

# Alibaba Cloud

## Apsara Stack Enterprise

User Guide - Middleware and  
Enterprise Applications

Product Version: 2012, Internal: V3.13.0









Document Version: 20210621

# Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

# Document conventions

Style	Description	Example
 <b>Danger</b>	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 <b>Danger:</b> Resetting will result in the loss of user configuration data.
 <b>Warning</b>	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 <b>Warning:</b> Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 <b>Notice</b>	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 <b>Notice:</b> If the weight is set to 0, the server no longer receives new requests.
 <b>Note</b>	A note indicates supplemental instructions, best practices, tips, and other content.	 <b>Note:</b> You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click <b>Settings &gt; Network &gt; Set network type</b> .
<b>Bold</b>	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click <b>OK</b> .
<code>Courier font</code>	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[ ] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

# Table of Contents

1. Enterprise Distributed Application Service (EDAS)	14
1.1. What is EDAS?	14
1.2. Quick start	15
1.2.1. Get started with EDAS	15
1.2.2. Log on to the EDAS console	16
1.2.3. Deploy microservice applications in a Kubernetes environment	16
1.2.4. Use custom images to deploy multi-language applications	19
1.2.5. Deploy microservice applications in ECS environments	22
1.2.6. How do I use the new application creation process?	24
1.3. Application development	26
1.3.1. Application development overview	26
1.3.2. Use Spring Cloud to develop applications	28
1.3.2.1. Spring Cloud overview	28
1.3.2.2. Implement service registration and discovery	30
1.3.2.3. Implement load balancing	37
1.3.2.4. Implement configuration management	39
1.3.2.5. Build service gateways	44
1.3.2.6. Implement job scheduling	51
1.3.3. Use Dubbo to develop applications	54
1.3.3.1. Dubbo overview	54
1.3.3.2. Use Spring Boot to develop Dubbo microservice applications	56
1.3.4. Use HSF to develop applications	62
1.3.4.1. HSF overview	62
1.3.4.2. Start the light-weight configuration registry	64
1.3.4.3. Use Ali-Tomcat to develop applications	66
1.3.4.3.1. Ali-Tomcat overview	66

1.3.4.3.2. Install Ali-Tomcat and Pandora .....	66
1.3.4.3.3. Configure startup for an IDE runtime environme.....	67
1.3.4.3.3.1. Configure an Eclipse development environme.....	67
1.3.4.3.3.2. Configure the IntelliJ IDEA development envi.....	70
1.3.4.3.4. Develop HSF applications (EDAS-SDK) .....	71
1.3.4.3.4.1. Download a demo project .....	71
1.3.4.3.4.2. Define service interfaces .....	72
1.3.4.3.4.3. Implement services as a provider .....	72
1.3.4.3.4.4. Subscribe to services as a consumer .....	76
1.3.4.3.4.5. Use HSF features .....	79
1.3.4.3.4.6. Query services .....	82
1.3.4.3.5. Migrate Dubbo applications to HSF (not recomm.....	84
1.3.4.3.5.1. Notes for developing Dubbo applications .....	84
1.3.4.3.5.2. Modify Dubbo application configurations .....	84
1.3.4.3.5.3. Convert JAR packages to WAR packages .....	86
1.3.4.3.5.4. Run programs .....	87
1.3.4.3.5.5. Compatibility between Dubbo and HSF .....	87
1.3.4.4. Use Pandora Boot to develop applications .....	90
1.3.4.4.1. Pandora Boot overview .....	90
1.3.4.4.2. Configure the local repository path and the ligh.....	91
1.3.4.4.3. Use Pandora Boot to develop HSF applications .....	93
1.3.4.4.3.1. Example of HSF application development .....	93
1.3.4.4.3.2. Advanced HSF features .....	97
1.3.4.4.3.3. Local debugging .....	101
1.3.4.4.3.4. Publish applications to EDAS .....	103
1.3.4.4.4. Develop RESTful applications (not recommended.....	103
1.3.4.4.4.1. Terms .....	103
1.3.4.4.4.2. Service registration and discovery .....	104

1.3.4.4.4.3. Distributed tracing	112
1.3.4.4.5. Migrate Dubbo applications to HSF (not recommended)	116
1.3.4.5. One call process	117
1.3.4.6. Asynchronous calls	119
1.3.4.7. Generic calls	122
1.3.4.8. Call context	124
1.3.4.9. Select a serialization method	126
1.3.4.10. Configure a timeout period	127
1.3.4.11. Configure a provider thread pool	130
1.3.4.12. API manual	131
1.3.4.13. Set JVM -D startup configuration parameters	142
1.4. Resource management	143
1.4.1. VPC	143
1.4.2. Manage a namespace	144
1.4.3. Manage ECS resources	145
1.4.3.1. Import ECS instances	145
1.4.3.2. Create an ECS cluster	146
1.4.4. Manage Kubernetes resources	147
1.4.4.1. Import Container Service Kubernetes clusters	147
1.4.5. Manage resource groups	148
1.5. Application management for ECS clusters	149
1.5.1. Application deployment	149
1.5.1.1. Deploy applications in the console	149
1.5.1.1.1. Deploy web applications in ECS clusters	149
1.5.1.2. Use CLI to deploy applications	151
1.5.1.2.1. Use toolkit-maven-plugin to automatically deploy...	151
1.5.1.2.2. Use CLI to deploy applications in EDAS	156
1.5.1.2.3. Use Alibaba Cloud Toolkit for Eclipse to deploy ...	158

1.5.1.2.4. Use Alibaba Cloud Toolkit for IntelliJ IDEA to de...	159
1.5.1.3. Canary release for ECS clusters	162
1.5.1.4. Deploy applications in hybrid clouds	164
1.5.2. Lifecycle management for applications in ECS clusters	166
1.5.2.1. Deploy an application (applicable to ECS clusters)	167
1.5.2.2. Scale out and scale in applications (ECS clusters)	169
1.5.2.3. Create branch versions of an application	169
1.5.2.4. Roll back an application	171
1.5.2.5. Delete an application	171
1.5.3. Application settings	171
1.5.3.1. Set JVM parameters	171
1.5.3.2. Configure Tomcat	172
1.5.3.3. Enable the Intra-zone Provider First feature	173
1.5.3.4. Bind an SLB instance to EDAS	174
1.5.3.5. Set JVM -D startup parameters	176
1.5.4. View application changes	177
1.5.5. Log management	178
1.5.6. Perform auto scaling on applications in ECS clusters	179
1.5.7. Throttling and degradation (only applicable to HSF ap...	182
1.5.7.1. Throttling management	183
1.5.7.2. Degradation management	185
1.5.8. Container version management (only applicable to HSF...	186
1.6. Application management for Container Service for Kuberne..	186
1.6.1. Application deployment	187
1.6.1.1. Configure a Kubernetes base image	187
1.6.1.2. Create an application image	190
1.6.1.3. Deploy an application to a Container Service for Ku...	197
1.6.1.4. Canary release for Kubernetes clusters	203

1.6.1.5. Batch release (applicable to Kubernetes clusters) -----	207
1.6.2. Application lifecycle management of Kubernetes cluster...-----	210
1.6.3. View application changes -----	210
1.6.4. View application events -----	211
1.6.5. Logs -----	212
1.6.5.1. View real-time logs -----	212
1.6.5.2. Log directories -----	212
1.6.6. Monitoring -----	214
1.6.6.1. Dashboard -----	214
1.6.6.2. Infrastructure monitoring -----	214
1.6.6.3. Service monitoring -----	216
1.6.6.4. Advanced monitoring -----	217
1.6.6.5. Alerts -----	218
1.6.6.5.1. Create contacts -----	218
1.6.6.5.2. Create and manage a contact group -----	219
1.6.6.5.3. Create an alert -----	220
1.6.6.5.4. Manage alerts -----	230
1.6.6.5.5. Configure a DingTalk chatbot to send alert noti...-----	231
1.6.7. Auto scaling -----	235
1.6.7.1. Auto scaling (applicable to Kubernetes clusters) -----	235
1.7. Configuration management -----	237
1.7.1. Configuration management overview -----	237
1.7.2. Manage configurations -----	238
1.7.3. View historical versions and rollback configurations -----	239
1.7.4. Query listening -----	239
1.8. Microservice governance -----	239
1.8.1. Overview of microservice governance -----	240
1.8.2. Spring Cloud service governance -----	240

