters, digits, underscores (_), and hyphens (-). It must start with a letter. 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.
source_location_type	No	String	Model location type. The options are as follows: ● **OBS_SOURCE**: OBS path. (Default value) ● **LOCAL_SOURCE**: local path.
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

Parameter	Mandatory	Type	Description
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. If the model is deployed by 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**.
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 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.

Parameter	Mandatory	Type	Description
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 7-3 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.

Parameter	Mandatory	Type	Description
param_desc	No	String	Parameter description, which contains a maximum of 100 characters. By default, this parameter is left blank.

Table 7-4 dependency parameters

Parameter	Mandatory	Type	Description
installer	Yes	String	Installation mode. Only **pip** is supported.
packages	Yes	**package** array	Collection of dependency packages

Table 7-5 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 7-6 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")
```

7.2 Querying the Model List

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: Query all models of a user.

```
from modelarts.session import Session
from modelarts.model import Model
session = Session()
model_list = Model.get_model_list(session)
print(model_list)
```

- Scenario 2: Query a model based on the 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")
print(model_list)
```

Parameters

Table 7-7 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 query 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**

Parameter	Mandatory	Type	Description
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 7-8 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 7-9 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

Parameter	Type	Description
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.

7.3 Querying the Model List

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: Query 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: Query a model object based on the 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 query 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 **7.4 Querying the 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 queried based on the query parameters. **Table 7-10** describes the query parameters.
- When the model list is queried, details about the models are returned. See **Table 7-11** and **Table 7-12**.
- Currently, a maximum of 150 model objects can be obtained.

Table 7-10 Query parameters

Parameter	Mandatory	Type	Description
model_name	No	String	Model name. Fuzzy match is supported.

Parameter	Mandatory	Type	Description
model_version	No	String	Model version
model_status	No	String	Model status. The value can be **publishing**, **published**, or **failed**. You can query 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 7-11 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 7-12 model parameters

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version

Parameter	Type	Description
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**.

7.4 Querying the Details About a Model

You can use the API to query 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: Query the details about the model created in **7.1 Importing a Model**.

```
from modelarts.session import Session
from modelarts.model import Model
session = Session()
model_instance = Model(session, model_id="input your model_id")
model_info = model_instance.get_model_info()
print(model_info)
```

- Method 2: Query the details about a model object returned in **7.3 Querying the Model List**.

```
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 7-13 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
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 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.

Parameter	Type	Description
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 7-14 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 7-15 dependency parameters

Parameter	Type	Description
installer	String	Installer
packages	**package** array	Collection of dependency packages

Table 7-16 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 7-17 metric parameters

Parameter	Mandatory	Type	Description
f1	Yes	Double	Mean
recall	Yes	Double	Recall
precision	Yes	Double	Precision
accuracy	Yes	Double	Accuracy

7.5 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 created in **7.1 Importing a Model**.
```
from modelarts.session import Session
from modelarts.model import Model
session = Session()
model_instance = Model(session, model_id="input your model_id")
model_instance.delete_model()
```
- Method 2: Delete the model object returned in **7.3 Querying the Model List**.
```
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()
```

8 Service Management

8.1 Service Management Overview

Service management indicates deploying a model that has been successfully created as a real-time. This feature provides functions such as real-time 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**.

8.2 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="input your service_id")
```
- Method 2: Deploy a real-time service predictor.
```
from modelarts.session import Session
from modelarts.model import Model
from modelarts.config.model_config import ServiceConfig,TransformerConfig
session = Session()
model_instance = Model(session, model_id="input your model_id")
predictor_instance = model_instance.deploy_predictor(
                                  service_name="input 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 parameters. For
details, see the format description of the configs parameter.
```
  The **model_id** parameter specifies the model to be deployed as a real-time service. You can obtain **model_id** by calling the API described in **7.2 Querying the Model List** or from the ModelArts management console.
- Method 3: Deploy a batch service transformer.
```
transformer = model_instance.deploy_transformer(
                                  service_name="input 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 the format description of the configs parameter.
```
- Definition formats of the **configs** parameter group used for deploying a real-time service predictor and a batch service transformer
  – Deploying a real-time service predictor:

    The SDK provides the definition of the **configs** parameter. The type of **configs** is list, and those of the tuple objects in the list are ServiceConfig. The code is as follows:
```
configs = []
service_config1 = ServiceConfig(
                    model_id="model_id1",
                    weight="70",
                    specification="specification",
                    instance_count=2,
                    envs=envs)
service_config2 = ServiceConfig(
                    model_id="model_id2",
                    weight="30",
                    specification="specification",
                    instance_count=2,
                    envs=envs)
configs.append(service_config1, service_config2)
```

– Deploying a batch service transformer:

  The SDK provides the definition of the **configs** parameter. The type of **configs** is list, and those of the tuple objects in the list are TransformerConfig. The code is as follows:

```
configs = []
transformer_config1 = TransformerConfig(
                                model_id="model_id",
                                specification="specification",
                                instance_count=2,
                                src_path="src_path",
                                dest_path="dest_path",
                                req_uri="req_uri",
                                mapping_type="mapping_type",
                                mapping_rule="mapping_rule",
                                envs=envs)
configs.append(transformer_config1)
```

Parameters

Table 8-1 Parameter description

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 8-2 Parameters for deploying the predictor and transformer

Parameter	Mandatory	Type	Description
service_name	No	String	Service name, which consists of 1 to 64 characters. It must start with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.
description	No	String	Service description, which contains a maximum of 100 characters. By default, this parameter is left blank.

