

ModelArts Studio (MaaS)

User Guide

Issue	01
Date	2025-09-17



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

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

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 ModelArts Studio (MaaS) Usage.....	1
2 Configuring ModelArts Studio (MaaS) Access Authorization.....	4
2.1 Creating an IAM User and Granting ModelArts Studio (MaaS) Permissions.....	4
2.2 Configuring ModelArts Agency Authorization for Using ModelArts Studio (MaaS).....	8
2.3 Configuring the Missing ModelArts Studio (MaaS) Permissions.....	16
3 Preparing ModelArts Studio (MaaS) Resources.....	23
4 ModelArts Studio (MaaS) Real-Time Inference Services.....	24
4.1 Viewing a Built-in Model in ModelArts Studio (MaaS).....	24
4.2 Deploying a Model Service in ModelArts Studio (MaaS).....	28
4.3 Managing My Services in ModelArts Studio (MaaS).....	32
4.3.1 Starting, Stopping, or Deleting a Service in ModelArts Studio (MaaS).....	32
4.3.2 Scaling Model Service Instances in ModelArts Studio (MaaS).....	33
4.3.3 Modifying the QPS of a Model Service in ModelArts Studio (MaaS).....	35
4.3.4 Upgrading a Model Service in ModelArts Studio (MaaS).....	35
4.4 Calling a Model Service in ModelArts Studio (MaaS).....	36
4.5 ModelArts Studio (MaaS) API Call Specifications.....	46
4.5.1 Sending a Chat Request (Chat/POST).....	46
4.5.2 Obtaining the Model List (Models/GET).....	55
4.5.3 Error Codes.....	57
4.6 Creating a Multi-Turn Dialogue in ModelArts Studio (MaaS).....	60
5 ModelArts Studio (MaaS) Management and Statistics.....	61
5.1 Managing API Keys in ModelArts Studio (MaaS).....	61

1 ModelArts Studio (MaaS) Usage

MaaS offers a complete toolchain for creating foundation models using Ascend compute. It comes with ready-to-use popular open-source foundation models and helps with tasks like data production, fine-tuning, prompt engineering, and application integration. It is designed for users who need to develop production-ready models using a MaaS platform.

Context

AI models now play a vital role in driving enterprise digital transformation due to their advanced abilities in understanding language, generating content, and making decisions. Many businesses aim to improve operations using foundation models for tasks like customer support, data analytics, and automatic reporting. However, they encounter three main hurdles when they train or fine-tune foundation models: expensive compute, complicated technical demands, and integrating these systems into existing workflows. Since most companies lack skilled AI teams, building and refining models from scratch proves challenging. This often leads to slow deployments and project failures.

To tackle these challenges, MaaS offers a one-stop solution:

- **Toolchain:** Offers a visual training platform that simplifies model customization for businesses, requiring minimal AI expertise.
- **Resource sharing:** Cloud compute enables resource sharing by allowing companies to share compute and reuse pretrained models, cutting down on duplicate expenses and lowering overall compute costs.
- **Scenario adaptation:** Preset model templates tailored for specific industries help speed up the deployment of enterprise AI applications.

Use Cases

This section describes MaaS use cases:

- **Integration of leading open-source models**
MaaS integrates leading open-source models, including DeepSeek. All models are fully adapted and optimized for AI Cloud Service, resulting in improved accuracy and performance. You no longer need to build models from scratch; instead, you can simply choose suitable pre-trained models for direct application, reducing the workloads of model integration.

- **Easy access to resources, pay-per-use billing, scalability, fault recovery, and resumable training**

Enterprises must balance the model's performance with its costs and real-world feasibility when integrating AI models into their systems.

MaaS enables flexible model development, leveraging the Ascend Cloud compute backbone to streamline model usage in customer applications.

It allows you to scale resources on demand, with pay-per-use billing. This minimizes wasted resources and makes AI more accessible by lowering initial investment.

The architecture prioritizes high availability by duplicating data centers, so you can rest assured that your work is backed up at all times. In case of failure, the system automatically shifts to a standby setup, maintaining uninterrupted progress with no wasted time or resources.

- **Application development, helping you build applications quickly**

In enterprises, complex project-level tasks often require understanding the task, breaking it down into multiple decision-making questions, and then calling various subsystems to execute. MaaS uses advanced open-source models to accurately grasp business goals, break down tasks, and create multiple solutions. It helps businesses quickly and intelligently build and deploy LLM-powered applications.

Supported Region

MaaS is available only in the Hong Kong region.

Usage Process

The table below shows the core process of using MaaS.

Table 1-1 MaaS usage process

Module	Step	Operation	Helpful Links
Authorization	Configuring access authorization	All users (including individual users) can use MaaS only after agency authorization on ModelArts. Otherwise, unexpected errors may occur.	<ul style="list-style-type: none"> • 2.1 Creating an IAM User and Granting ModelArts Studio (MaaS) Permissions • 2.2 Configuring ModelArts Agency Authorization for Using ModelArts Studio (MaaS)

Module	Step	Operation	Helpful Links
Real-time inference service	Checking built-in models in the Model Square	ModelArts Studio provides various open-source models. You can check them on the Model Square page. The model details page shows all necessary information. You can choose suitable models for training and inference to incorporate into your enterprise systems.	4.1 Viewing a Built-in Model in ModelArts Studio (MaaS)
	Deploying a model	Deploy built-in models from Model Square using compute resources so that you can call the models in Model Experience or other service environments.	4.2 Deploying a Model Service in ModelArts Studio (MaaS)
API calls	Calling a model service	After a model is deployed, call the model service in other service environments for prediction.	4.4 Calling a Model Service in ModelArts Studio (MaaS)

2 Configuring ModelArts Studio (MaaS) Access Authorization

[2.1 Creating an IAM User and Granting ModelArts Studio \(MaaS\) Permissions](#)

[2.2 Configuring ModelArts Agency Authorization for Using ModelArts Studio \(MaaS\)](#)

[2.3 Configuring the Missing ModelArts Studio \(MaaS\) Permissions](#)

2.1 Creating an IAM User and Granting ModelArts Studio (MaaS) Permissions

The agency created in [2.2 Configuring ModelArts Agency Authorization for Using ModelArts Studio \(MaaS\)](#) almost has all permissions of dependent services. If your Huawei Cloud account meets your permissions requirements, you can skip this section.

ModelArts allows you to configure fine-grained permissions for refined management of resources and permissions. This type of feature is commonly used in scenarios with large-scale enterprise users. If you want to precisely control the scope of permissions granted to an agency, use custom authorization for MaaS.

This section describes how to configure fine-grained permissions for IAM users.

Operation Scenarios

Identity and Access Management (IAM) provides permissions management for secure access to your Huawei Cloud services and resources.

IAM is free of charge. You pay only for the cloud resources in your account. When you successfully sign up for Huawei Cloud, your account is automatically created. Your account has full access permissions for your cloud services and resources and makes payments for the use of these resources. For details, see [What Is IAM?](#)

Authorization Process

Creating a user group and assigning permissions: If you do not want to create an account for every person in your enterprise, you can use IAM. Only the enterprise's

administrator needs to create an account. The account can be used to create user groups and assign permissions. Then, the IAM users created for the enterprise personnel can be added to different user groups based on their job responsibilities. For details, see [Creating a User Group and Assigning Permissions](#).

Creating an IAM user and logging in: Create an IAM user and add it to the user group to obtain permissions. Log in to the [ModelArts Studio \(MaaS\) console](#) as an IAM user and use resources within the permissions scope. For details, see [Creating an IAM User and Logging In](#).

Billing

IAM manages user access to ModelArts resources at no cost.

Prerequisites

- You have logged in to the console as an administrator.
- You have learned about the permissions that can be added to user groups for using ModelArts and dependent services so that you can select them based on your requirements. For details about the permissions supported by MaaS, see [Table 2-1](#).

Table 2-1 Service authorizations

Target Service	Authorization Description	IAM Permission	Mandatory
ModelArts	This permission allows IAM users to use ModelArts. The IAM users with the ModelArts CommonOperations permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to sub-users.	ModelArts CommonOperations	Yes
	The IAM users with the ModelArts FullAccess permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.	ModelArts FullAccess	No Select either ModelArts FullAccess or ModelArts CommonOperations .

Target Service	Authorization Description	IAM Permission	Mandatory
Object Storage Service (OBS)	This permission allows IAM users to use OBS. ModelArts data management, development environments, training jobs, and model deployment require OBS for forwarding data .	OBS OperateAccess	Yes
Software Repository for Container (SWR)	This permission allows IAM users to use SWR. ModelArts custom images require the SWR FullAccess permission.	SWR OperateAccess	Yes
Cloud Eye	This permission allows IAM users to use Cloud Eye. Using Cloud Eye, you can view the running statuses of ModelArts real-time services and model loads, and set monitoring alarms.	CES FullAccess	Yes
Simple Message Notification (SMN)	This permission allows IAM users to use SMN. SMN is used with Cloud Eye.	SMN FullAccess	Yes
Virtual Private Cloud (VPC)	When creating a ModelArts dedicated resource pool, IAM users require VPC permissions so that they can customize networks.	VPC FullAccess	No

Configuring Common Operation Permissions for MaaS

Step 1 Create a user group.

Log in to the [IAM console](#) as an administrator. Choose **User Groups** and **Create User Group**. Enter a user group name and click **OK**.

Step 2 Configure permissions for the user group.

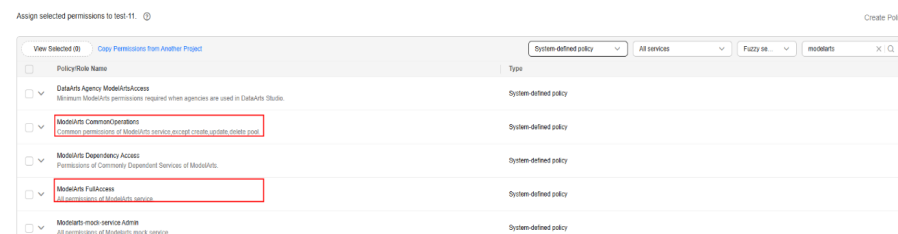
In the user group list, locate the created user group, click **Authorize** in the **Operation** column, and perform the following operations.

1. Assign permissions for using ModelArts. Choose a system policy and type **ModelArts** into the search box. Select either **ModelArts FullAccess** or **ModelArts CommonOperations**.

The differences between the options are as follows:

- The users with the **ModelArts CommonOperations** permission can only use resources, but cannot create, update, or delete any dedicated resource pool. You are advised to assign this permission to IAM users.
- The IAM users with the **ModelArts FullAccess** permission have all access permissions, including creating, updating, and deleting dedicated resource pools. Exercise caution when assigning this permission.

Figure 2-1 Assigning permissions for using ModelArts



2. Configure permissions for using other dependent services. For example, to use OBS, search for **OBS** and select **OBS OperateAccess**. ModelArts training jobs use OBS to forward data. Therefore, the permissions for using OBS are necessary.

For permissions of more cloud services, such as SWR, see [Table 2-1](#).

3. Click **Next** and set the minimum authorization scope. Select **Region-specific projects**, select the region to be authorized, and click **OK**.
4. A message is displayed, indicating that the authorization is successful. View the authorization information and click **Finish**. It takes 15 to 30 minutes for the authorization to take effect.

Step 3 Create an IAM user. In the navigation pane on the left of the IAM console, choose **Users**. In the right pane, click **Create User** in the upper right corner. On the **Create User** page, add multiple users. Set parameters as prompted and click **Next**.

Step 4 Add the IAM user to the user group. On the **Add User to Group** page, select a user group and click **Create**. The system adds the created users to the user group.

Step 5 [Log in](#) and verify the permission.

In the authorized region, perform the following operations:

- Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose your desired type of AI dedicated resource pools and create one. You should not be able to create a new resource pool if the **ModelArts CommonOperations** permission has taken effect.
- Choose any other service in **Service List**. (Assume that the current policy contains only **ModelArts CommonOperations**.) If a message appears indicating that you have insufficient permissions to access the service, the **ModelArts CommonOperations** permission has taken effect.