1.8.2.1. Query Spring Cloud services .....	241
1.8.2.2. Query Spring Cloud service traces .....	242
1.8.2.3. Ensure Spring Cloud application availability through... ..	242
1.8.2.4. Implement access control of Spring Cloud applicati... ..	245
1.8.3. Dubbo service governance .....	248
1.8.3.1. Query Dubbo services .....	248
1.8.3.2. Query Dubbo service traces .....	249
1.8.3.3. Ensure Dubbo application availability through out-o... ..	249
1.8.3.4. Implement access control of Dubbo applications thr... ..	252
1.8.4. HSF service governance .....	254
1.8.4.1. Query HSF services .....	254
1.8.4.2. Query HSF service traces .....	255
1.8.4.3. View HSF service reports .....	255
1.8.4.4. End-to-end canary release .....	256
1.8.4.4.1. Overview .....	256
1.8.4.4.2. Update a single application by using end-to-en... ..	257
1.8.4.4.3. Troubleshoot application problems by using end... ..	261
1.8.4.4.4. Monitor traffic for an application in a canary re... ..	263
1.8.4.4.5. Restrictions on canary release .....	264
1.8.4.4.6. Canary release policies .....	265
1.8.4.4.7. Parameters of traffic throttling rules .....	266
1.9. Batch operations .....	270
1.10. System management .....	271
1.10.1. Introduction to the EDAS account system .....	271
1.10.2. Manage RAM users .....	272
1.10.2.1. RAM user overview .....	272
1.10.2.2. Create a RAM role .....	272
1.10.2.3. Use a primary account for RAM user operations .....	272

1.10.3. Manage roles	273
1.10.4. View all permissions	273
1.11. FAQ	273
1.11.1. Known issues and solutions	273
1.11.2. Development FAQ	275
1.11.2.1. Ali-Tomcat FAQ	275
1.11.2.2. Lightweight configuration center FAQ	276
1.11.2.3. HSF FAQ	278
1.11.2.4. HSF error codes	279
1.11.2.5. Other development problems	284
1.11.3. Usage FAQ	284
1.11.3.1. Account management	284
1.11.3.2. Resource management	284
1.11.3.3. Application lifecycle	286
1.11.3.4. Troubleshoot EDAS Agent issues	287
1.11.3.5. Troubleshoot change process issues	288
2.API Gateway	290
2.1. What is API Gateway?	290
2.2. Log on to the API Gateway console	290
2.3. Quick start	291
2.3.1. Create an API with HTTP as the backend service	291
2.4. Call an API	297
2.4.1. Manage applications	297
2.4.1.1. Create an app	297
2.4.1.2. View app details	298
2.4.1.3. Edit an app	298
2.4.1.4. Delete an app	298
2.4.2. View created APIs	298

2.4.3. Authorize an application .....	299
2.4.4. Encrypt a signature .....	299
2.4.5. Request signatures .....	299
2.4.6. API call examples .....	302
2.5. APIs .....	304
2.5.1. Manage groups .....	304
2.5.1.1. Create an API group .....	304
2.5.1.2. Manage domain names .....	304
2.5.1.3. Manage certificates .....	305
2.5.1.4. Delete an API group .....	305
2.5.1.5. Manage environments .....	306
2.5.2. Create an API .....	307
2.5.2.1. Overview .....	307
2.5.2.2. Create an API .....	307
2.5.2.3. Security authentication .....	311
2.5.2.4. Configure a network protocol .....	311
2.5.2.5. Configure a request body .....	312
2.5.2.6. Configure an API in Mock mode .....	312
2.5.2.7. Return the Content-Type header .....	313
2.5.3. API management .....	313
2.5.3.1. View and modify an API .....	313
2.5.3.2. Publish an API .....	313
2.5.3.3. Authorize an app .....	314
2.5.3.4. Revoke an authorization .....	315
2.5.3.5. Unpublish an API .....	315
2.5.3.6. View the version history of an API .....	315
2.5.3.7. Change the version of an API .....	316
2.5.4. Plugin management .....	316

2.5.4.1. Use parameters and conditional expressions .....	316
2.5.4.2. Create a plugin .....	323
2.5.4.2.1. Create an IP address-based access control plugin .....	323
2.5.4.2.2. Create a throttling plugin .....	325
2.5.4.2.3. Create a backend signature plugin .....	328
2.5.4.2.4. Create a CORS plugin .....	329
2.5.4.2.5. Create a backend routing plugin .....	331
2.5.4.2.6. Create a caching plugin .....	336
2.5.4.2.7. JWT authentication plugin .....	338
2.5.4.2.8. Access control plugin .....	345
2.5.4.2.9. Error code mapping plugin .....	347
2.5.4.3. Bind a plugin to an API .....	353
2.5.4.4. Delete a plugin .....	354
2.5.4.5. Unbind a plugin .....	354
2.6. Manage monitoring .....	354
2.6.1. View monitoring information and configure alerts .....	355
2.6.2. View statistical information on the dashboard of API G.. .....	358
2.7. Advanced usage .....	359
2.7.1. Customize business parameters for logs .....	359
2.7.2. Configure Log Service logs for API Gateway .....	360
2.7.2.1. Initialize the default Log Service configuration of A... .....	360
2.7.2.2. Configure API Gateway to ship logs to your Log Se... .....	363
2.7.3. Cross-user VPC authorization .....	368
2.7.3.1. User authorization across VPCs .....	368
2.7.3.2. Configure APIs .....	370
2.7.4. Call an API over HTTPS .....	370
2.8. FAQ .....	373
2.8.1. How do I obtain error information? .....	373

2.8.2. Error codes	374
--------------------	-----

# 1. Enterprise Distributed Application Service (EDAS)

## 1.1. What is EDAS?

Enterprise Distributed Application Service (EDAS) is a Platform as a Service (PaaS) platform for application hosting and microservice management, providing full-stack solutions such as application development, deployment, monitoring, and O&M. It supports Dubbo, Spring Cloud, and other microservice runtime environments, helping you easily migrate applications to the cloud.

### Diverse application hosting environments

You can select instance-exclusive Elastic Compute Service (ECS) clusters, Container Service Kubernetes clusters, and user-created Kubernetes clusters based on your application systems and resource needs.

### Abundant microservice frameworks

You can develop applications and services in the native Dubbo, native Spring Cloud, and High-Speed Service Framework (HSF) frameworks, and host the developed applications and services to EDAS.

- You can host Dubbo and Spring Cloud applications to EDAS by adding dependencies and modifying a few configurations. You have access to the features of EDAS, such as enterprise-level application hosting, service governance, monitoring and alerting, and application diagnosis, without having to build ZooKeeper, Eureka, and Consul. This lowers the costs of deployment and O&M.
- HSF is the distributed remote procedure call (RPC) framework that is widely used within Alibaba Group. It interconnects different service systems and decouples inter-system implementation dependencies. HSF unifies the service publishing and call methods for distributed applications to help you conveniently and quickly develop distributed applications. HSF provides or uses common functional modules, and frees developers from various complex technical details involved in distributed architectures, such as remote communication, serialization, performance loss, and the implementation of synchronous and asynchronous calls.

### Comprehensive application management

You can perform end-to-end management, service governance, and microservice management for your applications in the EDAS console.

- Application lifecycle management  
EDAS provides end-to-end application management, allowing you to deploy, scale out, scale in, stop, and delete applications. Applications of all sizes can be managed in the EDAS console.
- Service governance  
EDAS integrates a wide variety of service governance components, such as auto scaling, throttling and degradation, and health check, to deal with unexpected traffic spikes and crashes caused by dependencies. This greatly improves platform stability.
- Microservice management  
EDAS provides the service topology, service report, and trace query features to help you manage every component and service in a distributed system.

### Comprehensive monitoring and diagnosis

You can monitor the status of resources and services in applications in the EDAS console to promptly identify problems and quickly locate their causes through the logging and diagnosis components.

EDAS is connected to the Application Real-Time Monitoring Service (ARMS) to monitor the health status of application resources and services at the Infrastructure as a Service (IaaS) layer in real time, helping you quickly locate problems.