Parameter	Mandatory	Type	Description
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.
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. If you need 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**.

Parameter	Mandatory	Type	Description
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 8-6**.

Table 8-3 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 **7.2 Querying the Model List** 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.
			`{`
			`"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`
			`}`
			`]`
			`}`

specification	Yes	String	Resource specifications.
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 8-4 configs parameters of **transformer**

Parameter	Mandatory	Type	Description
model_id	Yes	String	Model ID

Parameter	Mandatory	Type	Description
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": {`
			`"input5": {`
			`"type": "number",`
			`"index": 0`

Parameter	Mandatory	Type	Description
			```
			},
			"input4": {
			"type": "number",
			"index": 1
			},
			"input3": {
			"type": "number",
			"index": 2
			},
			"input2": {
			"type": "number",
			"index": 3
			},
			"input1": {
			"type": "number",
			"index": 4
			}
			}
			}]
			}
			}
			}
			}
			}
			}]
			}
			```

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:
			```
			{
			"data": {
			"req_data": [{
			"input1": 1,
			"input2": 2,
			"input3": 3,
			"input4": 4,
			"input5": 5
			}]
			}
			}
			```

Table 8-5 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 8-6 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 = "input you model id")
configs = []
config1 = ServiceConfig(model_id="input 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 = "input your model id")
configs = []
config1 = TransformerConfig(model_id="input 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)
```

8.3 Querying the Details of a Service

You can use the API to query 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: Query details about the service created in **8.2 Deploying a Real-Time Service**.
```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_info = predictor_instance.get_service_info()
print(predictor_info)
```
- Method 2: Query details about the service object returned in **8.5 Querying the List of 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 8-7 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 8-8 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 **7.2 Querying the Model List** or from the ModelArts management console.
model_name	String	Model name
model_version	String	Model version
source_type	String	Model source.
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 8-9 config parameters corresponding to **batch**

Parameter	Type	Description
model_id	String	Model ID. You can obtain the value by calling the API described in **7.2 Querying the Model List** 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**.

8.4 Querying a Service List

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 a service based on the 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 8-10 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**
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.

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 8-11 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 8-12 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

8.5 Querying the List of 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: Query 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: Query a service object based on the 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 query 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 8-13** describes the query parameters.
- When the model list is queried, details about the services are returned. See **Table 8-14** and **Table 8-15**.

Table 8-13 Query parameter description

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**
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 8-14 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 8-15 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

8.6 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 the service created in **8.2 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="input your service_id")
```

```
configs = [ServiceConfig(weight="100", instance_count=1, specification="modelarts.vm.cpu.
2u",model_id="input your model_id")]
service_config = predictor_instance.update_service_config(description="description",
                                                          status="running",
                                                          configs=configs)
```

- Method 2: Update the configurations of the service object returned in **8.5 Querying the List of 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="input your model_id")]
predictor_config = predictor_instance.update_service_config(description="description",
                                                            status="running",
                                                            configs=configs)
```

Parameters

Table 8-16 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 **8.2 Deploying a Real-Time Service**.

Table 8-17 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 **7.2 Querying the Model List** 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.
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 8-18 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 **7.2 Querying the Model List** 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.
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

Parameter	Mandatory	Type	Description
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 8-19 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.

8.7 Querying Service Monitoring Details

You can use the API to query 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: Query the monitoring information about the service created in **8.2 Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_monitor = predictor_instance.get_service_monitor()
print(predictor_monitor)
```

- Method 2: Query the monitoring information about the service object returned in **8.5 Querying the List of 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 8-20 get_service_monitor response parameters

Parameter	Type	Description
service_id	String	Service ID
service_name	String	Service name
monitors	**monitor** array corresponding to **infer_type** of a service	Monitoring details

Table 8-21 monitor parameters corresponding to **real-time**

Parameter	Type	Description
model_id	String	Model ID
model_name	String	Model name
model_version	String	Model version

Parameter	Type	Description
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

8.8 Querying Service Logs

You can use the API to query 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: Query service logs based on the service created in **8.2 Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_log = predictor_instance.get_service_logs()
print(predictor_log)
```

- Method 2: Query service logs based on the service object returned in **8.5 Querying the List of 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 8-22 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 8-23 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 8-24 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
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 8-25 result parameters

Parameter	Type	Description
node_name	String	Name of an edge node

Parameter	Type	Description
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.

8.9 Delete a Service

You can delete a service in either of the following ways:

- Delete the service created in **8.2 Deploying a Real-Time Service**.
- Delete the service object returned in **8.5 Querying the List of 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 the service created in **8.2 Deploying a Real-Time Service**.

```
from modelarts.session import Session
from modelarts.model import Predictor
session = Session()
predictor_instance = Predictor(session, service_id="input your service_id")
predictor_instance.delete_service()
```

- Method 2: Delete the service object returned in **8.5 Querying the List of 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()
```