- Choose **Service List > ModelArts**. In the navigation pane of the ModelArts console, choose **Algorithm Management** and click **Create Algorithm**. You should be able to access the corresponding OBS path if the OBS permission has taken effect.
- Verify other optional permissions according to [Table 2-1](#).

----End

2.2 Configuring ModelArts Agency Authorization for Using ModelArts Studio (MaaS)

Using ModelArts' MaaS service requires proper permission management for smooth operation and data safety. ModelArts uses the IAM system to control all features. Administrators can assign specific user permissions. Incorrect settings may cause errors that disrupt service access.

All users must authorize ModelArts access before using the MaaS service; otherwise, errors might happen. Once authorized, individual users can start using ModelArts without managing detailed permissions.

Operation Scenarios

If you have already obtained ModelArts access authorization, you are all set for MaaS – no extra configuration needed. Otherwise, get ModelArts access authorization first, then you can use MaaS.

ModelArts needs to access other services for executing tasks. For example, ModelArts needs to access OBS to read your data for training. For security purposes, ModelArts must be authorized to access other cloud services. This is agency authorization.

ModelArts provides one-click automatic authorization. You can quickly configure agency authorization on the **Permission Management** page of ModelArts. Then, ModelArts will automatically create an agency for you and configure it in ModelArts.

This section introduces one-click automatic authorization. It allows you to grant permissions to IAM users, federated users (virtual IAM users), agencies, or all users with one click.

Notes and Constraints

- Huawei Cloud account
 - Only a Huawei Cloud account can use an agency to authorize the current account or all IAM users under the current account.
 - Multiple IAM users or accounts can use the same agency.
 - A maximum of 50 agencies can be created under an account.
 - If you use ModelArts for the first time, add an agency. Generally, common user permissions are sufficient for your requirements. You can also customize permissions for refined permissions management.
- IAM user

- If you have obtained the authorization, you can view the authorization information on the **Permission Management** page.
- If you have not been authorized, ModelArts will display a message indicating that you have not been authorized when you access the **Add Authorization** page. In this case, contact your administrator to add authorization.

Billing

IAM manages user access to ModelArts resources at no cost.

Billing depends on resource usage, including compute resources like vCPUs, GPUs, and NPUs, as well as storage resources like EVS disks and OBS. For details, see [Billing](#).

Prerequisites

You have [registered a Huawei account and enabled Huawei Cloud services](#).

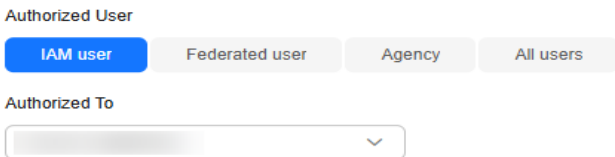
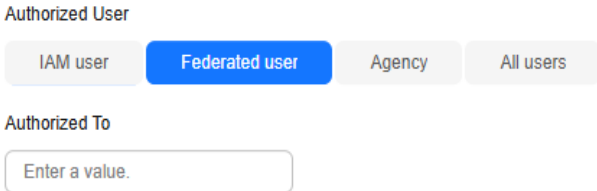
MaaS Agency Authorization

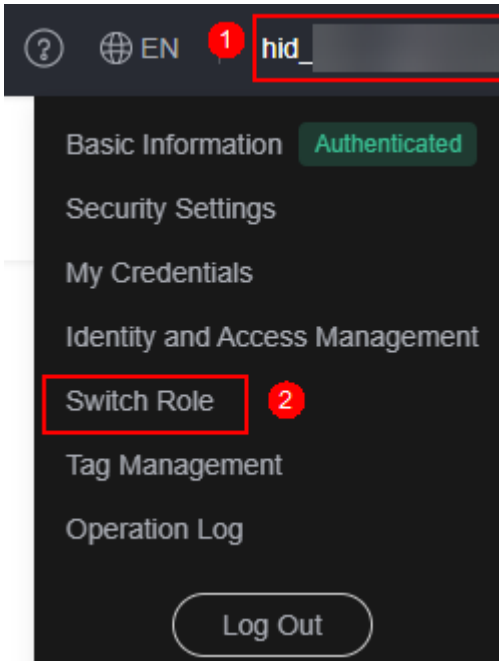
1. Log in to the [ModelArts console](#) and follow these steps based on your console version.
 - New version: In the navigation pane on the left, choose **System Management > Permission Management**.
 - Old version: In the navigation pane, choose **Settings**.
2. Click **Add Authorization**. On the displayed page, set parameters based on [Table 2-2](#).

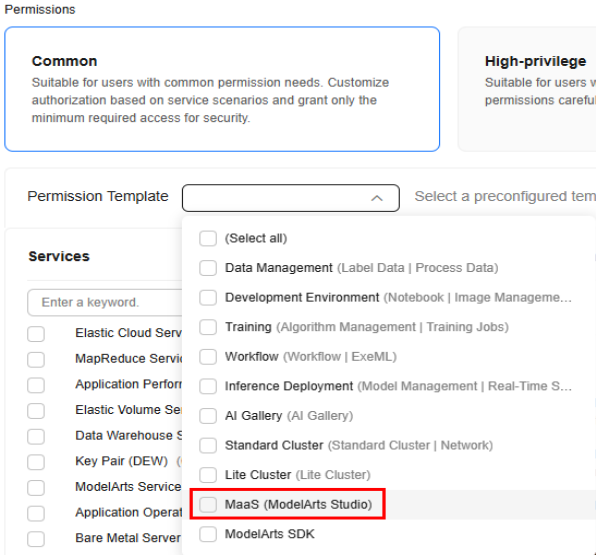
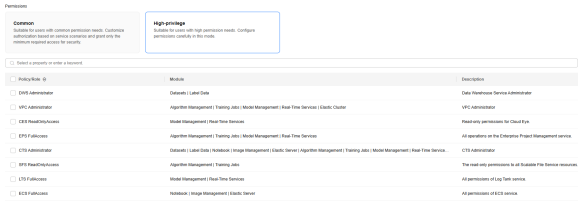
This example shows how to add authorization for an IAM user. For details, see the "Example" column in [Table 2-2](#).

Table 2-2 Authorization parameters

Parameter	Description	Example
Authorized User	<p>The options are IAM user, Federated user, Agency, and All users.</p> <ul style="list-style-type: none">• IAM user: You can use a tenant account to create IAM users and assign permissions for specific resources. Each IAM user has their own identity credentials (password and access keys) and uses cloud resources based on the assigned permissions. For details about IAM users, see IAM User.• Federated user: A federated user is also called a virtual enterprise user. For details about federated users, see Configuring Identity Federation.• Agency: You can create agencies in IAM. For details about how to create an agency in IAM, see Creating an Agency and Assigning Permissions.• All users: If you select this option, the agency permissions will be granted to all IAM users under the account, including those created in the future. For individual users, select All users.	Select IAM user .

Parameter	Description	Example
Authorized To	<p>This parameter is not displayed when Authorized User is set to All users.</p> <ul style="list-style-type: none">• IAM user: Select an IAM user and configure an agency for the IAM user. <p>Figure 2-2 Selecting an IAM user</p>  <p>• Federated user: Enter the username or user ID of the target federated user.</p> <p>Figure 2-3 Entering a federated user</p>  <ul style="list-style-type: none">• Agency: Select an agency name. You can create an agency under account A and grant the agency permissions to account B. When using account B to log in to ModelArts console, you can switch the role in the upper right corner of the console ModelArts console to account A and use the agency permissions of account A.	Select an IAM user and configure an agency for the IAM user.

Parameter	Description	Example
	<p>Figure 2-4 Switch Role</p>  <p>The screenshot shows a dark-themed user interface. At the top, there is a header bar with a question mark icon, a globe icon, the text 'EN', and a red circle with the number '1' next to the text 'hid_'. Below the header, there is a menu with several options: 'Basic Information' (with a green 'Authenticated' badge), 'Security Settings', 'My Credentials', 'Identity and Access Management', 'Switch Role' (highlighted with a red box and a red circle with the number '2'), 'Tag Management', and 'Operation Log'. At the bottom of the menu is a 'Log Out' button.</p>	
Agency	<ul style="list-style-type: none">• Use existing: If there are agencies in the list, select an available one to authorize the selected user. Click the drop-down arrow next to an agency name to view its permission details.• Add agency: If there is no available agency, create one. If you use ModelArts for the first time, select Add agency.	Select Add agency .
Add agency > Agency Name	<p>ModelArts automatically creates an agency name for you, but it is editable.</p> <p>ModelArts creates agency names using these rules:</p> <ol style="list-style-type: none">1. When authorizing an IAM user, the agency name defaults to ma_agency_[IAM username].2. When authorizing other types of users, the agency name defaults to modelarts_agency.3. If an agency name generated based on preceding rules already exists, a four-digit random code is automatically suffixed to the name.	N/A

Parameter	Description	Example
Add agency > Permissions > Common	<p>Choose this option to customize permissions with minimal access for enhanced security.</p> <p>Select MaaS (ModelArts Studio) from the Permission Template drop-down list.</p> <p>Figure 2-5 Common</p> 	Select MaaS (ModelArts Studio) .
Add agency > Permissions > High-privilege	<p>High-privilege grants elevated access permissions and is ideal for users needing administrator permissions.</p> <p>Choose this option for advanced access but manage permissions cautiously when using it.</p> <p>Figure 2-6 High-privilege</p> 	N/A

3. Click **Create**.
- Log in to the **ModelArts Studio (MaaS) console**. If no permission configuration information is displayed on the page, the agency configuration is complete.
- You can also go to the **Permission Management** page to check permission details. For details, see **Viewing Authorized Permissions**.

Viewing Authorized Permissions

You can view the configured authorizations on the **Permission Management** page. To do so, click **View Permissions** in the **Operation** column to view the details.

Figure 2-7 Viewing permissions

Authorized To	Authorized User	Authorization Type	Authorization Content	Creation Time	Operation
ht...	iam user	Agency	modelarts_agency	Apr 08, 2025 17:03:16 GMT+08:00	View Permissions Delete

Figure 2-8 Common permission details

View Permissions

Authorized Toht_...
Agency Namemodelarts_agency
Agency Permission22 permissions

Services

Elastic Cloud Server (ECS) (13/13)

MapReduce Service (MRS) (8/8)

Application Performance M... (1/1)

Elastic Volume Service (EVS) (4/4)

Data Warehouse Service (... (3/3)

Key Pair (DEW) (4/4)

ModelArts Service (ModelA... (86/86)

Application Operations Man... (7/7)

Bare Metal Server (BMS) (2/2)

Image Management Servic... (2/2)

Data Lake Insight (DLI) (14/14)

Simple Message Notificatio... (5/5)

Cloud Container Engine (C... (12/12)

Object Storage Service (O... (23/23)

Cloud Eye (CES) (1/1)

Virtual Private Cloud (VPC) (11/11)

Permissions

ModuleElastic Cluster

Total permissions: 13

ecs:cloudServers:create
create servers

ecs:cloudServers:delete
delete servers

ecs:cloudServers:get
Display Details About an ECS

ecs:cloudServers:start
Start ECSs in a Batch

ecs:cloudServers:stop
Stop ECSs in a Batch

ecs:cloudServers:reboot
Restart ECSs in a Batch

ecs:cloudServers:redeploy
Redeploy an ECS based on scheduled...

ecs:cloudServers:listServerInterfaces
Query NICs of an ECS

ecs:cloudServers:changeVpc
Change a VPC for an ECS

ecs:cloudServers:flavors:get
Query Specifications and Expansion D...

ecs:quotas:get
Query Quota

ecs:cloudServers:batchSetServerTags
Add tags to or delete tags from an ECS

ecs:cloudServers:list
Display Details About ECSs

Cancel

OK

Modifying the Authorization Scope

- To modify the authorization scope, click **To view all agency permissions, access IAM** in the **View Permissions** dialog box.