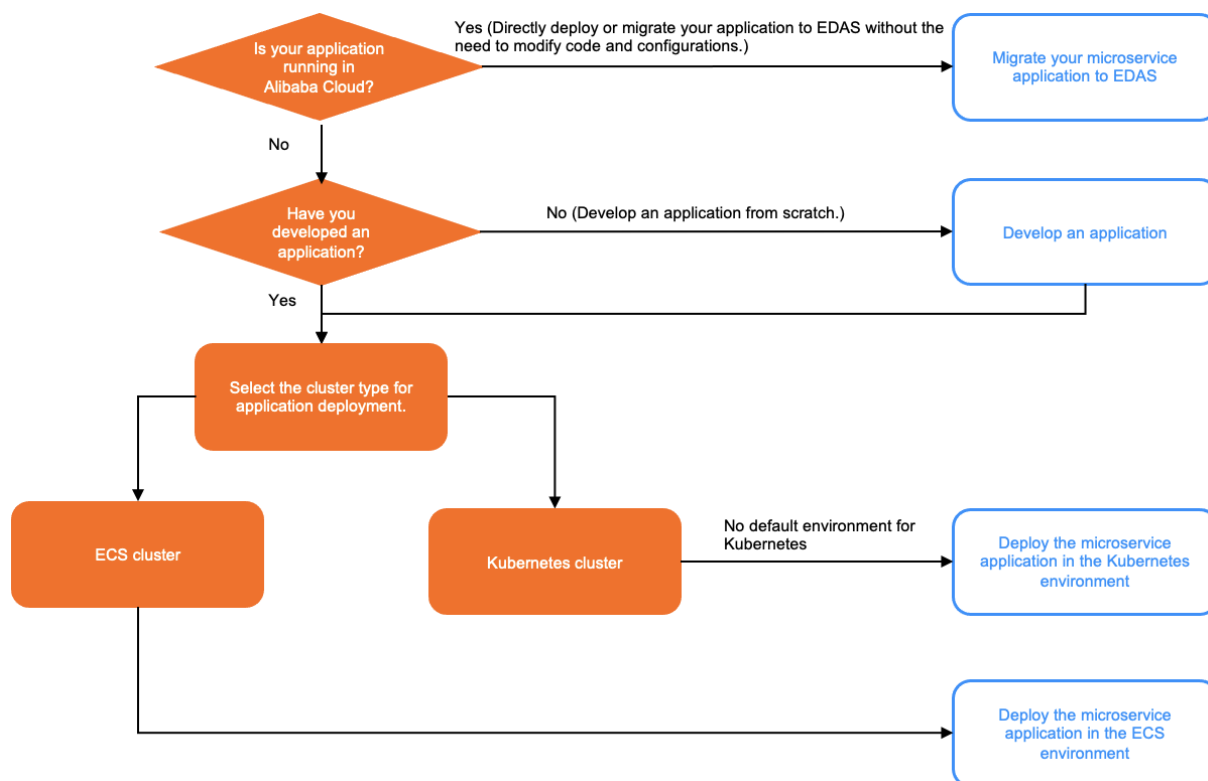
## 1.2. Quick start

This topic describes how to use EDAS to publish a simple web application that only contains a welcome page in Alibaba Cloud Virtual Private Cloud (VPC).

### 1.2.1. Get started with EDAS

To help you experience EDAS, Quick Start describes how to deploy microservice application demos based on Spring Cloud and Apache Dubbo (Dubbo for short) to different EDAS environments. Each deployment scenario can be completed within 30 minutes.

Nacos is used as the registry for the application demo that is provided by EDAS. You can also create a registry another type of registry, such as Eureka or ZooKeeper, based on your requirements. Make sure that the registry and the environment where the applications are to be deployed are connected over a network. If this condition is met, you can deploy your applications to EDAS and use the application hosting, microservices, and cloud native application PaaS platform capabilities of EDAS without modifying code. For more information, see [What is EDAS?](#).



#### ? Note

- **Default environment:** the default ECS cluster in the default VPC and the default namespace that are provided by EDAS. EDAS provides a default environment for only ECS clusters but not for Kubernetes clusters.
- **Custom environment:** the ECS cluster or the Kubernetes cluster that you create.

## 1.2.2. Log on to the EDAS console


This topic describes how to log on to the Enterprise Distributed Application Service (EDAS) console.

### Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

### Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.  
Obtain the username and password that you can use to log on to the console from the operations administrator.

 **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:

- Uppercase or lowercase letters
- Digits
- Special characters, which include ! @ # \$ %

3. Click **Login**.
4. In the top navigation bar of the page, choose **Products > Middleware > Enterprise Distributed Application Service**.
5. On the **EDAS** page, select an **organization** and a **region**, and then click **EDAS**.

## 1.2.3. Deploy microservice applications in a Kubernetes environment

To help you experience how to deploy microservice applications in EDAS, EDAS provides microservice application demos of Spring Cloud, Dubbo, and High-Speed Service Framework (HSF). You can deploy application demos to a specified Kubernetes environment (cluster). This topic uses an example to describe how to deploy a microservice application in a Kubernetes environment.

### Prerequisites

- [Create Docker images for applications](#).

### Context

- **Kubernetes environment**: a Kubernetes cluster that is deployed in a specific region and a specified namespace.
- **Application demo**: EDAS provides the demo image for a pair of microservice applications (a service provider and a service consumer) in the console. You can use the image to perform simple service

calls. In addition, EDAS provides the JAR and WAR packages for a pair of microservice application demos of the Spring Cloud, Dubbo, and HSF frameworks. For more information about how to deploy the JAR or WAR package, see [Deploy an application to a self-managed Container Service for Kubernetes cluster by using WAR or JAR packages](#).

This topic uses the demo image method as an example to describe how to deploy a microservice application. For more information about demos, see [alibabacloud-microservice-demo](#).

- For more information about how to implement the features that are related to microservice applications, see [Application development overview](#).

## Deploy the demo image of a microservice application

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar of the **Applications** page, select the region where you want to deploy the application. On the page that appears, select the namespace that you want to manage and click **Create Application**.
4. On the **Basic Information** tab of the **Create Application** wizard, specify the basic information about the application and click **Next**.

The screenshot shows the 'Create Application' wizard in the EDAS console. The 'Application Information' tab is active. The 'Cluster Type' section has two options: 'ECS Cluster' and 'Kubernetes Cluster'. The 'Application Runtime Environment' section has four options: 'Custom', 'Java', 'Tomcat', and 'EDAS-Container (HSF)'. The 'Custom' option is selected. The 'Next' button is visible at the bottom left.

- i. In the **Cluster Type** section, click **Kubernetes Cluster**.
  - ii. In the **Application Runtime Environment** section, click **Custom**.
5. On the **Application Configuration** tab, configure the application environment, select the demo image, and then click **Next**.

The screenshot shows the 'Create Application' page in the EDAS console. The form is divided into several sections:

- Microservice space:** A dropdown menu with 'edas-demo-project' selected.
- Cluster:** A dropdown menu with 'edas-test-cluster' selected.
- K8s Namespace:** A dropdown menu with 'default' selected.
- Application Name:** A text input field with the placeholder 'Enter an application name'.
- Application Description:** A text input field with the placeholder 'Enter application description'.
- Image Type:** Two radio buttons: 'Configure Image' (selected) and 'Demo Image'.
- Select Image:** A section titled 'Configure Image: edas-demo-project/consumer:1.0'. It includes a dropdown for 'Image Repository' (selected 'edas-demo-project') and a text input for 'Image Repository Name'. Below this is a table of image configurations:

Image Repository Namespace	Image Name	Type	Source	Version
edas-demo-project	edas-demo-project/consumer	PUBLIC	ALI_HUB	1.0
edas-demo-project	edas-demo-project/details-ruby	PUBLIC	ALI_HUB	Select an option
edas-demo-project	edas-demo-project/productpage-python	PUBLIC	ALI_HUB	Select an option
edas-demo-project	edas-demo-project/provider	PUBLIC	ALI_HUB	Select an option
edas-demo-project	edas-demo-project/ratings-js	PUBLIC	ALI_HUB	Select an option

At the bottom of the form, there are sections for 'Total Pods' (set to 1) and 'Single Pod Resource Quota' (CPU Reserved CPU: 0.000, CPU CPU Limit: 0.000, Mem Reserved CPU (MB): 1024, Mem CPU Limit (MB): 0). A note at the bottom states: 'The default values of the CPU and Memory parameters of a single pod are 0, with no specified limits. If you need to specify a limit, enter a specific number. Note: If the CPU of a single pod is not limited (that is, the default value 0 is used), the auto scaling rules configured for the target CPU value will not take effect.'

- i. Select *Default* from the drop-down list on the right of the **Namespace** parameter. This topic helps you experience how to deploy the microservice application demo in a Kubernetes cluster. Therefore, we recommend that you do not create a namespace. If resources and services need to be isolated when you use EDAS in practice, you can create namespaces.
  - ii. Select the Kubernetes cluster that you create and import to EDAS from the drop-down list on the right of the **Cluster** parameter. If the drop-down list does not have a Kubernetes cluster, it indicates that you have not created a Kubernetes cluster or imported this cluster to EDAS. In this case, you must create a Kubernetes cluster and import this cluster to EDAS.
  - iii. Select *default* from the drop-down list on the right of the **K8s Namespace** parameter.
  - iv. In the field on the right of the **Application Name** parameter, enter an application name.
  - v. (Optional) In the field below the **Application Description** parameter, enter the description of the application.
  - vi. Select the repository that you create and upload to the application image repository from the drop-down list on the right of the **Image Repository Namespace** parameter. Then, in the **Configure Image** section, select the image **Project**, image, and image version. If multiple images exist, you can search for the image by image name.
  - vii. In the field on the right of the **Total Pods** parameter, enter *1*.
  - viii. On the right of the **Single Pod Resource Quota** parameter, set **CPU Cores** to *1*, and **Memory (MB)** to *2048*.
6. On the **Advanced Settings** tab, click **Create Application**. The **Advanced Settings** tab includes a series of advanced settings that you can specify based on your actual experience requirements. For more information, see the *Configure advanced settings for application deployment* topic of *Kubernetes User Guide*.
  7. In the **Creation Completed** page, confirm **Basic Information**, **Configurations**, and **Advanced Settings**, and then click **Create Application**.