Figure 2-9 Modifying permissions in IAM

View Permissions

Agency Namemodelarts_agency
Agency Permission22 permissions

Services

Elastic Cloud Server (ECS) (13/13)

MapReduce Service (MRS) (8/8)

Application Performance M... (1/1)

Elastic Volume Service (EVS) (4/4)

Data Warehouse Service (... (3/3)

Key Pair (DEW) (4/4)

ModelArts Service (ModelA... (86/86)

Application Operations Man... (7/7)

Bare Metal Server (BMS) (2/2)

Image Management Servic... (2/2)

Data Lake Insight (DLI) (14/14)

Simple Message Notificatio... (5/5)

Cloud Container Engine (C... (12/12)

Object Storage Service (O... (23/23)

Cloud Eye (CES) (1/1)

Virtual Private Cloud (VPC) (11/11)

Permissions

ModuleElastic Cluster

Total permissions: 13

ecs:cloudServers:create
create servers

ecs:cloudServers:delete
delete servers

ecs:cloudServers:get
Display Details About an ECS

ecs:cloudServers:start
Start ECSs in a Batch

ecs:cloudServers:stop
Stop ECSs in a Batch

ecs:cloudServers:reboot
Restart ECSs in a Batch

ecs:cloudServers:redeploy
Redeploy an ECS based on scheduled...

ecs:cloudServers:listServerInterfaces
Query NICs of an ECS

ecs:cloudServers:changeVpc
Change a VPC for an ECS

ecs:cloudServers:flavors:get
Query Specifications and Expansion D...

ecs:quotas:get
Query Quota

ecs:cloudServers:batchSetServerTags
Add tags to or delete tags from an ECS

ecs:cloudServers:list
Display Details About ECSs

ModelArts permissions are displayed

To view all agency permissions, access IAM

Cancel

OK

- Go to the **Agencies** page on the **IAM console**, click the name of the agency you want to modify, and update the basic information as needed. Select your required validity period.

Issue 01 (2025-09-17)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

14

Figure 2-10 Agency information

The screenshot shows the 'Basic Information' tab of the 'Agency information' form. The 'Agency Name' field is populated with 'modelarts_agency'. The 'URN' field is empty. The 'Agency Type' is set to 'Cloud service'. The 'Cloud Service' dropdown is set to 'ModelArts'. The 'Validity Period' dropdown is set to 'Unlimited'. The 'Description' text area contains 'Created by ModelArts service.' and shows a character count of 29/255. The 'OK' and 'Cancel' buttons are at the bottom.

3. On the **Permissions** page, click **Authorize**, select policies or roles, and click **Next**. Select the scope for minimum authorization and click **OK**.

When setting the minimum authorization scope, you can select specified projects or all resources. If you select **All resources**, the selected permissions will be applied to all resources.

Deleting an Authorization

To better manage your authorization, you can delete the authorization of an IAM user or delete the authorizations of all users in batches. The deletion operation cannot be undone.

- **Deleting an authorization of a user**

The authorizations configured for the IAM user of the current account are displayed on the **Permission Management** page. You can locate an authorization, click **Delete** in the **Operation** column, enter **DELETE** in the text box, and click **OK**. After it is deleted, the user cannot use ModelArts functions.

- **Deleting authorizations in batches**

On the **Permission Management** page, click **Clear Authorization** above the authorization list. Enter **DELETE** in the text box and click **OK**. After the deletion, the account and all its IAM users cannot use ModelArts functions.

FAQs

- How do I configure authorization when I use ModelArts for the first time?
On the **Add Authorization** page, set **Agency** to **Add agency** and select **Common User**, which provides the permissions to use all basic ModelArts functions. For example, you can access data, and create and manage training jobs. Select this option generally.
- How do I obtain access keys (AK/SK)?
You will need to obtain an access key if you are using access key authentication to access certain functions like accessing model services. For details, see [How Do I Obtain an Access Key?](#)

- How do I delete an agency?
Go to the [IAM console](#), choose **Agencies** in the navigation pane, and delete the target agency. For details, see [Deleting or Modifying Agencies](#).
- Why is a message indicating insufficient permissions displayed when I access a page on the [ModelArts console](#)?
The permissions configured for the user agency are insufficient or the module has been upgraded. The authorization information needs to be updated. In this case, you can add authorization as prompted. For details, see [2.3 Configuring the Missing ModelArts Studio \(MaaS\) Permissions](#).

2.3 Configuring the Missing ModelArts Studio (MaaS) Permissions

When you use MaaS, the system will alert you when permissions are incorrect or absent. Without granting these permissions promptly, certain features will not work well, impacting your usage. To keep your services running smoothly, review and configure missing permissions quickly to prevent disruptions.

Prerequisites

You have registered a Huawei ID and enabled Huawei Cloud services, performed real-name authentication, and ensure your account is not frozen or in arrears before using ModelArts. For details, see [Signing Up for a HUAWEI ID and Enabling Huawei Cloud Services](#) and [Real-Name Authentication](#).

Billing

Authorization is free. But using data storage, importing models, or deploying services may incur fees because they rely on OBS and SWR. For details, see [Billing Overview](#).

Adding Dependency Service Authorization

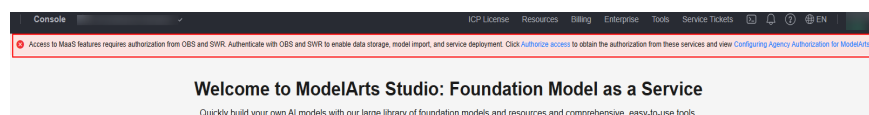
Access to MaaS features requires authorization from OBS and SWR. Authenticate with OBS and SWR to enable data storage, model import, and service deployment.

A message appears at the top of the [ModelArts Studio \(MaaS\) console](#) if dependency authorization has not been configured, prompting you to configure it.

Tenant user: Click **Authorize access** to go to the permissions management page on the [ModelArts console](#) and add dependency service permissions. For details, see [MaaS Agency Authorization](#).

IAM user: Contact the administrator.

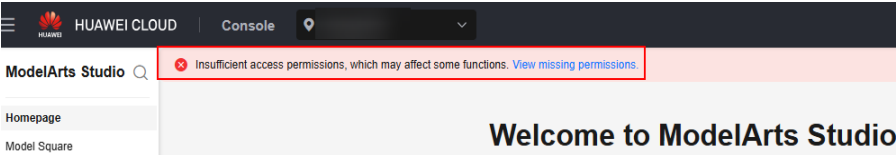
Figure 2-11 Dependency authorization prompt



Adding Missing Permissions Using the Tenant Account

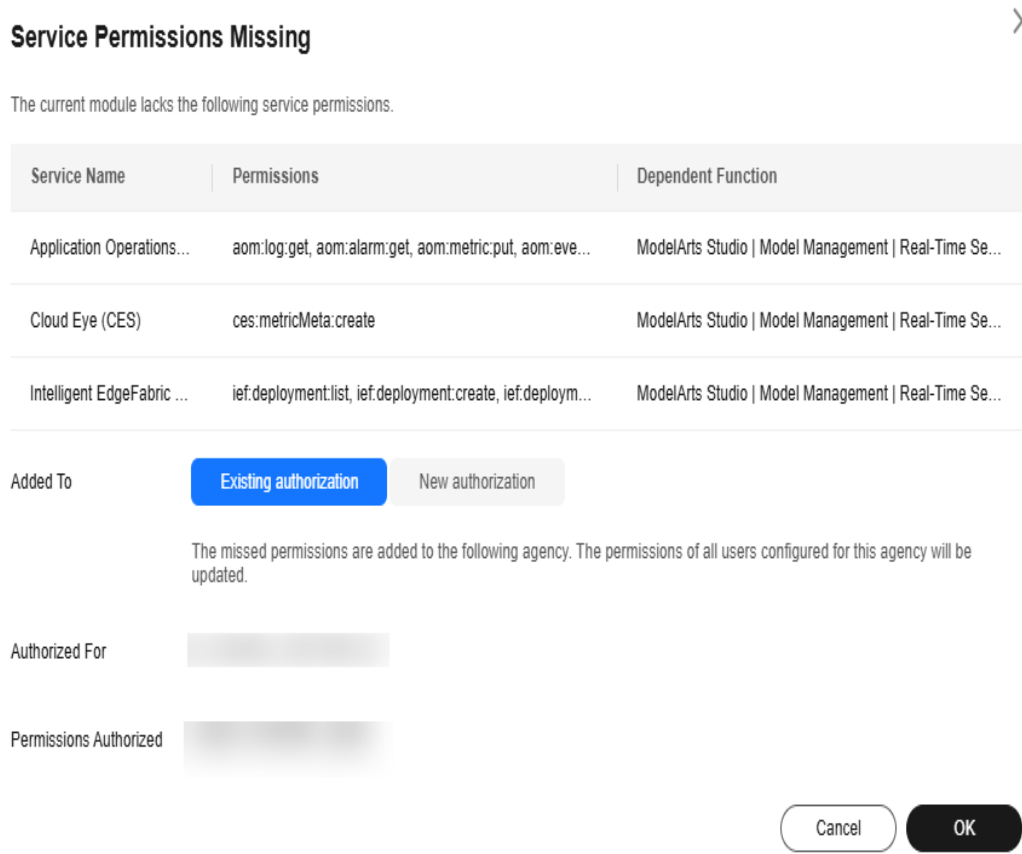
A permission error message appears at the top of the **ModelArts Studio (MaaS) console** if necessary permissions are not granted.

Figure 2-12 Permissions missing



You can click **View missing permissions** in the pop-up prompt. In the **Service Permissions Missing** dialog box, select **Existing authorization** or **New authorization** as required, and click **OK**.

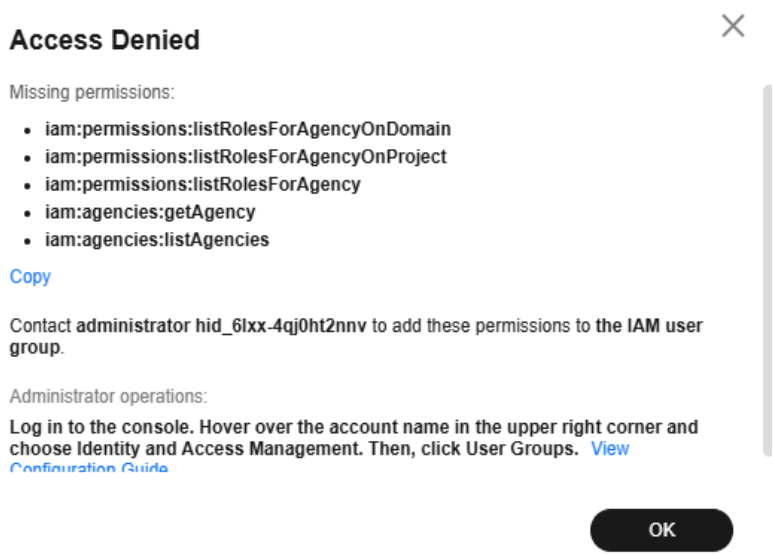
Figure 2-13 Service permissions missing



Adding Missing Permissions Using an IAM Account

The **ModelArts Studio (MaaS) console** will display an **Access Denied** dialog box if you lack necessary permissions. Create a custom policy, assign it to the target user group, review any missing service permissions, and contact your administrator to configure the necessary permissions.

Figure 2-14 Access Denied dialog box



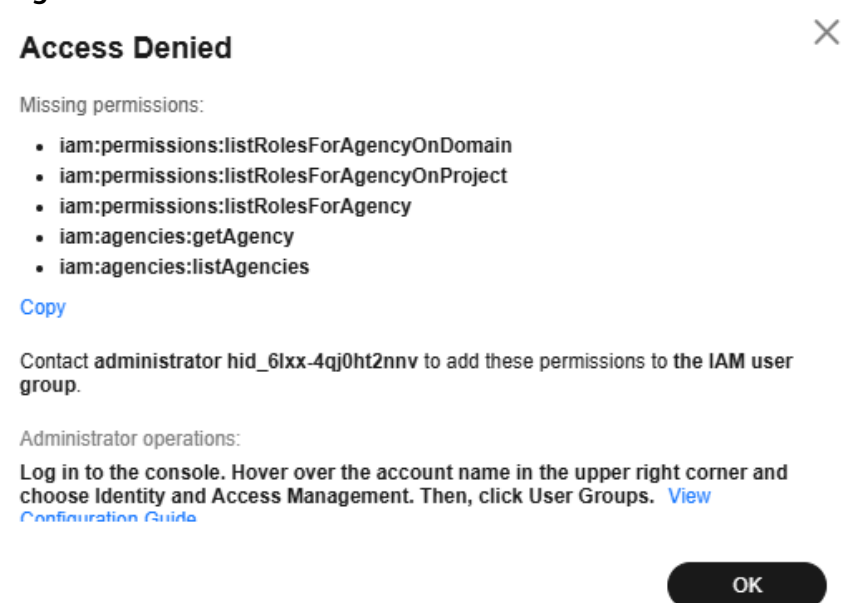
For details about the missing permissions, see [Table 2-3](#). For details, see [Actions](#).