After the system starts to deploy the application, the message **A change process is ongoing for this application. The application is in Executing state.** is prompted on the top of the **Basic Information** page. It requires about 2 minutes to deploy the application.

You can also click **View Details** next to the prompted message to navigate to the **Change Details** page of the application. On this page, you can check the deployment progress and log data.

## 1.2.4. Use custom images to deploy multi-language applications to Kubernetes clusters

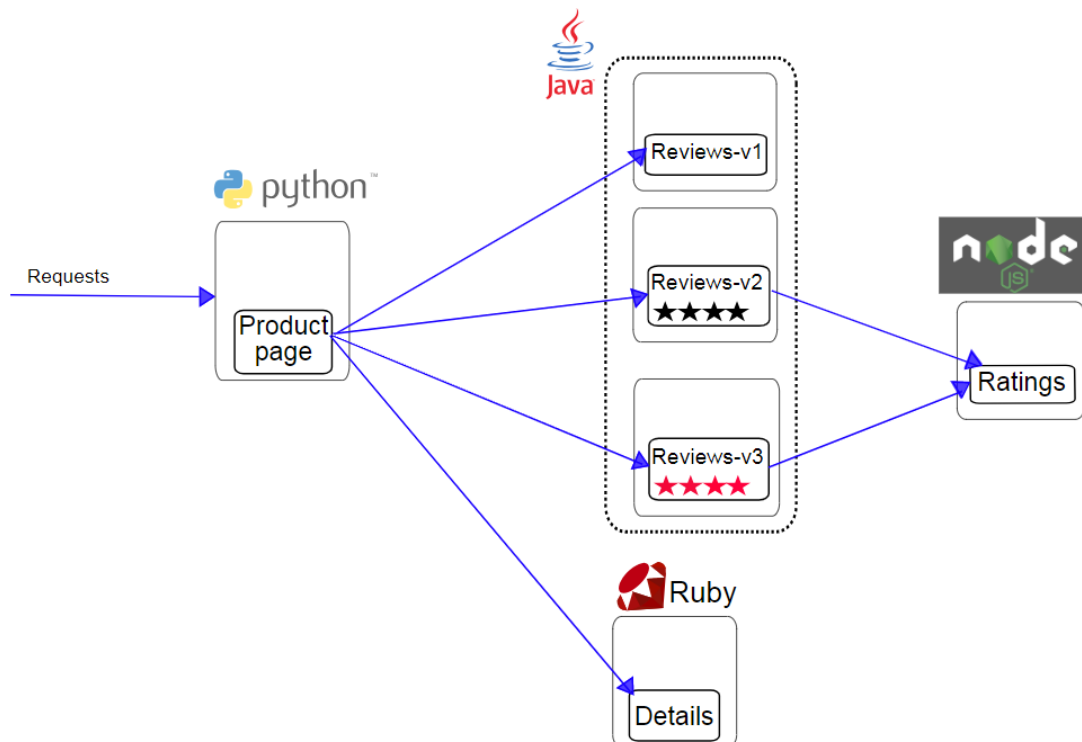
As languages such as Python and Node.js rapidly develop, more and more multi-language microservice-oriented applications are launched. EDAS allows you to deploy multi-language microservice applications through a service mesh. It also provides service governance capabilities such as application hosting, service discovery, distributed tracing, load balancing, and application monitoring. This topic uses an example to describe how to use EDAS to deploy a multi-language microservice application.

### Example scenario description

BookInfo is a sample application that simulates a category of an online bookstore and displays information about a book. The application page displays the description of a book, book details such as the International Standard Book Number (ISBN) and the page number, and some reviews on the book.

BookInfo is a heterogeneous application that consists of several microservice applications. These microservice applications are written in different languages. These microservices constitute a representative service mesh sample. This sample consists of multiple services and languages. In addition, the Reviews service has multiple versions.

Multi-language service architecture



The BookInfo application contains four independent services:

- **Productpage:** a Python service that calls the **Details** and **Reviews** services to generate a page. The


Product page also provides the sign-in and sign-out features.

- Details: a Ruby service that contains book information.
- Reviews: a Java service that contains reviews on the book and calls the Ratings service. The Reviews microservice has the following three versions:
  - Version v1 does not call the Ratings service.
  - Version v2 calls the Ratings service and displays each rating as one to five black stars.
  - Version v3 calls the Ratings service and displays each rating as one to five red stars.
- Ratings: a Node.js service. This service contains rating information that consists of book reviews.

## Prerequisites

- A user-created Kubernetes cluster is imported.
- An image of the sample application is created and uploaded to the Alibaba Cloud image repository. For more information about the operation, see [Create Docker images for applications](#). Address for downloading the sample application: [BookInfo Sample](#).

## Step 1: Install the service mesh for the Container Service for Kubernetes cluster

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Resource Management > Container Service Kubernetes Clusters**.
3. In the top navigation bar, select a region.
4. On the **Container Service Kubernetes Cluster** page, select the **namespace** to which the Container Service for Kubernetes cluster is imported. Click the ID of the imported Container Service for Kubernetes cluster.
5. At the bottom of the cluster details page, click **Install Service Mesh** in the **Service Mesh** section.
6. In the dialog box that appears, click **OK**.  
The dialog box displays **Executing** and disappears. **Installing service mesh** appears at the top of the Cluster Details page. Wait about 1 minute. When **Installing service mesh** disappears, the installation is complete.
7. In the **Service Mesh** section, click the  icon on the right to expand the section and view **Component Version**, **Component Health**, and **Trace Sampling Rate**.

## Step 2: Deploy multi-language applications in the Container Service for Kubernetes cluster

1. Log on to the EDAS console.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the page, select a namespace. In the upper-left corner of the **Applications** page, click **Create Application**.
3. On the **Application Information** tab, specify Cluster Type and Application Runtime Environment, and click **Next**.

Parameter	Description
Cluster Type	Select Kubernetes Cluster.

Parameter	Description
Application Runtime Environment	Select <b>Custom</b> .

4. On the **Application Configuration** tab, configure the environment information, basic information, deployment mode, and image parameters for the application and click **Next**.

Parameter	Description
Namespace	The namespace where the imported Kubernetes cluster is located. If you do not create a namespace or do not select a namespace, the default namespace is selected by default.
Cluster	Select the imported Container Service for Kubernetes cluster from the drop-down list on the right.
K8s Namespace	In this example, <b>default</b> is selected.
Application Name	Enter the application name. The name must start with a letter and can contain a combination of digits, letters, and hyphens (-). The name can be up to 36 characters in length.
Application Description	Enter the description of the application. The description is up to 128 characters in length.
Image Type	Select <b>Configure Image</b> . Then, select the region where the image is located and the image that you want to configure. Set the image version.
Total Pods	Set the number of pods to be deployed for this application.
Single Pod Resource Quota	Enter the CPU and the memory for a pod. To set the quota, enter a specific number. The default value 0 indicates no quota.

5. Configure the advanced settings for the application. After you complete the configuration, click **Create Application**.

i. Configure the service mesh.

Parameter	Description
Protocol	Select a supported protocol from the drop-down list.
Service Name	Enter the service name provided by the application. The service name must be the same as that in the application code to ensure that the service can be registered and called.
Service Port	Enter the service port provided by the application. The service port must be the same as that in the application code to ensure that the service can be registered and called.

ii. (Optional) On the **Advanced Application Settings** tab, you can specify advanced settings based on your requirements. The advanced settings include **Startup Command**, **Environment Variables**, **Persistent Storage**, **Local Storage**, and **Application Lifecycle Management**.

- After the settings are configured, click **Create Application**. On the **Creation Completed** tab, click **Confirm Application Creation**.  
It may take several minutes to create the application. During the creation process, you can click **View Details** at the top of the page to go to the **Change List** page to view the deployment progress and related logs. After the application is created, the **Application Overview** page appears. You can check the running status of the instance pod. If the pod is in the **Running** state, the application is published. You can click the running status of the pod to view the advanced configuration information about the application instance, such as **Deployment**, **Pods**, and **Startup Command**.
- Repeat the preceding steps to deploy the consumer.

## 1.2.5. Deploy microservice applications in ECS environments

To help you experience how to deploy microservice applications in EDAS, EDAS provides microservice application demos of Spring Cloud, Dubbo, and HSF. You can deploy application demos to a ECS environment (specified ECS cluster). This topic uses an example to describe how to deploy microservice applications in a ECS environment.

### Prerequisites

- [Import ECS instances](#)
- [Create an ECS cluster](#)

### Context

- **Application demos:** You can use the microservices-based application demos in the EDAS console to deploy and call simple services. The microservices-based application demos use the Spring Cloud, Dubbo, and HSF frameworks. Each demo contains two applications. The procedures to deploy application demos that use different frameworks are similar. This topic describes how to deploy an application demo that uses the Spring Cloud framework.

For more information about the microservices-based application demos, see [alibabacloud-microservice-demo](#).

- For more information about how to implement the features that are related to microservice applications, see [Application development overview](#).

## Deploy an application demo

Each microservices-based application demo contains a server-side application (service provider) and a client-side application (service consumer). The following example shows how to deploy the server-side application. After you deploy the server-side application, repeat the same steps in this section to deploy the client-side application.

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar of the **Applications** page, select the region where you want to deploy the application. On the page that appears, select the namespace that you want to manage and click **Create Application**.
4. In the **Basic Information** step of the **Create Application** wizard, enter basic application information and click **Next**.
  - i. In the **Cluster Type** section, click **ECS Clusters**.
  - ii. In the **Application Runtime Environment** section, click **Java** and select *Open JDK 8* from the **Java Environment** drop-down list. The Spring Cloud application demo to be deployed in the ECS cluster is a JAR package. Therefore, **Java** is selected as **Application Runtime Environment** in this example. When you deploy an application for production purposes, you can follow the instructions on this page to select a proper application runtime and Java environment.
  - iii. Enter the name of the application in the **Application Name** field.
  - iv. (Optional) Enter the description of the application in the **Application Description** field.
5. On the **Application Configuration** tab, configure the deployment package and the environment for the application and click **Next**.
  - i. Select **Official Demo** as **Source of Deployment Package**.
  - ii. Select *Spring Cloud Server Application* from the **Demo Type** drop-down list.
  - iii. Below the **Environment** parameter, select **Custom Environment**.
  - iv. Select the namespace where the created cluster is located from the drop-down list below the **Namespace** parameter. If you create a cluster in the default namespace, select the default namespace.
  - v. Select the cluster that you create from the drop-down list below the **Cluster** parameter.
  - vi. Select ECS instances that are added to the cluster from the instances below the **Select Instance** parameter.
6. On the **Advanced Settings** tab, specify the **Version**, (Optional) **Application Health Check**, **Batch**, and **Wait Time Before Next Batch** parameters and click **Create Application**.
  - **Version**: By default, EDAS uses the current configured timestamp as the version. The version is in the format of `yyyymmdd:hhmmss`.
  - In this example, the application demo is deployed to only an ECS instance. Therefore, set the **Batch** and **Wait Time Before Next Batch** parameters to *1 Batches* and *Automatic*.
7. In the **Creation Completed** page, confirm **Basic Information**, **Configurations**, and **Advanced**

**Settings**, and then click **Create Application**.

After the system starts to deploy the application, the message **A change process is ongoing for this application. The application is in Executing state.** is prompted on the top of the **Basic Information** page. It requires about 2 minutes to deploy the application.

You can also click **View Details** next to the prompted message to navigate to the **Change Details** page of the application. On this page, you can check the deployment progress and log data.

8. Repeat the preceding steps to deploy the client-side application.

## Verify the result

The client-side application contains web services. After both the server-side and client-side applications are deployed, check whether you can log on to the web interface of the client-side application.

1. In the top navigation bar of the **Applications** page, select the region where the client-side application is deployed. Then, click the name of the client-side application.
2. On the **Basic Information** page of the client-side application, click **Instance Information**.
3. On the **Instance Information** tab, copy the public IP address of the ECS instance where the client-side application is deployed.
4. Open a web browser and enter the connect string of the client-side application into the address bar. The connect string must be in the `http://<Public IP address of the ECS instance>:8080` format. Then, check whether you can log on to the web interface of the client-side application.
5. Enter a string into the **Echo this string** field, for example, `Hello EDAS`. After you enter a string, click **Click here** to check whether a successful response is displayed at the bottom of the page. The **Data Return After Call** section displays the procedure how the client-side application (service consumer) calls the server-side application (service provider). If the string that you entered is returned, this indicates that service call is successful and the microservices-based application is deployed.


```
2020-08-25T10:00:01.866Z : Consumer received.  
2020-08-25T10:00:01.878Z : Provider received.  
    Provider processed after sleep 1 second! Echo String: "Hello EDAS"  
2020-08-25T10:00:02.878Z : Provider Return  
2020-08-25T10:00:02.882Z : Consumer Return
```

## 1.2.6. How do I use the new application creation process?

This topic describes FAQ about the new application creation process. If you encounter any of the problems in the new application creation process, try the following methods.

### How do I select an ECS cluster and a Kubernetes cluster?

The first step to create an application is selecting a cluster type. You can select a suitable cluster type based on the metrics shown in the following table.

 **Note** If you are confused about the cluster type, we recommend that you select Elastic Compute Service (ECS) Cluster.

Cluster type	Deployment mode	Billing	O&M cost	Integration with cloud native	Integration with Apsara DevOps
ECS cluster	One application is deployed on a single ECS instance.	<ul style="list-style-type: none"> <li>IaaS layer: This cluster is billed based on the specifications of the ECS instance.</li> <li>PaaS layer: In Enterprise Distributed Application Service (EDAS), this cluster is billed based on application instances, that is, ECS instances where applications are deployed.</li> </ul>	The O&M is similar to that in the conventional Linux system. It is easy to get started with and perform O&M operations.	N/A	In Apsara DevOps, you can deploy an application to an ECS cluster.
Kubernetes cluster	One application is deployed on a single pod, and multiple pods can run on a single ECS instance.	<ul style="list-style-type: none"> <li>IaaS layer: This is billed based on the used Container Service Kubernetes cluster.</li> <li>PaaS layer: In EDAS, this cluster is billed based on application instances, that is, pods where applications are deployed.</li> </ul>	You need to have a basic understanding of virtualization, container technology, and Kubernetes clusters. It is not easy to get started with and requires high learning costs.	This type of cluster is integrated with the cloud-native Kubernetes environment. One image can be run in multiple regions.	In Apsara DevOps, you cannot create and deploy any application in a Kubernetes cluster.

## Why are there only instance types available for selection in the custom environment?

- **Symptom**  
In the **Application Configuration** wizard, after Custom was selected for Instances, only instance specifications were available for selection on the page. Virtual Private Cloud (VPC) and the specific ECS instance did not appear.
- **Possible causes**  
The current account had not created any VPC, namespace, cluster, or ECS instance in the current region. EDAS will create a default VPC and an ECS instance of the specified type for you, and also create a default namespace and cluster.
- **Solution**  
If you have special requirements for the network, service isolation, and ECS instance, you can create a VPC, a namespace, a cluster, and an ECS instance first. In this way, you can select the desired resources to deploy the application.

## What can I do when failing to import an ECS instance into a cluster?

- **Symptom**  
In the **Application Configuration** wizard, Custom was selected for Instances. In the **Instances** section, **Working with an existing instance** and **Select an instance that is not imported into the cluster** were selected successively. When Create Application was clicked to create the application, an error message appeared, indicating that the ECS instance failed to be imported into the cluster, as shown in the following figure.
- **Possible causes**  
EDAS does not support the operating system of the selected ECS instance.
- **Solution**  
Select another ECS instance, whose operating system must be 64-bit Aliyun Linux 2.1903 or 64-bit CentOS 6.9, 7.2, 7.3, 7.4, or 7.6.

## How do I modify an application port?

- To modify the application port of an application deployed in an ECS cluster, you can do as follows: Click the application name to open the Basic Information tab of the application. Click Settings next to the value of Tomcat Context. Modify the value of Application Port on the Application Settings page that appears.
- To modify the application port of an application deployed in a Kubernetes cluster, currently you can only modify the code or add a Server Load Balancer (SLB) instance to the application.

# 1.3. Application development

## 1.3.1. Application development overview

Enterprise Distributed Application Service (EDAS) supports three microservice application frameworks: Spring Cloud, Dubbo, and High-speed Service Framework (HSF). You can use Spring Cloud, Dubbo, and HSF to develop applications and host them to EDAS.