Table 2-3 Missing permissions

Permission	Description
iam:permissions:listRolesForAgencyOnDomain	Querying permissions of an agency for a global service project
iam:permissions:listRolesForAgencyOnProject	Querying permissions of an agency for a region-specific project
iam:permissions:listRolesForAgency	Querying all permissions of an agency
iam:agencies:getAgency	Querying agency details
iam:agencies:listAgencies	Querying agencies in specified conditions

1. Create a custom policy.
 - a. In the **Access Denied** dialog box, click **Copy** to save the missing permissions and click **OK**.

Figure 2-15 Access Denied



- Hover over your account in the upper right corner and click **Identity and Access Management**.
- In the navigation pane of the [IAM console](#), choose **Permissions > Policies/Roles**.
- On the **Policies/Roles** page, click **Create Custom Policy** in the upper right corner.
- On the **Create Custom Policy** page, configure parameters and click **OK**.

Figure 2-16 Creating a custom policy

★ Policy Name

policy0ya08y

Policy View

Visual editor

JSON

★ Policy Content

1 {

2 "Version": "1.1",

3 "Statement": [

4 {

5 "Effect": "Allow",

6 "Action": [

7 "iam:permissions:listRolesForAgencyOnDomain",

8 "iam:permissions:listRolesForAgencyOnProject",

9 "iam:permissions:listRolesForAgency",

10 "iam:agencies:getAgency",

11 "iam:agencies:listAgencies"

12]

13 }

14]

15 }

Select Existing Policy/Role

Format Content

Description

Enter a brief description.

Scope

Global services

OK

Cancel

Table 2-4 Parameters for creating a custom policy

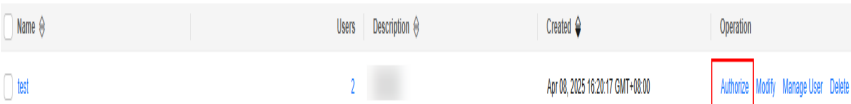
Parameter	Description	Example
Policy Name	Enter a policy name.	policykl631g
Policy View	Click JSON .	JSON

Parameter	Description	Example
Policy Content	Paste the permission policy saved in step 1.a in [] of the Statement parameter.	<pre>{ "Version": "1.1", "Statement": [{ "Effect": "Allow", "Action": ["iam:permissions:listRolesForAgencyOnDomain", "iam:permissions:listRolesForAgencyOnProject", "iam:permissions:listRolesForAgency", "iam:agencies:getAgency", "iam:agencies:listAgencies"] }] }</pre>
Description	Enter a policy description.	N/A
Scope	Use the default value Global services .	Global services

2. Add the custom policy to the target user group.
- a. In the navigation pane on the left of the **IAM console**, click **User Groups**.

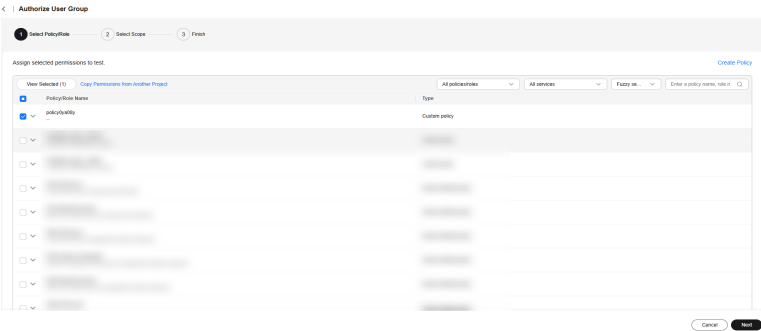
b. On the **User Groups** page, search for the target user group and click **Authorize** in the **Operation** column.

Figure 2-17 Authorize



- c. On the displayed page, select the policy created in [step 1](#), click **Next**, select the authorization scope as required, and click **OK**.

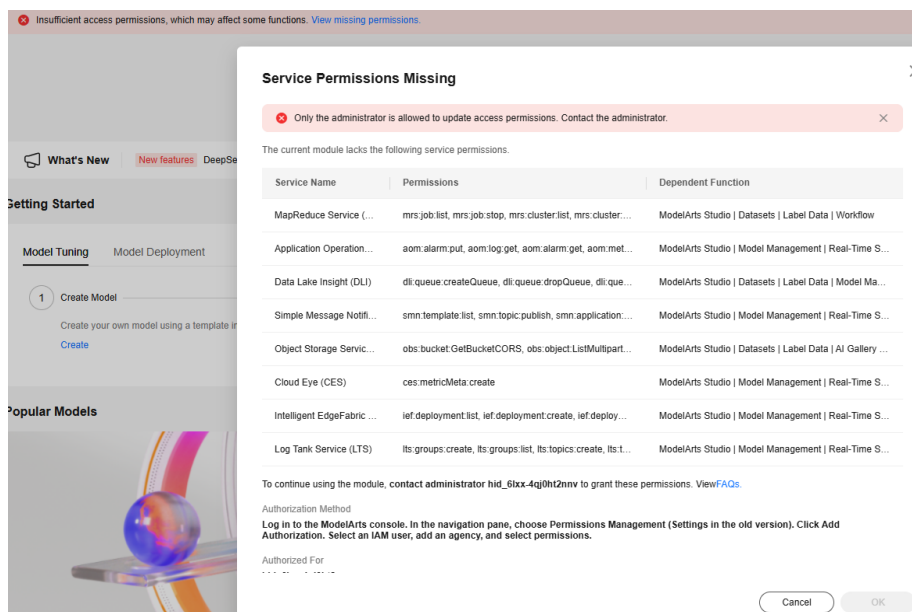
Figure 2-18 Authorization page



- d. In the **Information** dialog box, read the information carefully and click **OK**.

3. Review and configure the missing service permissions.
 - a. Log in to the **ModelArts Studio (MaaS) console** and click **View missing permissions** in the upper part of the page. In the **Service Permissions Missing** dialog box, view the missing service permissions.

Figure 2-19 Service permissions missing



- b. Contact the administrator to configure the missing service permissions. For details, see **MaaS Agency Authorization**.

FAQs

- How do I obtain access keys (AK/SK)?

You will need to obtain an access key if you are using access key authentication to access certain functions like accessing model services. For details, see **How Do I Obtain an Access Key?**
- How do I delete an agency?

Go to the **IAM console**, choose **Agencies** in the navigation pane, and delete the target agency. For details, see **Deleting or Modifying Agencies**.

3

Preparing ModelArts Studio (MaaS) Resources

Before using MaaS, create resource pools.

Creating a Resource Pool

When deploying a model in ModelArts Studio, you need to select a resource pool. MaaS supports dedicated resource pools.

The resources provided in a dedicated resource pool are exclusive and more controllable. To use a dedicated resource pool, create it and select the dedicated resource pool during AI development. MaaS can use ModelArts Standard dedicated resource pools for model training and inference. For details about how to create a dedicated resource pool, see [Creating a Standard Dedicated Resource Pool](#).

NOTE

The resource pool and MaaS must be in the same region. Otherwise, the resource pool cannot be selected.

4 ModelArts Studio (MaaS) Real-Time Inference Services

- [4.1 Viewing a Built-in Model in ModelArts Studio \(MaaS\)](#)
- [4.2 Deploying a Model Service in ModelArts Studio \(MaaS\)](#)
- [4.3 Managing My Services in ModelArts Studio \(MaaS\)](#)
- [4.4 Calling a Model Service in ModelArts Studio \(MaaS\)](#)
- [4.5 ModelArts Studio \(MaaS\) API Call Specifications](#)
- [4.6 Creating a Multi-Turn Dialogue in ModelArts Studio \(MaaS\)](#)

4.1 Viewing a Built-in Model in ModelArts Studio (MaaS)

ModelArts Studio provides various open-source models. You can check them on the Model Square page. The model details page shows all necessary information. You can choose suitable models for training and inference to incorporate into your enterprise systems.

Prerequisites

You have [registered a Huawei account and enabled Huawei Cloud services](#).

Accessing the Model Square

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Model Square**.
3. In the **Model Filtering** area on the **Model Square** page, filter models by series, type, supported jobs, and context length, or search by model name.
For details about model series, see [Models](#).

Table 4-1 Model filtering

Filter Criteria	Description
Model	You can filter models by series, including all and DeepSeek.
Type	You can filter models by type, including all and text generation.
Supported Job Types	You can filter models by job type, including all and deployment.
Context Length	You can filter models by context length, including all, fewer than 16K, and 16K.

4.

Click **Model Details** below the target model. On the model details page, check the model's introduction, basic information, and version details.
5.

On the **Model Details** page, click **Deploy** in the upper right corner to use the model for training and inference.
The model does not support that task if you see a grayed-out button. For details about how to deploy a model service, see [4.2 Deploying a Model Service in ModelArts Studio \(MaaS\)](#).

Models

The table below lists the models supported by ModelArts Studio. For details about the models, go to the model details page.

Table 4-2 Models in the Model Square

Model		Type	Use Case	Supported Language	Supported Region	Model Introduction
DeepSeek	DeepSeek-R1	Text generation	Q&A and text generation inference	Chinese and English	CN-Hong Kong	DeepSeek-R1 model. It uses advanced technology for long-context understanding and fast inference. The model supports multi-modal interactions and API integrations. It enhances applications like intelligent customer service and data analytics, offering top cost-effectiveness for intelligent upgrades of enterprises.
	DeepSeek-V3	Text generation	Q&A and translation	Chinese and English	CN-Hong Kong	DeepSeek-V3 is a strong Mixture of Experts (MoE) language model. It uses a new load balancing method without extra loss and aims for better performance with multi-token predictions.
	DeepSeek-R1-Distill-Qwen-14B	Text generation	Q&A and text generation inference	Chinese and English	CN-Hong Kong	Qwen-14B is distilled from DeepSeek-R1 outputs and matches the capabilities of OpenAI o1-mini. DeepSeek-R1 performs similarly to OpenAI-o1 in math, coding, and inference tasks.

Model		Type	Use Case	Supported Language	Supported Region	Model Introduction
	DeepSeek-R1-Distill-Qwen-32B	Text generation	Q&A and text generation inference	Chinese and English	CN-Hong Kong	Qwen-32B is distilled from DeepSeek-R1 outputs and matches the capabilities of OpenAI o1-mini. DeepSeek-R1 performs similarly to OpenAI-o1 in math, coding, and inference tasks.
Deepseek-Coder		Text generation	Q&A and text inference	Chinese and English	CN-Hong Kong	DeepSeek Coder includes several code language models. Every model trains from scratch using 2 trillion tokens, with 87% being code and 13% English or Chinese text. It excels in multiple programming languages and performs well across various benchmarks among open-source code models.
Qwen	QwQ	Text generation	Q&A	English	CN-Hong Kong	QwQ is part of the Tongyi series of inference models. Unlike standard instruction-tuning models, QwQ excels at thinking and reasoning, delivering better results on complex tasks.
Qwen 2.5	Qwen 2.5	Text generation	Multilingual processing, mathematical inference, and Q&A	Chinese and English	CN-Hong Kong	Alibaba Cloud's Qwen 2.5 is a new addition to the Qwen series of LLMs. Qwen 2.5 offers various base and instruction-tuned language models, spanning sizes from 0.5 billion to 72 billion parameters.

Model		Type	Use Case	Supported Language	Supported Region	Model Introduction
	Qwen2.5-VL	Image understanding	Image understanding and Q&A	Chinese and English	CN-Hong Kong	Qwen-2.5-VL is an open-source multimodal visual language model created by Alibaba Cloud's Qwen team. It excels in visual and language understanding.
Qwen3	Qwen3	Text generation	Q&A	Chinese and English	CN-Hong Kong	The Qwen3 series includes LLMs and multimodal models created by the Qwen team. These models undergo extensive training using vast amounts of language and multimodal data, followed by fine-tuning with top-tier datasets.