Application development



### Spring Cloud

Spring Cloud overview
Implement service registration and discovery
Implement load balancing
Implement configuration management
Build service gateways
Implement OSS
Implement job scheduling
Implement throttling and graceful service degradation
Migrate a Spring Cloud cluster (multi-application) to EDAS on the fly

### Dubbo

Dubbo overview
Use Spring Boot to develop Dubbo microservice applications
Use Cloud Toolkit to develop Dubbo microservice application sample projects
Migrate Dubbo applications to EDAS on the fly
User documentation (open source)
Developer guide (open source)
Introduction to source code (open source)

### HSF

HSF overview
Start the lightweight configuration and registry center
Ali-Tomcat overview
Pandora Boot overview
Container release description
Migrate applications developed based on Ali-Tomcat from the HSF architecture to Dubbo

One call process
Asynchronous calls
Generic calls
Call context
Select a serialization method
Configure a timeout period
Configure provider thread pools
API reference
Set JVM -D start up configuration parameters

## 1.3.2. Use Spring Cloud to develop applications

### 1.3.2.1. Spring Cloud overview

Enterprise Distributed Application Service (EDAS) supports the native Spring Cloud microservice framework. You can deploy applications to EDAS to use its enterprise-level features, such as application hosting, microservice governance, monitoring and alerting, and application diagnosis, without the need to modify code or add configurations. The microservice governance includes the service query, canary release, outlier instance removal, and graceful shutdown features. The features of EDAS help you make your applications more stable and secure.

#### Overview

Spring Cloud provides a series of standards and specifications to simplify application development. These standards and specifications cover service discovery, load balancing, circuit breaking, configuration management, Spring Cloud Stream, and Spring Cloud Bus. Based on these standards and specifications, Spring Cloud provides implementation components for service gateways, distributed tracing, security, distributed job scheduling, and distributed job coordination.

Popular Spring Cloud implementation components in the industry include Spring Cloud Netflix, Spring Cloud Consul, Spring Cloud Gateway, and Spring Cloud Sleuth. Spring Cloud Alibaba that is open sourced by the middleware team of Alibaba is also a popular implementation component in the industry.

If you have developed applications by using Spring Cloud components, such as Spring Cloud Netflix and Spring Cloud Consul, you can deploy the applications to EDAS for running and manage the applications in EDAS. In addition, you can use the advanced monitoring features of EDAS, such as distributed tracing, monitoring and alerting, and application diagnosis, without modifying a line of code.

To use more service governance features in EDAS to manage your Spring Cloud applications, you can deploy the applications to EDAS and use its comprehensive microservice governance capabilities, without modifying code and configurations.

#### Compatibility

Currently, Enterprise Distributed Application Service (EDAS) supports Spring Cloud Greenwich, Spring Cloud Finchley, and Spring Cloud Edgware.

The following table lists the Spring Cloud features, open-source components, and compatibility with EDAS.

Spring Cloud feature		Open-source component	Compatibility with EDAS
Common features	Service registration and discovery	<ul style="list-style-type: none"><li>• Netflix Eureka</li><li>• Consul Discovery</li></ul>	Compatible, with an equivalent component
	Load balancing	Netflix Ribbon	Compatible
	Service calls	<ul style="list-style-type: none"><li>• Feign</li><li>• RestTemplate</li></ul>	Compatible
Configuration management		<ul style="list-style-type: none"><li>• Config Server</li><li>• Consul Config</li></ul>	Compatible, with an equivalent component
Gateways		<ul style="list-style-type: none"><li>• Spring Cloud Gateway</li><li>• Netflix Zuul</li></ul>	Compatible
Distributed tracing		Spring Cloud Sleuth	Compatible, with an equivalent component
Message-driven application development: Spring Cloud Stream		<ul style="list-style-type: none"><li>• RabbitMQ binder</li><li>• Kafka binder</li></ul>	Compatible, with an equivalent component
Spring Cloud Bus		<ul style="list-style-type: none"><li>• RabbitMQ</li><li>• Kafka</li></ul>	Compatible, with an equivalent component
Security		Spring Cloud Security	Compatible
Distributed job scheduling		Spring Cloud Task	Compatible
Distributed coordination		Spring Cloud Cluster	Compatible

## Version mappings

The following table describes the mappings among Spring Cloud, Spring Boot, Spring Cloud Alibaba, and the commercially available components that are provided by EDAS. For more information, see [Introduction to versions](#).

Spring Cloud	Spring Boot	Spring Cloud Alibaba	Commercially available components of EDAS
			<ul style="list-style-type: none"> <li>Nacos Registry</li> <li>Nacos Config</li> </ul>
Hoxton	2.2.x	2.2.0 RELEASE	2.2.0 RELEASE
Greenwich	2.1.x	2.1.1.RELEASE	2.1.1.RELEASE
Finchley	2.0.x	2.0.1.RELEASE	2.0.1.RELEASE
Edgware	1.5.x	1.5.1.RELEASE	1.5.1.RELEASE

### 1.3.2.2. Implement service registration and discovery

Enterprise Distributed Application Service (EDAS) provides the general availability (GA) version of the Nacos registry. For the applications developed in Nacos to use this shared registry of EDAS, you do not need to modify the application code. You need only to deploy the applications to EDAS. This topic describes how to develop a pair of Spring Cloud sample microservice applications in an on-premises environment based on Nacos. The pair includes a service provider and a service consumer.

#### Context

You can implement service registration and discovery for your application based on the instructions in this topic. You can also download the application demos: [service-provider](#) and [service-consumer](#).

#### Before you begin

Before you develop an application, make sure that you have completed the following operations:

- Download [Maven](#) and set the environment variables.
- Download the latest version of [Nacos Server](#).
- Start Nacos Server based on the following steps:
  - Decompress the downloaded Nacos Server package.
  - Go to the `nacos/bin` directory to start Nacos Server.
    - For Linux, UNIX, or macOS operating systems, run the `sh startup.sh -m standalone` command.
    - For Windows operating systems, double-click the `startup.cmd` file to run the file.

#### Create a service provider


Create an application project of a service provider in an on-premises environment, add dependencies, enable the service registration and discovery feature, and specify Nacos Server as the registry.

- Create a Maven project that is named `nacos-service-provider`.
- Add dependencies to the `pom.xml` file.  
The following code provides an example on how to add dependencies. Spring Boot 2.1.4.RELEASE and Spring Cloud Greenwich.SR1 are used in this example.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.4.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>2.1.1.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Spring Cloud Greenwich is used in this example and the corresponding Spring Cloud Alibaba version is 2.1.1.RELEASE.

- If you use Spring Cloud Finchley, the corresponding Spring Cloud Alibaba version is 2.0.1.RELEASE.
- If you use Spring Cloud Edgware, the corresponding Spring Cloud Alibaba version is 1.5.1.RELEASE.

 **Note** The Spring Cloud Edgware release has reached the end of life. Therefore, we recommend that you do not use this release to develop applications.

3. In `src/main/java`, create a package that is named `com.aliware.edas`.
4. In the `com.aliware.edas` package, create a startup class that is named `ProviderApplication` for the service provider. and add the following code.  
The `@EnableDiscoveryClient` annotation indicates that the service registration and discovery feature must be enabled for the application.

```
package com.aliware.edas;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication.class, args);
    }
}
```

5. In the `com.aliware.edas` package, create `EchoController`. In `EchoController`, specify `/echo/{string}` as the URL mapping and GET as the HTTP method. Retrieve the method parameter from the URL path, and echo the received parameter.

```
package com.aliware.edas;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class EchoController {
    @RequestMapping(value = "/echo/{string}", method = RequestMethod.GET)
    public String echo(@PathVariable String string) {
        return string;
    }
}
```

6. In the `src/main/resources` path, create a file that is named `application.properties` and add the following configuration to `application.properties` to specify the IP address of Nacos Server.

```
spring.application.name=service-provider
server.port=18081
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

In the preceding configuration, `127.0.0.1` specifies the IP address of Nacos Server. If your Nacos Server is deployed on another machine, you must change the value to the corresponding IP address. If you have other requirements, you can add configurations to the `application.properties` file. For more information, see [Configuration items for reference](#).

7. Verify the result.
- Use the `main` function of `ProviderApplication` in `nacos-service-provider` to start the application.
  - Log on to the on-premises Nacos Server console at `http://127.0.0.1:8848/nacos`. The default username and password of the on-premises Nacos Server console are both `nacos`.
  - In the left-side navigation pane, choose **Service Management** > **Services**. You can view that `service-provider` appears in the service list. You can also query the detailed information about the service in **Details**.

## Create a service consumer

This section describes the service registration feature and explains how Nacos Server works with RestTemplate and FeignClient.

1. Create a Maven project that is named `nacos-service-consumer`.
2. Add dependencies to the `pom.xml` file.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.4.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>2.1.1.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

3. In `src/main/java`, create a package that is named `com.aliware.edas`.
4. In the `com.aliware.edas` package, configure `RestTemplate` and `FeignClient`.

- i. In the `com.aliware.edas` package, create an interface class that is named `EchoService`, add the `@FeignClient` annotation, and then configure the corresponding HTTP URL and HTTP method.

```
package com.aliware.edas;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
@FeignClient(name = "service-provider")
public interface EchoService {
    @RequestMapping(value = "/echo/{str}", method = RequestMethod.GET)
    String echo(@PathVariable("str") String str);
}
```

- ii. In the `com.aliware.edas` package, create a startup class that is named `ConsumerApplication` and add related configurations.

- Use the `@EnableDiscoveryClient` annotation to enable service registration and discovery.
- Use the `@EnableFeignClients` annotation to activate FeignClient.
- Use the `@LoadBalanced` annotation to integrate RestTemplate with service discovery.

```
package com.aliware.edas;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;
@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class ConsumerApplication {
    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
    public static void main(String[] args) {
        SpringApplication.run(ConsumerApplication.class, args);
    }
}
```

5. In the `com.aliware.edas` package, create a class that is named `TestController` to demonstrate and verify the service discovery feature.

```
package com.aliware.edas;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;
@RestController
public class TestController {
    @Autowired
    private RestTemplate restTemplate;
    @Autowired
    private EchoService echoService;
    @RequestMapping(value = "/echo-rest/{str}", method = RequestMethod.GET)
    public String rest(@PathVariable String str) {
        return restTemplate.getForObject("http://service-provider/echo/" + str,
            String.class);
    }
    @RequestMapping(value = "/echo-feign/{str}", method = RequestMethod.GET)
    public String feign(@PathVariable String str) {
        return echoService.echo(str);
    }
}
```

6. In the `src/main/resources` path, create a file that is named `application.properties` and add the following configuration to `application.properties` to specify the IP address of Nacos Server.

```
spring.application.name=service-consumer
server.port=18082
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

7. Verify the result.
- Run the `main` function of `ConsumerApplication` in `nacos-service-consumer` to start the application.
  - Log on to the on-premises Nacos Server console at `http://127.0.0.1:8848/nacos`. The default username and password of the on-premises Nacos Server console are both `nacos`.
  - In the left-side navigation pane, choose **Service Management** > **Services**. You can view that `service-consumer` appears in the service list. You can also query the details about the service in Details.

## Test the result in an on-premises environment

Initiate a call from the consumer to the provider in an on-premises environment and test the result.

- For Linux, UNIX, and macOS operating systems, run the following commands:

```
curl http://127.0.0.1:18082/echo-rest/rest-rest
curl http://127.0.0.1:18082/echo-feign/feign-rest
```

- For Windows operating systems, enter `http://127.0.0.1:18082/echo-rest/rest-rest` and `http://127.0.0.1:18082/echo-feign/feign-rest` in the browser.

## Configuration items for reference

Configuration item	Key	Default value	Description
Server address	spring.cloud.nacos.discovery.server-addr	None	The IP address and the port number on which Nacos Server listens.
Service name	spring.cloud.nacos.discovery.service	\${spring.application.name}	The name of the current service.
Network interface card (NIC) name	spring.cloud.nacos.discovery.network-interface	None	If no IP address is specified, the registered IP address is the IP address of the corresponding NIC. If no NIC is specified, the IP address of the first NIC is used by default.
Registered IP address	spring.cloud.nacos.discovery.ip	None	The highest priority.
Registered port	spring.cloud.nacos.discovery.port	-1	No configuration is required by default. The system automatically discovers the port.
Namespace	spring.cloud.nacos.discovery.namespace	None	Namespaces are widely used to isolate resources in different environments. For example, you can use namespaces to isolate the resources, such as configurations and services, in the development, test, and production environments.
Metadata	spring.cloud.nacos.discovery.metadata	None	Configure this item in the Map format. You can specify an amount of metadata information that is related to the service based on your requirements.
Cluster	spring.cloud.nacos.discovery.cluster-name	DEFAULT	Specify the name of the Nacos cluster.

Configuration item	Key	Default value	Description
Endpoint	spring.cloud.nacos.discovery.endpoint	UTF-8	The domain name of a service in the region. You can dynamically retrieve the server address based on this domain name. You do not need to specify this configuration item when you deploy the service to EDAS.
Integrate with Ribbon	ribbon.nacos.enabled	true	Modify this configuration item only when you have actual needs.

For more information about Spring Cloud Alibaba Nacos Discovery, see the open source version of [Nacos Discovery](#).

### 1.3.2.3. Implement load balancing

Spring Cloud uses Ribbon components to balance loads. Ribbon provides load balancing algorithms for software at clients. In Spring Cloud, the underlying load balancing for RestTemplate and FeignClient is implemented by Ribbon.

Spring Cloud Alibaba ANS integrates the features of Ribbon, and AnsServerList implements the com.netflix.loadbalancer.ServerList API operation that is provided by Ribbon.

This API operation is for generic use. Other similar service discovery components, such as Nacos, Eureka, Consul, and ZooKeeper, also implement the corresponding ServerList API operations, such as NacosServerList, DomainExtractingServerList, ConsulServerList, and ZookeeperServerList.

If you implement the com.netflix.loadbalancer.ServerList API operation, you comply with the load balancing specification of Spring Cloud, because the specification is also for generic use. This indicates that no code modification is required to implement load balancing if you change the service discovery solution from Eureka, Consul, or ZooKeeper to Spring Cloud Alibaba. This is true for RestTemplate, FeignClient, and the outdated AsyncRestTemplate.

The following sections describe how to implement load balancing for RestTemplate and FeignClient in your application.

This topic describes key information about how to develop applications in on-premises environments. For more information about Spring Cloud, download [service-provider](#) and [service-consumer](#).

The methods to implement load balancing for RestTemplate and FeignClient are different and are separately described in the following sections.

#### RestTemplate

RestTemplate is a client that is provided by Spring Cloud to access RESTful services. It provides multiple convenient methods to access remote HTTP services. This makes code compiling at clients more efficient.

To use load balancing of RestTemplate, modify the code in your application based on the following example:

```
public class MyApp {
    // Inject the RestTemplate that is built through the @LoadBalanced annotation.
    // This annotation adds the following interceptor to RestTemplate: LoadBalancerInterceptor.
    // LoadBalancerInterceptor internally uses the implementation class RibbonLoadBalancerClient of the LoadBalancerClient API operation to balance loads.
    @Autowired
    private RestTemplate restTemplate;
    @LoadBalanced // Modify the RestTemplate that is built through the @LoadBalanced annotation to enable the load balancing feature.
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
    // RestTemplate internally uses the load balancing feature to call services. public void doSomething() {
    Foo foo = restTemplate.getForObject("http://service-provider/query", Foo.class);
    doWithFoo(foo);
    }
    ...
}
```

## Feign

Feign is an HTTP client that is implemented in Java to simplify RESTful API calls. To implement load balancing on Feign, perform the following steps:

1. To enable load balancing on Feign, add the Ribbon dependency.

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
<version>{version}</version>
</dependency>
```

2. Use `@EnableFeignClients` and `@FeignClient` to initiate a load balancing request.

- i. Use `@EnableFeignClients` to enable the features of Feign.

```
@SpringBootApplication
@EnableFeignClients // Enable the features of Feign. public class MyApplication {
    ...
}
```

- ii. Use `@FeignClient` to build a `FeignClient` instance.

```
@FeignClient(name = "service-provider")
public interface EchoService {
    @RequestMapping(value = "/echo/{str}", method = RequestMethod.GET)
    String echo(@PathVariable("str") String str);
}
```

- iii. Inject EchoService and call the echo method.  
When you call the echo method, an HTTP request is initiated.