4.2 Deploying a Model Service in ModelArts Studio (MaaS)

In MaaS, you can deploy built-in models from Model Square as services so that they can be called in service environments.

Operation Scenarios

Choose a model from **Model Square** for deployment. Once deployed, the model appears in the **My Services** list.

Billing

Model inference in MaaS uses compute and storage resources, which are billed. Compute resources are billed for running the model service. Storage resources are billed for storing data in OBS. You will also be billed for using SMN. For details, see [Model Inference Billing Items](#).

Constraints

ModelArts Studio has predefined the maximum input and output lengths for inference.

Table 4-3 Default maximum input and output lengths

Model	Default Maximum Input and Output Lengths (Token)
DeepSeek-R1-8K DeepSeek-V3-8K DeepSeek-R1-Distill-Qwen-14B-8K DeepSeek-R1-Distill-Qwen-32B-8K	8,192
DeepSeek-R1-16K DeepSeek-V3-16K QwQ-32B-16K	16,384
DeepSeek-R1-32K DeepSeek-V3-32K QwQ-32B-32K Qwen2.5-VL-7B-32K Qwen3-8B-32K DeepSeek-R1-Distill-Qwen-32B-32K	32,768
DeepSeek-V3-64K Qwen2.5-32B-64K	65,536
Other models	4,096

Prerequisites

You have prepared a dedicated resource pool. For details, see [3 Preparing ModelArts Studio \(MaaS\) Resources](#).

Procedure

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Real-Time Inference**.
3. In the **Real-Time Inference > My Services** tab, click **Deploy Model** in the upper right corner.

Table 4-4 Parameters for deploying a model service

Parameter		Description
Service Settings	Name	Enter a service name. The name can contain 1 to 64 characters, including only letters, digits, hyphens (-), and underscores (_). It must start with a letter.
	Description	Enter a description of up to 256 characters.
Model Settings	Model	Click Select Model and select a model from Model Square .
Resource Settings	Resource Pool Type	Only dedicated resource pools are supported. Dedicated resource pools are created separately and used exclusively.
	Instance Specifications	Select required instance specifications, which include the server type and model. Only resource specifications supported by the model are displayed.
	Instances	Set the number of servers.
Resource Settings	Queries Per Second (QPS)	Set the Queries Per Second (QPS) of the model service. Unit: queries/s NOTE If error code ModelArts.4206 is displayed during deployment, the allowed QPS has been exceeded. In this case, restart the service after the traffic limiting is finished.

Parameter		Description
More Settings	Event Notification	<p>Choose whether to enable event notification.</p> <ul style="list-style-type: none">This function is disabled by default, which means SMN is disabled.After this function is enabled, you will be notified of specific events, such as task status changes or suspected suspensions. In this case, you must configure the topic name and events.<ul style="list-style-type: none">Topic: topic of event notifications. Click Create Topic to create a topic on the SMN console.Event: events you want to subscribe to, for example, Running, Terminated, or Failed. <p>NOTE</p> <ul style="list-style-type: none">After you create a topic on the SMN console, add a subscription to the topic, and confirm the subscription. Then, you will be notified of events. For details about how to subscribe to a topic, see Adding a Subscription.SMN charges you for the number of notification messages. For details, see Billing.
	Auto Stop	<p>When using paid resources, choose whether to enable auto stop.</p> <ul style="list-style-type: none">If this function is enabled, configure the auto stop time. The value can be 1 hour, 2 hours, 4 hours, 6 hours, or Customize. When you enable this function, the service stops automatically when the time limit is reached. The time limit does not count down when the service is paused.This function is disabled by default, the service keeps running.

4. Click **Submit**.

In the **My Services** list, when the service status changes to **Running**, the model is deployed.

 **NOTE**

Using a dedicated resource pool for service deployments incurs no additional costs since its fees were already covered during purchase.

5. After the model is deployed, call its API. For details, see [4.4 Calling a Model Service in ModelArts Studio \(MaaS\)](#).

Viewing Service Information

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Real-Time Inference**.
3. Click the target service to access its details page.
 - **Details:** You can view the basic information about the service, including the service, model, and resource settings.
 - **Resources:** You can view the compute usage, video RAM usage, and resource monitoring information of the service.

Table 4-5 Resource monitoring parameters

Parameter	Description
Compute Usage	Compute usage of the service. When the request rate is low, the usage is displayed as 0.
Video Memory Usage	Video memory usage of the service.

- **Events:** You can view the event information of the service. Events are saved for one month and will be automatically cleared then.
- **Logs:** You can search for and view service logs.

Related Operations

- During AI development, you need to manage the service lifecycle, optimize deployed model services, and upgrade model services. For details, see [4.3 Managing My Services in ModelArts Studio \(MaaS\)](#).
- For details about how to call APIs, see [4.4 Calling a Model Service in ModelArts Studio \(MaaS\)](#).

4.3 Managing My Services in ModelArts Studio (MaaS)

4.3.1 Starting, Stopping, or Deleting a Service in ModelArts Studio (MaaS)

Stopping or Starting a Service

You can only stop a service when the service is in the Queuing, Starting, Running, Deploying, or Alarm state. You can only start a service when the service is in the **Deployment Failed** or Stopped state.

- Stopping a service
 - a. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
 - b. In the navigation pane on the left, choose **Real-Time Inference**.

- c. On the **Real-Time Inference** page, click the **My Services** tab. Click **Stop** in the **Operation** column of the target service.
- d. In the displayed dialog box, click **OK**.
- Starting a service
 - a. On the **Real-Time Inference** page, click the **My Services** tab. Click **Start** in the **Operation** column of the target service.
 - b. In the displayed dialog box, read the information carefully and click **OK**.
The service is billed when it is running.

Deleting a Service

A deleted task cannot be restored.

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Real-Time Inference**.
3. Click the **My Services** tab.
4. Find the target service. Choose **More > Delete** in the **Operation** column. In the displayed dialog box, enter **DELETE** and click **OK** to delete the service.

4.3.2 Scaling Model Service Instances in ModelArts Studio (MaaS)

When using inference with LLMs, service requirements can vary greatly, necessitating flexible scaling to handle load changes and ensure high availability and efficient resource use.

ModelArts Studio enables manual scaling of model service instances without disrupting service operation.

Prerequisites

[A model has been deployed](#) in ModelArts Studio (MaaS).

Notes and Constraints

The number of instances can only be changed when the model service is in the **Running** or **Alarm** state.

Billing

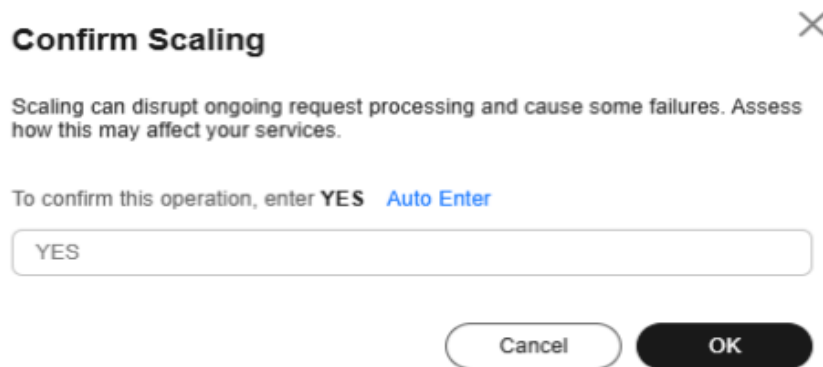
- Adding more model service instances increases resource usage for model inference in MaaS, leading to billing for compute and storage.
- Compute resources are billed for running the model service. Storage resources are billed for storing data in OBS. For details, see [Table 4-6](#).

Table 4-6 Billing items

Billing Item		Description	Billing Mode	Billing Formula
Compute resources	Public resource pools	Usage of compute resources. For details, see ModelArts Pricing Details .	Pay-per-use	Flavor unit price x Number of instances x Usage duration Package duration is preferentially deducted.
	Dedicated resource pools	Fees for dedicated resource pools are paid upfront upon purchase. There are no additional charges for service deployment. For details, see Billing Item .	N/A	N/A
Event notification (billed only when enabled)		This function uses Simple Message Notification (SMN) to send a message to you when the event you selected occurs. To use this function, enable event notification when creating a training job. For pricing details, see SMN Pricing Details .	Pay by actual usage	<ul style="list-style-type: none">• SMS: SMS notifications• Email: Email notifications + Downstream Internet traffic• HTTP or HTTPS: HTTP or HTTPS notifications + Downstream Internet traffic

Scaling Instances

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Real-Time Inference**.
3. On the **Real-Time Inference** page, click the **My Services** tab. Choose **More > Scale** in the **Operation** column of the target service.
4. Perform the following operations as required.
 - Scale-out: Increase the number of instances as required and click **OK**. In the **Scale Service** dialog box, click **OK**.
 - Scale-in: Reduce the number of instances as required and click **OK**. In the **Confirm Scaling** dialog box, confirm the information, enter **YES**, and click **OK**.

Figure 4-1 Confirm Scaling

In the **My Services** tab, click the service name to access its details page and check whether the change takes effect.

4.3.3 Modifying the QPS of a Model Service in ModelArts Studio (MaaS)

QPS is a crucial metric for evaluating a model service's processing capability. It measures the number of requests the system can handle per second in high-concurrency scenarios, directly impacting response speed and efficiency. Improper QPS configuration can increase user waiting time and reduce satisfaction. Therefore, it is essential to adjust the QPS flexibly to maintain service performance, optimize user experience, ensure continuity, and control costs.

ModelArts Studio allows you to manually modify the QPS limit of a model service instance without disrupting service operation.

Notes and Constraints

The QPS can be modified only when the model service is in the **Running** or **Alarm** state.

Modifying QPS

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane on the left, choose **Real-Time Inference**.
3. In the **Real-Time Inference > My Services** tab, choose **More > Set QPS** in the **Operation** column of the target service. In the displayed dialog box, modify the value and click **Submit**.

In the **My Services** tab, click the service name to access its details page and check whether the change takes effect.

4.3.4 Upgrading a Model Service in ModelArts Studio (MaaS)

Service upgrades involve optimizing deployed model services to enhance performance, add new features, fix bugs, and meet new requirements. This includes updating the model version, which means to replace the existing model with a newly trained version to improve prediction accuracy and adaptability.

Prerequisites

[A model has been deployed](#) in ModelArts Studio (MaaS).

Notes and Constraints

The model service can be upgraded only when it is in the **Running** or **Alarm** state.

Upgrading the Service

NOTE

- The service upgrade cannot be rolled back. During the upgrade, the existing service will continue to run normally.
 - During and after the upgrade, billing will continue based on the service's existing billing mode.
1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
 2. In the navigation pane on the left, choose **Real-Time Inference**.
 3. On the **Real-Time Inference** page, click the **My Services** tab.
 4. Find the target model service and choose **More > Upgrade Service** in the **Operation** column.
 5. In the displayed dialog box, select the target version and click **OK**.

4.4 Calling a Model Service in ModelArts Studio (MaaS)

Model services deployed in ModelArts Studio can be called in other service environments. This section describes how to call your deployed model from **My Services**. You can also call built-in services (free, commercial, or custom access point).

Operation Scenarios

When developing AI applications, developers need to deploy trained models into real-world services. This often involves manually setting up environments, handling dependencies, and writing deployment scripts. This approach takes significant time, risks errors, and leads to challenges like complicated setups, hard migrations, costly maintenance, and awkward updates.

MaaS offers a one-stop solution with unified APIs for system integration, along with built-in monitoring and logging features for efficient O&M.

Prerequisites

In the **My Services** tab of the **Real-Time Inference** page, there is a model service in the **Running**, **Updating**, or **Upgrading** state. For details, see [4.2 Deploying a Model Service in ModelArts Studio \(MaaS\)](#).

Step 1: Obtaining an API Key

When calling a model service deployed in MaaS, you need to enter an API key for API authentication. You can create a up to 30 keys. Each key is displayed only once after creation. Keep it secure. If the key is lost, it cannot be retrieved. In this case, create a new API key. For more information, see [5.1 Managing API Keys in ModelArts Studio \(MaaS\)](#).

1. Log in to [ModelArts Studio \(MaaS\) console](#) and select the target region on the top navigation bar.
2. In the navigation pane, choose **API Keys**.
3. On the **API Keys** page, click **Create API Key**, enter the tag and description, and click **OK**.

The tag and description cannot be modified after the key is created.

Table 4-7 Parameters

Parameter	Description
Tag	Tag of the API key. The tag must be unique. The tag can contain 1 to 100 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed.
Description	Description of the custom API key. The value can contain 1 to 100 characters.

4. In the **Your Key** dialog box, copy the key and store it securely.
5. After the key is saved, click **Close**.

After you click **Close**, the key cannot be viewed again.

Step 2: Calling a Model Service for Prediction

1. In the left navigation pane of the [ModelArts Studio \(MaaS\) console](#), choose **Real-Time Inference**.
2. On the **Real-Time Inference** page, click the **My Services** tab. Choose **More > View Call Description** in the **Operation** column of the target service.
3. On the displayed page, select an API type, copy the example call, modify the API information and API key, and use the information to call the model service API in the service environment.

The following shows sample code for REST APIs and OpenAI SDK.

- Use a common **requests** package.

```
import requests
import json

if __name__ == '__main__':
    url = "https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06****/v1/chat/completions"
    api_key = "<your_apiKey>" # Replace <your_apiKey> with the obtained API key.

    # Send a request.
    headers = {
        'Content-Type': 'application/json',
        'Authorization': f'Bearer {api_key}'
    }
```

```
data = {
    "model": "*****", # Model name for calling
    "max_tokens": 1024, # Maximum number of output tokens.
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "hello"}
    ],
    # Controls whether to enable streaming inference. The default value is False, indicating
    # that streaming inference is disabled.
    "stream": False,
    # Controls whether to show the number of tokens used during streaming output. This
    # parameter is valid only when stream is set to True.
    # "stream_options": {"include_usage": True},
    # A floating-point number that controls the sampling randomness. Smaller values make
    # the model more deterministic, while larger values make it more creative. The value 0 indicates
    # greedy sampling. The default value is 0.6.
    "temperature": 0.6
}
response = requests.post(url, headers=headers, data=json.dumps(data), verify=False)
# Print result.
print(response.status_code)
print(response.text)
```

– Use the OpenAI SDK.

```
from openai import OpenAI

if __name__ == '__main__':
    base_url = "https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06*****/v1"
    api_key = "<your_apiKey>" # Replace <your_apiKey> with the obtained API key.

    client = OpenAI(api_key=api_key, base_url=base_url)

    response = client.chat.completions.create(
        model="*****",
        messages=[
            {"role": "system", "content": "You are a helpful assistant"},
            {"role": "user", "content": "Hello"},
        ],
        max_tokens=1024,
        temperature=0.6,
        stream=False
    )
    # Print result.
    print(response.choices[0].message.content)
```

The model service API is the same as that of vLLM. [Table 4-8](#) only covers key parameters. For more information, see the vLLM official website. When enabling stream output for a model using the Ascend Cloud 909 image, you need to add the **stream_options** parameter with the value **{"include_usage": true}** to print the number of tokens used.

Table 4-8 Request parameters

Parameter	Mandatory	Default Value	Type	Description
url	Yes	No	Str	API URL. Assume that the URL is https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06*****/{endpoint} . The <i>{endpoint}</i> only supports the following APIs. For details, see API Calling . <ul style="list-style-type: none">• /v1/chat/completions• /v1/models
model	Yes	No	Str	Model name for calling. To obtain the model name, go to the Real-Time Inference page of ModelArts Studio, and choose More > Call in the Operation column; from there, you can see the model name.
messages	Yes	N/A	Array	Input question of the request.
messages.role	Yes	No	Str	Different roles correspond to different message types. <ul style="list-style-type: none">• system: developer-entered instructions like response formats and roles for the model to follow.• user: user-entered messages including prompts and context information.• assistant: responses generated by the model.• tool: information returned by the tool when the model calls it.

Parameter	Mandatory	Default Value	Type	Description
messages.content	Yes	No	Str	<ul style="list-style-type: none">When role is set to system, this parameter indicates the AI model's personality. <code>{"role": "system", "content": "You are a helpful AI assistant."}</code>When role is set to user, this parameter indicates the question asked by the user. <code>{"role": "user", "content": "Which number is larger, 9.11 or 9.8?"}</code>When role is set to assistant, this parameter indicates the content output by the AI model. <code>{"role": "assistant", "content": "9.11 is larger than 9.8."}</code>When role is set to tool, this parameter indicates the responses returned by the tool when the model calls it. <code>{"role": "tool", "content": "The weather in Shanghai is sunny today. The temperature is 10°C."}</code>
stream_options	No	No	Object	Controls whether to show the number of tokens used during streaming output. This parameter is valid only when stream is set to True . You need to set stream_options to {"include_usage": true} to print the number of tokens used.
max_tokens	No	16	Int	Maximum number of tokens that can be generated for the current task, including tokens generated by the model and reasoning tokens for deep thinking.
top_k	No	-1	Int	The candidate set size determines the sampling range during generation. For example, setting it to 50 means only the top 50 scoring tokens are sampled at each step. A larger size increases randomness; a smaller one makes the output more predictable.

Parameter	Mandatory	Default Value	Type	Description
top_p	No	1.0	Float	<p>Nucleus sampling. It keeps only the words with combined probabilities above the threshold p and removes the rest. These selected words are then normalized and sampled again.</p> <p>Lower settings reduce word options, making outputs focused and cautious. Higher settings expand word choices, creating varied and creative outputs.</p> <p>Adjust either temperature or top_p separately for best results, not both at once.</p> <p>Value range: 0 to 1. The value 1 indicates that all tokens are considered.</p>
temperature	No	0.6	Float	<p>Model sampling temperature. The higher the value, the more random the model output; the lower the value, the more deterministic the output.</p> <p>Adjust either temperature or top_p separately for best results, not both at once.</p> <p>Recommended value of temperature: 0.6 for DeepSeek-R1, DeepSeek-V3, and Qwen3 series, and 0.2 for Qwen2.5-VL series.</p>
stop	No	None	None/Str/List	<p>A list of strings used to stop generation. The output does not contain the stop strings.</p> <p>For example, if the value is set to ["You," "Good"], text generation will stop once either You or Good is reached.</p>
stream	No	False	Bool	<p>Controls whether to enable streaming inference. The default value is False, indicating that streaming inference is disabled.</p>

Parameter	Mandatory	Default Value	Type	Description
n	No	1	Int	<p>Number of responses generated for each input message.</p> <ul style="list-style-type: none">• If beam_search is not used, the recommended value range of n is $1 \leq n \leq 10$. If n is greater than 1, ensure that greedy_sample is not used for sampling, that is, top_k is greater than 1 and temperature is greater than 0.• If beam_search is used, the recommended value range of n is $1 < n \leq 10$. If n is 1, the inference request will fail. <p>NOTE For optimal performance, keep n at 10 or below. Large values of n can significantly slow down processing. Inadequate video RAM may cause inference requests to fail.</p>
use_beam_search	No	False	Bool	<p>Controls whether to use beam_search to replace sampling.</p> <p>When this parameter is used, the following parameters must be configured as required:</p> <ul style="list-style-type: none">• n: > 1• top_p: 1.0• top_k: -1• temperature: 0.0
presence_penalty	No	0.0	Float	<p>Applies rewards or penalties based on the presence of new words in the generated text. The value range is [-2.0,2.0].</p>
frequency_penalty	No	0.0	Float	<p>Applies rewards or penalties based on the frequency of each word in the generated text. The value range is [-2.0,2.0].</p>
length_penalty	No	1.0	Float	<p>Imposes a larger penalty on longer sequences in a beam search process.</p> <p>When this parameter is used, the following parameters must be configured as required:</p> <ul style="list-style-type: none">• top_k: -1• use_beam_search: true• best_of: > 1

- The following shows the sample response of a common **requests** package and OpenAI SDK.

```
{
  "id": "cmpl-29f7a172056541449eb1f9d31c*****",
  "object": "chat.completion",
  "created": 17231****,
  "model": "*****",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello. I'm glad to help. Is there anything I can help with?"
      },
      "logprobs": null,
      "finish_reason": "stop",
      "stop_reason": null
    }
  ],
  "usage": {
    "prompt_tokens": 20,
    "total_tokens": 38,
    "completion_tokens": 18
  }
}
```

- The following is a sample response of the chain-of-thought model:

```
messages = [{"role": "user", "content": "9.11 and 9.8, which is greater?"}]
response = client.chat.completions.create(model=model, messages=messages)
reasoning_content = response.choices[0].message.reasoning_content
content = response.choices[0].message.content
print("reasoning_content:", reasoning_content)
print("content:", content)
```

Table 4-9 Response parameters

Parameter	Type	Description
id	Str	Request ID
object	Str	Request task
created	Int	Timestamp when the request is created
model	Str	Model to call
choices	Array	Model-generated content
usage	Object	Request input length, output length, and total length <ul style="list-style-type: none">• prompt_tokens: number of input tokens.• completion_tokens: number of output tokens.• total_tokens: total number of tokens. Total number of tokens = Number of input tokens + Number of output tokens

Parameter	Type	Description
reasoning_content	Str	Model's thought process when it supports a chain of thought. For models that support a chain of thought, when streaming output is enabled, the reasoning appears in the reasoning_content field first, followed by the answer in the content field.
content	Str	Response of the model.

If the calling fails, you can adjust the script or runtime environment based on the error code.

Table 4-10 Common error codes

Error Code	Content	Description
400	Bad Request	The request contains syntax errors.
403	Forbidden	The server refused the request.
404	Not Found	The server cannot find the requested web page.
500	Internal Server Error	Internal service error.

API Calling

Assume that the API URL is **https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06*****/{endpoint}**. The *{endpoint}* parameter only supports the following APIs:

- /v1/chat/completions
- /v1/models

Notes:

- **/v1/models** does not need a request body for GET requests. However, **/v1/chat/completions** requires the POST method and a JSON request body.
- The common request header is **Authorization: Bearer YOUR_API_KEY**. For POST requests, **Content-Type: application/json** is also required.

Table 4-11 APIs

Type/API	/v1/models	/v1/chat/completions
Request method	GET	POST

Type/API	/v1/models	/v1/chat/completions
Usage	Obtains the list of supported models.	Chats with users.
Request body	No request body is needed. Just add the authentication information to the request header.	<ul style="list-style-type: none">• model: identifier of the model used.• messages: an array of messages. Each message must contain role (for example, user or assistant) and content.• Optional parameters like temperature and max_tokens control the diversity and length of the output.
Example request	GET https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06*****/v1/models HTTP/1.1 Authorization: Bearer YOUR_API_KEY	POST https://example.com/v1/infers/937cabe5-d673-47f1-9e7c-2b4de06*****/v1/chat/completions HTTP/1.1 Content-Type: application/json Authorization: Bearer YOUR_API_KEY <pre>{ "model": "*****", "messages": [{ "role": "user", "content": "Hello, how are you?" }], "temperature": 0.7 }</pre>
Response example	<pre>{ "data": [{ "id": "*****", "description": "Next-generation foundation model" }, { "id": "*****", "description": "Cost-effective alternative solution" }] }</pre>	<pre>{ "id": "*****", "object": "chat.completion", "choices": [{ "index": 0, "message": { "role": "assistant", "content": "I'm doing well, thank you! How can I help you today?" } }], "usage": { "prompt_tokens": 15, "completion_tokens": 25, "total_tokens": 40 } }</pre>

FAQs

How Long Does It Take for an API Key to Become Valid After It Is Created in MaaS?

A MaaS API key becomes valid a few minutes after creation.

Helpful Links

- [4.5 ModelArts Studio \(MaaS\) API Call Specifications](#)

- [4.6 Creating a Multi-Turn Dialogue in ModelArts Studio \(MaaS\)](#)

4.5 ModelArts Studio (MaaS) API Call Specifications

4.5.1 Sending a Chat Request (Chat/POST)

MaaS provides powerful real-time inference. You can deploy models on dedicated instances. This chapter describes the specifications for calling chat APIs.

API Information

Table 4-12 API information

Parameter	Description	Example Value
API Host	API URL for calling the model service.	https://api.modelarts-maas.com/v1/chat/completions
model	model parameter in an API call	Obtain the value from the View Call Description page. For more information, see 4.4 Calling a Model Service in ModelArts Studio (MaaS) .

Creating a Chat Request

- Authentication description
MaaS inference services support API key authentication. The authentication header is in the following format:
`'Authorization': 'Bearer API key of the region where the service is deployed'`
- The request and response parameters are as follows.

Table 4-13 Request parameters

Parameter	Mandatory	Default Value	Type	Description
model	Yes	None	Str	Model name for calling. For details about the value, see Table 4-12 .

Parameter	Mandatory	Default Value	Type	Description
messages	Yes	N/A	Array	<p>Input question. role shows the role, and content shows the dialog content. Example:</p> <pre>"messages": [{"role": "system", "content": "You are a helpful AI assistant."}, {"role": "user", "content": "Which number is larger, 9.11 or 9.8?"}]</pre> <p>For more information, see Table 4-14.</p>
stream_options	No	None	Object	<p>Specifies whether to display the number of used tokens during streaming output. This parameter is only valid when stream is set to True. You need to set stream_options to {"include_usage": true} to print the number of tokens used. For more information, see Table 4-15.</p>
max_tokens	No	None	Int	<p>Maximum number of tokens that can be generated for the current task, including tokens generated by the model and reasoning tokens for deep thinking.</p>
top_k	No	-1	Int	<p>The candidate set size determines the sampling range during generation. For example, setting it to 50 means only the top 50 scoring tokens are sampled at each step. A larger size increases randomness; a smaller one makes the output more predictable.</p>
top_p	No	1.0	Float	<p>Nucleus sampling. It keeps only the words with combined probabilities above the threshold p and removes the rest. These selected words are then normalized and sampled again.</p> <p>Lower settings reduce word options, making outputs focused and cautious. Higher settings expand word choices, creating varied and creative outputs.</p> <p>Adjust either temperature or top_p separately for best results, not both at once.</p> <p>Value range: 0 to 1. The value 1 indicates that all tokens are considered.</p>

Parameter	Mandatory	Default Value	Type	Description
temperature	No	1.0	Float	<p>Model sampling temperature. The higher the value, the more random the model output; the lower the value, the more deterministic the output.</p> <p>Adjust either temperature or top_p separately for best results, not both at once.</p> <p>Recommended value of temperature: 0.6 for DeepSeek-R1, DeepSeek-V3, and Qwen3 series, and 0.2 for Qwen2.5-VL series.</p>
stop	No	None	None/String/List	<p>A list of strings used to stop generation. The output does not contain the stop strings.</p> <p>For example, if the value is set to ["You," "Good"], text generation will stop once either You or Good is reached.</p>
stream	No	False	Boolean	<p>Controls whether to enable streaming inference. The default value is False, indicating that streaming inference is disabled.</p>
n	No	1	Integer	<p>Number of responses generated for each input message.</p> <ul style="list-style-type: none">• If beam_search is not used, the recommended value range of n is $1 \leq n \leq 10$. If n is greater than 1, ensure that greedy_sample is not used for sampling, that is, top_k is greater than 1 and temperature is greater than 0.• If beam_search is used, the recommended value range of n is $1 < n \leq 10$. If n is 1, the inference request will fail. <p>NOTE</p> <ul style="list-style-type: none">• For optimal performance, keep n at 10 or below. Large values of n can significantly slow down processing. Inadequate video RAM may cause inference requests to fail.• You cannot set n higher than 1 for DeepSeek-R1 and DeepSeek-V3.

Parameter	Mandatory	Default Value	Type	Description
use_beam_search	No	False	Boolean	Controls whether to use beam_search to replace sampling. When this parameter is used, the following parameters must be configured as required: <ul style="list-style-type: none">• n: > 1• top_p: 1.0• top_k: -1• temperature: 0.0 NOTE You cannot set n higher than 1 for DeepSeek-R1 and DeepSeek-V3.
presence_penalty	No	0.0	Float	Applies rewards or penalties based on the presence of new words in the generated text. The value range is [-2.0,2.0].
frequency_penalty	No	0.0	Float	Applies rewards or penalties based on the frequency of each word in the generated text. The value range is [-2.0,2.0].
length_penalty	No	1.0	Float	Imposes a larger penalty on longer sequences in a beam search process. When this parameter is used, the following parameters must be configured as required: <ul style="list-style-type: none">• top_k: -1• use_beam_search: true• best_of: > 1 NOTE You cannot set length_penalty for DeepSeek-R1 and DeepSeek-V3.

Table 4-14 Request parameter **messages**

Parameter	Mandatory	Default Value	Type	Description
role	Yes	None	Str	Different roles correspond to different message types. <ul style="list-style-type: none">• system: developer-entered instructions like response formats and roles for the model to follow.• user: user-entered messages including prompts and context information.• assistant: responses generated by the model.• tool: information returned by the tool when the model calls it.
content	Yes	None	Str	<ul style="list-style-type: none">• When role is set to system, this parameter indicates the AI model's personality. <code>{"role": "system", "content": "You are a helpful AI assistant."}</code>• When role is set to user, this parameter indicates the question asked by the user. <code>{"role": "user", "content": "Which number is larger, 9.11 or 9.8?"}</code>• When role is set to assistant, this parameter indicates the content output by the AI model. <code>{"role": "assistant", "content": "9.11 is larger than 9.8."}</code>• When role is set to tool, this parameter indicates the responses returned by the tool when the model calls it. <code>{"role": "tool", "content": "The weather in Shanghai is sunny today. The temperature is 10°C."}</code>

Table 4-15 Request parameter **stream_options**

Parameter	Mandatory	Default Value	Type	Description
include_usage	No	true	Boolean	Specifies whether the streaming response outputs token usage information. <ul style="list-style-type: none">• true: Each chunk outputs a usage field that shows the total token usage.• false: The token usage is not displayed.

Table 4-16 Response parameters

Parameter	Type	Description
id	String	Unique ID of the request.
object	String	chat.completion type: Multi-turn dialogs are returned.
created	Integer	Timestamp.
model	String	Model name for calling.
choices	Array	Model output, including the index and message parameters. In message : <ul style="list-style-type: none">• content is the model's final reply.• reasoning content is the model's deep thinking content (for DeepSeek models only).
usage	Object	Statistics on tokens consumed by the request: <ul style="list-style-type: none">• This parameter is returned by default for non-streaming requests.• This parameter is returned by default for streaming requests. Each chunk outputs a usage field that shows the token usage. Parameters: <ul style="list-style-type: none">• prompt tokens: number of input tokens.• completion tokens: number of output tokens.• total tokens: total number of tokens.
prompt_logprobs	Float	Log probability. You can use this to measure the model's confidence in its output or to explore other options the model provides.

Text Generation (Non-Streaming) Request Example

- REST API request example:

- Python request example:

```
import requests
import json

if __name__ == '__main__':
    url = "https://api.modelarts-maas.com/v1/chat/completions" # API address
    api_key = "MAAS_API_KEY" # Replace MAAS_API_KEY with the obtained API key.

    # Send a request.
    headers = {
        'Content-Type': 'application/json',
        'Authorization': f'Bearer {api_key}'
    }
    data = {
        "model": "deepseek-v3", # Model name
        "messages": [
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": "Hello"}
        ]
    }
    response = requests.post(url, headers=headers, data=json.dumps(data), verify=False)

    # Print result.
    print(response.status_code)
    print(response.text)
```

- cURL request example

```
curl -X POST "https://api.modelarts-maas.com/v1/chat/completions" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $MAAS_API_KEY" \
-d '{
  "model": "deepseek-v3",
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello"}
  ]
}'
```

- OpenAI SDK request example:

```
from openai import OpenAI

base_url = "https://api.modelarts-maas.com/v1" # API address
api_key = "MAAS_API_KEY" # Replace MAAS_API_KEY with the obtained API key.

client = OpenAI(api_key=api_key, base_url=base_url)

response = client.chat.completions.create(
    model="deepseek-v3", # Model name
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Hello"}
    ]
)

print(response.choices[0].message.content)
```

Text Generation (Streaming) Request Example

- Python request example:

```
from openai import OpenAI

base_url = "https://api.modelarts-maas.com/v1" # API address
api_key = "MAAS_API_KEY" # Replace MAAS_API_KEY with the obtained API key.

client = OpenAI(api_key=api_key, base_url=base_url)
```

```
response = client.chat.completions.create(
    model="deepseek-v3", # Model name
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Hello"}
    ],
    stream = True
)

for chunk in response:
    if not chunk.choices:
        continue

    print(chunk.choices[0].delta.content, end="")
```

- cURL request example:

```
curl -X POST "https://api.modelarts-maas.com/v1/chat/completions" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $MAAS_API_KEY" \
-d '{
  "model": "deepseek-v3",
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello"}
  ],
  "stream": true,
  "stream_options": { "include_usage": true }
}'
```

Image Understanding (Non-Streaming) Request Example

- REST API request example:

- Python request example:

```
import requests
import json
import base64

# Convert the image to the Base64 encoding format.
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode("utf-8")

base64_image = encode_image("test.png")

if __name__ == '__main__':
    url = "https://api.modelarts-maas.com/v1/chat/completions" # API address
    api_key = "MAAS_API_KEY" # Replace MAAS_API_KEY with the obtained API key.

    # Send a request.
    headers = {
        'Content-Type': 'application/json',
        'Authorization': f'Bearer {api_key}'
    }
    data = {
        "model": "qwen2.5-vl-7b", # Model
        "messages": [
            {
                "role": "user",
                "content": [
                    {
                        "type": "text",
                        "text": "Describe the content in the image."
                    },
                    {
                        "type": "image_url",
                        # Ensure the Base64 encoding matches the image/{format} specified in the
                        # Content-Type header listed for supported images. The "f" represents the string formatting
                        # method.
                        # PNG image: f"data:image/png;base64,{base64_image}"
                    }
                ]
            }
        ]
    }
```

```
# JPEG image: f"data:image/jpeg;base64,{base64_image}"
# WEBP image: f"data:image/webp;base64,{base64_image}"
"image_url": {
    "url": f"data:image/png;base64,{base64_image}"
}
}
]
}
response = requests.post(url, headers=headers, data=json.dumps(data), verify=False)

# Print result.
print(response.status_code)
print(response.text)
```

– cURL request example

```
curl -X POST "https://api.modelarts-maas.com/v1/chat/completions" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $MAAS_API_KEY" \
-d '{
  "model": "qwen2.5-vl-72b",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "type": "text", "text": "Describe the content in the image."
        },
        {
          "type": "image_url", "image_url": {
            "url": "data:image/png;base64,$BASE64_IMAGE"
          }
        }
      ]
    }
  ]
}'
```

• OpenAI SDK request example:

```
import base64
from openai import OpenAI

base_url = "https://api.modelarts-maas.com/v1" # API address
api_key = "MAAS_API_KEY" # Replace MAAS_API_KEY with the obtained API key.

# Convert the image to the Base64 encoding format.
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode("utf-8")

base64_image = encode_image("test.png")

client = OpenAI(api_key=api_key, base_url=base_url)

response = client.chat.completions.create(
    model="qwen2.5-vl-72b", # Model
    messages=[
        {
            "role": "user",
            "content": [
                {
                    "type": "text", "text": "Describe the content in the image."
                },
                {
                    "type": "image_url",
                    # Ensure the Base64 encoding matches the image/{format} specified in the Content-Type header listed for supported images. The "f" represents the string formatting method.
                    # PNG image: f"data:image/png;base64,{base64_image}"
                    # JPEG image: f"data:image/jpeg;base64,{base64_image}"
                    # WEBP image: f"data:image/webp;base64,{base64_image}"
                    "image_url": {
                        "url": f"data:image/png;base64,{base64_image}"
                    }
                }
            ]
        }
    ]
)
```

```
)  
print(response.choices[0].message.content)
```

Response Example

```
{  
  "id": "chat-71406e38b0d248c9b284709f8435****",  
  "object": "chat.completion",  
  "created": 1740809549,  
  "model": "DeepSeek-R1",  
  "choices": [  
    {  
      "index": 0,  
      "message": {  
        "role": "assistant",  
        "content": "\n\n Compare 9.11 and 9.8:\n\n1. **Compare the integer part**: The integer part of  
both is 9, which is equal.\n2. **Compare the tenths place**: \n - The tenths place of 9.11 is **1**\n - 9.8 can  
be considered as 9.80, and its tenths place is **8**\n - **8 > 1**, so 9.8 is larger.\n\n**Conclusion**: \n**9.8 >  
9.11**\n(When comparing decimals, line up the digits and compare them directly.)",  
        "reasoning_content": "Well, I now need to compare 9.11 and 9.8 which is larger. First of all, I  
have to recall the method of comparing decimals. When comparing decimals, start by comparing the  
integer parts. If the integer parts are the same, compare the tenths and hundredths of the decimal parts in  
sequence until the larger number is determined. \n\n The integer parts of the two numbers are both 9, so  
they are the same. Next, compare the tenths. The tenth digit of 9.11 is 1, while the tenth digit of 9.8 is 8.  
This can be problematic, as some people might directly treat 9.8 as 9.80, or focus on comparing the digits in  
the tenths place. \n\n Now, comparing the tenths place, 9.8 has an 8, while 9.11 has a 1. Clearly, 8 is  
greater than 1. So, should we conclude that 9.8 is greater than 9.11? \n\n However, it is important to note  
that some people might incorrectly assume that the more decimal places a number has, the larger its value.  
But this is not true; for instance, 0.9 is greater than 0.8999. Thus, having more decimal places does not  
necessarily mean a larger value. \n\n Additionally, the decimal parts of the two numbers can be aligned to  
have the same number of digits for comparison. For instance, 9.8 can be written as 9.80, where the tenths  
place is 8 and the hundredths place is 0. On the other hand, for 9.11, the tenths place is 1 and the  
hundredths place is 1. Since 8 in the tenths place is greater than 1, 9.80 (which is 9.8) is greater than 9.11.  
\n\n Therefore, the final conclusion is that 9.8 is larger than 9.11.\n",  
        "tool_calls": [],  
      },  
      "logprobs": null,  
      "finish_reason": "stop",  
      "stop_reason": null  
    }  
  ],  
  "usage": {  
    "prompt_tokens": 21,  
    "total_tokens": 437,  
    "completion_tokens": 416  
  },  
  "prompt_logprobs": null  
}
```

4.5.2 Obtaining the Model List (Models/GET)

This chapter describes how to query the model list through the **Models** API.

API Information

Table 4-17 API information

Parameter	Description	Example Value
API Host	API URL for calling the model service.	https://api.modelarts-maas.com/v1/models

Creating a Request

- Authentication description
MaaS inference services support API key authentication. The authentication header is in the following format:
`'Authorization': 'Bearer API key of the region where the service is deployed'`
- Response parameters

Table 4-18 Response parameters

Parameter	Type	Description
object	string	Type-list: (The queried information is listed.)
data	Array	Model information of the model service. The main parameters are as follows: <ul style="list-style-type: none">• id: model ID used when calling the API to create a request.• object: model type.• created: creation timestamp.

Example Request

```
import requests

url = "https://api.modelarts-maas.com/v1/models"

headers = {"Authorization": "Bearer yourApiKey"}

response = requests.request("GET", url, headers=headers)

print(response.text)
```

Response Example

```
{
  "object": "list",
  "data": [
    {
      "id": "DeepSeek-R1",
      "object": "model",
      "created": 0,
      "owned_by": ""
    },
    {
      "id": "DeepSeek-V3",
      "object": "model",
      "created": 0,
      "owned_by": ""
    }
  ]
}
```

4.5.3 Error Codes

The table below shows the error codes you might see when calling a model service deployed with MaaS.

Table 4-19 Error codes

HTTP Status Code	Error Code	Error Message	Description
400	ModelArts .81001	Invalid request body.	Parsing the body failed, for example, JSON formatting failed or empty model parameters.
400	ModelArts .81002	Failed to get the authorization header.	The Authorization parameter in the request header is either empty or lacks the Bearer prefix.
401	ModelArts .81003	Invalid authorization header.	Parsing the API key failed.
401	ModelArts .81004	Invalid request because you do not have access to it.	The built-in service is not enabled.
401	ModelArts .81005	The free quota has been used up.	Free quota has been used up.
401	ModelArts .81006	The resource is frozen.	The resident model has been frozen.
401	ModelArts .81109	No permission query task %s	No permission to query the video generation task.
403	ModelArts .81011	May contain sensitive.....	Input or non-streaming output risk control.
404	ModelArts .81009	Invalid model.	The model specified by the model parameter in the request body is not found.
404	ModelArts .81108	Task %s does not exist	Task not found.
429	ModelArts .81101	Too many requests, exceeded rate limit is {rpm} times per minute.	RPM traffic control verification failed.

HTTP Status Code	Error Code	Error Message	Description
429	ModelArts .81103	Too many requests. exceeded rate limit is %s tokens per minute.	TPM traffic control verification failed.
403	ModelArts .81109	No permission query task %s	No permission to query the task.
5XX	APIG.0203	"error_msg":"Backend timeout",error_code:APIG.0203	The service response times out.
400	"object": "error"	"object": "error", "message": "[{'type': 'missing', 'loc': ('body', 'model'), 'msg': 'Field required', 'input': {'max_tokens': 20, 'messages': [{'role': 'system', 'content': 'You are a helpful assistant.'}, {'role': 'user', 'content': 'Hello'}], 'stream': False, 'temperature': 1.0}]]", "type": "BadRequestError", "param": null, "code": 400	Mandatory parameters are missing in the request body.
400	"object": "error"	"object": "error", "message": "[{'type': 'extra_forbidden', 'loc': ('body', 'test'), 'msg': 'Extra inputs are not permitted', 'input': 15}]]", "type": "BadRequestError", "param": null, "code": 400	The request body has extra, unsupported parameters.
400	"object": "error"	"object": "error", "message": "[{'type': 'json_invalid', 'loc': ('body', 273), 'msg': 'JSON decode error', 'input': {}, 'ctx': {'error': '\\Expecting \',\' delimiter\\'}}]", "type": "BadRequestError", "param": null, "code": 400	The request body's JSON format is wrong.

HTTP Status Code	Error Code	Error Message	Description
400	"object": "error"	"object": "error", "message": "[{'type': 'missing', 'loc': ('body',), 'msg': 'Field required', 'input': None}]", "type": "BadRequestError", "param": null, "code": 400	No request body.
400	"object": "error"	"object": "error", "message": "This model's maximum context length is 4096 tokens. However, you requested 8242 tokens (20 in the messages, 8222 in the completion). Please reduce the length of the messages or completion.", "type": "BadRequestError", "param": null, "code": 400	The max_tokens value exceeds the model's upper limit.
404	"object": "error"	"object": "error", "message": "The model `DeepSeek-R1` does not exist.", "type": "NotFoundError", "param": null, "code": 404	The model parameter in the request body is incorrect.
404	APIG.0101	"error_msg": "The API does not exist or has not been published in the environment", "error_code": "APIG.0101", "request_id": "d0ddda0fcdd0cc23a1588fafa426****"	The API URL is wrong or does not exist.
405	N/A	"detail": "Method Not Allowed"	The request method is incorrect.
429	APIG.0308	"error_msg": "The throttling threshold has been reached: policy ip over ratelimit,limit:5,time:1 minute"	The API Gateway traffic limit is reached.

4.6 Creating a Multi-Turn Dialogue in ModelArts Studio (MaaS)

This chapter describes how to use the MaaS chat API to implement multi-turn dialogues.

The MaaS server does not save request details. You must include all conversation history each time you send a new request to the chat API. This example uses Python code. You can change the code as needed.

The following is an example of Python code for context concatenation and request:

```
from openai import OpenAI
client = OpenAI(api_key="MaaS API Key", base_url="https://xxxxxxxxxxxxxxxxx")
# First turn of dialogue
messages = [{"role": "user", "content": "Which number is larger, 9.11 or 9.8?"}]
response = client.chat.completions.create(
    model="DeepSeek-R1",
    messages=messages
)
messages.append(response.choices[0].message)
print(f'Messages Round 1: {messages}')
# Second turn of dialogue
messages.append({"role": "user", "content": "What is the sum of them?"})
response = client.chat.completions.create(
    model="DeepSeek-R1",
    messages=messages
)
messages.append(response.choices[0].message)
print(f'Messages Round 2: {messages}')
```

The messages in the request body for the first turn of the dialog are as follows:

```
[
  {"role": "user", "content": "Which number is larger, 9.11 or 9.8?"}
]
```

The steps for constructing messages in the request body in the second turn of the dialogue are as follows:

1. Add the output content of the model (**role** is **assistant**) in the first turn of the dialogue to the end of messages.
2. Add the new user question to the end of messages.
3. The messages in the request body finally transferred to the chat API are as follows:

```
[
  {"role": "user", "content": "Which number is larger, 9.11 or 9.8?"},
  {"role": "assistant", "content": "9.8 is larger."},
  {"role": "user", "content": "What is the sum of them?"}
]
```

5 ModelArts Studio (MaaS) Management and Statistics

5.1 Managing API Keys in ModelArts Studio (MaaS)

5.1 Managing API Keys in ModelArts Studio (MaaS)

To call a model service in MaaS, enter an API key for secure, authorized access. This chapter describes how to create and delete an API key.

Operation Scenarios

When you use a model service deployed in MaaS for tasks like data requests and model inference, the system checks your API key to verify your identity and access permissions. Only users with valid API keys can access model services, preventing unauthorized use.

- First access: Create an API key to authenticate your identity when you call the model API for the first time.
- Key loss or leakage: If your API key is lost or leaked, create a new key to replace it for security.
- Periodic key rotation: Rotate the key regularly according to the security policy to lower the risk of long-term exposure.

Notes and Constraints

- Each account can have up to 30 API keys. If you exceed this limit, delete old keys before adding new ones.
- After you create an API key, you must save it right away. The system does not store the key as plain text.
- After you delete an API key, it becomes invalid right away.

Prerequisites

- You have the permission to manage API keys.
- You have learned the scenarios and permission requirements for the target model service.

Creating an API Key

You can create a up to 30 keys. Each key is displayed only once after creation. Keep it secure. If the key is lost, it cannot be retrieved. In this case, create a new API key.

1. Log in to **ModelArts Studio (MaaS) console** and select the target region on the top navigation bar.
2. In the navigation pane, choose **API Keys**.
3. On the **API Keys** page, click **Create API Key**, enter the tag and description, and click **OK**.

The tag and description cannot be modified after the key is created.

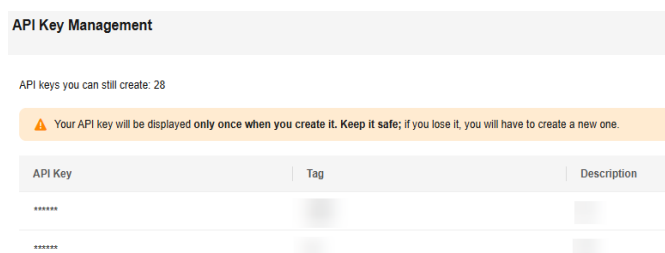
Table 5-1 Parameters

Parameter	Description
Tag	Tag of the API key. The tag must be unique. The tag can contain 1 to 100 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed.
Description	Description of the custom API key. The value can contain 1 to 100 characters.

4. In the **Your Key** dialog box, copy the key and store it securely.
5. After the key is saved, click **Close**.

After you click **Close**, the key cannot be viewed again. Once you create the API key, it appears on the **API Keys** page.

Figure 5-1 API key



Deleting an API Key

If you reach the API key limit or if a key is leaked or lost, delete unused API keys. Deleting the API key will permanently remove it and prevent its use or retrieval.

1. Log in to **ModelArts Studio (MaaS) console** and select the target region on the top navigation bar.
2. In the navigation pane, choose **API Keys**.
3. Click **Delete** on the right of the target API key.
4. In the **Delete API Key** dialog box, click **OK**.

FAQs

1. How long does it take for an API key to become valid after it is created?
An API key becomes valid a few minutes after creation.
2. Can I use an API key across regions?
API keys work only in their specific region. For example, an API key created in the CN-Hong Kong region can only be used in this region.