```
public class MyService {  
    @Autowired // Inject the EchoService that is built based on @FeignClient.  
    private EchoService echoService;  
    public void doSomething() {  
        // This is equivalent to initiating a request to http://service-provider/echo/test. echoService.echo(  
        "test");  
    }  
    ...  
}
```

## Verify the result


After `service-consumer` and multiple `service-provider` are enabled, access the URL that is provided by `service-consumer` to check whether load balancing is implemented.

- RestTemplate  
Access `/echo-rest/rest-test` multiple times and check whether the requests are forwarded to different instances.
- Feign  
Access `/echo-feign/feign-test` multiple times and check whether the requests are forwarded to different instances.

## 1.3.2.4. Implement configuration management

When you develop Spring Cloud applications, you can use Nacos (<https://nacos.io>) to manage application configurations in on-premises environments. Nacos is an open source version of Application Configuration Management (ACM). After you deploy an application to Enterprise Distributed Application Service (EDAS), you can manage and push application configurations by using the ACM service that is integrated with EDAS.


You can follow the instructions in this topic to develop a sample application from scratch and use Spring Cloud Alibaba Nacos Config to manage configurations. You can also download the [nacos-config-example](#) demo of the sample application.

 **Note** Spring Cloud Alibaba Nacos Config integrates Nacos and Spring Cloud frameworks and supports the configuration injection specification of Spring Cloud.

## Before you begin

Before you develop an application, make sure that you have performed the following operations:

- Download [Maven](#) and set the environment variables.
- Download the latest version of [Nacos Server](#).
- Start Nacos Server.
  - i. Decompress the downloaded Nacos Server package.
  - ii. Go to the `nacos/bin` directory to start Nacos Server.
    - For Linux, UNIX, or macOS operating systems, run the `sh startup.sh -m standalone` command.
    - For Windows operating systems, double-click the `startup.cmd` file to run the file.
- Create a configuration in the on-premises Nacos Server console.

- i. Log on to the on-premises **Nacos Server console**. The default username and password are both **nacos**.
- ii. In the left-side navigation pane, click **Configurations**. On the **Configurations** page, click the Create configuration icon  in the upper-right corner.
- iii. On the **Create configuration** page, specify the following information and click **Publish**.
  - **Data ID**: `nacos-config-example.properties`
  - **Group**: `DEFAULT_GROUP`
  - **Configuration content**: `test.name=nacos-config-test`


## Use Nacos Config to implement configuration management

1. Create a Maven project that is named `nacos-config-example`.
2. Add dependencies to the `pom.xml` file.  
Spring Boot 2.1.4.RELEASE and Spring Cloud Greenwich.SR1 are used in this example.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.4.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
    <version>2.1.1.RELEASE</version>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Spring Cloud Greenwich is used in this example and the corresponding Spring Cloud Alibaba version is 2.1.1.RELEASE.

- If you use Spring Cloud Finchley, the corresponding Spring Cloud Alibaba version is 2.0.1.RELEASE.
- If you use Spring Cloud Edgware, the corresponding Spring Cloud Alibaba version is 1.5.1.RELEASE.

 **Note** The Spring Cloud Edgware release has reached the end of life. Therefore, we recommend that you do not use this release to develop applications.

3. In `src/main/java`, create a package that is named `com.aliware.edas`.
4. In the `com.aliware.edas` package, create a startup class that is named `NacosConfigExampleApplication` for `nacos-config-example`.

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class NacosConfigExampleApplication {
    public static void main(String[] args) {
        SpringApplication.run(NacosConfigExampleApplication.class, args);
    }
}
```

5. In the `com.aliware.edas` package, create a simple controller that is named `EchoController`. The `userName` property is automatically injected. Then, add the `@Value` annotation to obtain the value of the `test.name` key from the configuration.

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RefreshScope
public class EchoController {
    @Value("${test.name}")
    private String userName;
    @RequestMapping(value = "/")
    public String echo() {
        return userName;
    }
}
```

6. In the `src/main/resources` path, create a configuration file that is named `bootstrap.properties`, and add the following configuration in `bootstrap.properties` to specify the IP address of Nacos Server.

In the following configuration, `127.0.0.1:8848` specifies the IP address of Nacos Server, and `18081` is the service port.

If your Nacos Server is deployed on another machine, change the IP address and service port to the corresponding ones. If you have other requirements, add configuration items in the `bootstrap.properties` file. For more information, see [Configuration items for reference](#).

```
spring.application.name=nacos-config-example
server.port=18081
spring.cloud.nacos.config.server-addr=127.0.0.1:8848
```

7. Implement the main function in `NacosConfigExampleApplication` to start the application.

## Verify the result in an on-premises environment

Visit <http://127.0.0.1:18081> in your browser. The value `nacos-config-test` is returned. This value is the value of `test.name` that you have configured in the on-premises Nacos Server console. For more information, see [Create a configuration in the on-premises Nacos Server console](#).

## Deploy an application to EDAS

After you develop and test your application in an on-premises environment, you can create it as a package and deploy it to EDAS. You can deploy your Spring Cloud application to an Elastic Compute Service (ECS) cluster or a Container Service for Kubernetes (ACK) cluster based on your needs. For more information about how to deploy an application, see [Deploy web applications in ECS clusters](#).

The ACM service that is integrated with EDAS is the general availability (GA) version of Nacos. When you deploy an application to EDAS, EDAS sets the following information based on the priority setting: the IP address and service port of Nacos Server, namespace, AccessKey ID, AccessKey secret, and context path.

Before you deploy an application, you must add the same configuration on the Configuration Management page of the EDAS console as that on the on-premises Nacos Server. To do so, perform the following steps:

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management > Configuration Management > Configurations**.
3. On the **Configurations** page, select a **region** and a **namespace**, and click **Create configuration**.
4. In the **Create configuration** panel, set **Data ID**, **Group**, and **Configuration content**, and then click **Create**.
  - **Data ID**: `nacos-config-example.properties`
  - **Group**: `DEFAULT_GROUP`
  - **Configuration content**: `test.name=nacos-config-test`

## Verify the result

1. After an application is deployed, check the log to see whether the application is started.
2. Run the `curl http://<application instance IP address>:<service port>` command, such as `curl http://192.168.0.34:8080`, to check whether `nacos-config-test` is returned.
3. In the EDAS console, change the previous configuration content to `nacos-config-test2`, and then run the `curl http://<application instance IP address>:<service port>` command, such as `curl http://192.168.0.34:8080`, to check whether `nacos-config-test2` is returned.

## Configuration items for reference

If you have other requirements, see the following table and add configurations in the `bootstrap.properties` file.

Configuration item	key	Default value	Description
Server address	<code>spring.cloud.nacos.config.server-addr</code>	None	None
Data ID prefix	<code>spring.cloud.nacos.config.prefix</code>	<code>\${spring.application.name}</code>	The prefix of the data ID.
Group	<code>spring.cloud.nacos.config.group</code>	<code>DEFAULT_GROUP</code>	

Configuration item	key	Default value	Description
Data ID suffix and content file format	spring.cloud.nacos.config.file-extension	properties	The suffix of the data ID. This configuration item also specifies the file format of the configuration content. The default value is properties, and yaml and yml are also supported.
Encoding mode of the configuration content	spring.cloud.nacos.config.encode	UTF-8	The encoding mode of the configuration.
Timeout period for retrieving the configuration	spring.cloud.nacos.config.timeout	3000	Unit: ms.
Namespace of the configuration	spring.cloud.nacos.config.namespace		Namespaces are widely used to isolate the configurations in different environments. For example, you can use namespaces to isolate the resources in development, test, and production environments.
Relative path	spring.cloud.nacos.config.context-path		The relative path of the server API.
Endpoint	spring.cloud.nacos.config.endpoint	UTF-8	The domain name of a service in the region. You can dynamically retrieve the server address based on this domain name.
Enable listener and auto-refresh	spring.cloud.nacos.config.refresh.enabled	true	The default value is true. You do not need to modify this configuration item.

For more information about configuration items, see the open source [Spring Cloud Alibaba Nacos Config documentation](#).

### 1.3.2.5. Build service gateways

This topic describes how to use Nacos to build service gateways from scratch based on Spring Cloud Gateway and Spring Cloud Netflix Zuul.

#### Why use the EDAS registry for service gateways?

The Enterprise Distributed Application Service (EDAS) registry provides `Spring Cloud Alibaba Nacos Discovery`, which is the general availability (GA) version of the open source Nacos Server. You can directly use the GA version of the EDAS registry for applications that are developed based on Spring Cloud Alibaba Nacos Discovery.

The GA version of the EDAS registry has the following advantages over open source Nacos, Eureka, and Consul:

- Components are shared. This reduces the costs of deploying, managing, and maintaining Nacos, Eureka, or Consul.
- Links are encrypted for calls during service registration and discovery. This protects your services from being discovered by unauthorized applications.
- The EDAS registry is tightly integrated with other EDAS components to provide you with a complete set of microservice solutions, including environment isolation, smooth connection and disconnection, and phased release.

## Before you begin

- Download `Maven` and set the environment variables. Skip this step if you have installed Maven.
- Download the latest version of `Nacos Server`. Skip this step if you have installed Nacos Server.
- Start Nacos Server.  
Decompress the downloaded Nacos Server package and go to the `nacos/bin` directory. For Linux, UNIX, or macOS operating systems, run the `sh startup.sh -m standalone` command. For Windows operating systems, run the `cmd startup.cmd` command or double-click `startup.cmd` to run the file.
- Use demos.  
This topic describes key information about how to develop applications in an on-premises environment. For more information about Spring Cloud, download `spring-cloud-gateway-nacos`, `spring-cloud-zuul-nacos`, and `nacos-service-provider`.

## Build a service gateway based on Spring Cloud Gateway

This section describes how to use Nacos to build service gateways based on Spring Cloud Gateway for applications from scratch.

1. Create a service gateway.
  - i. Create a Maven project that is named `spring-cloud-gateway-nacos`.

- ii. In the `pom.xml` file, add the dependencies of Spring Boot and Spring Cloud. Spring Boot 2.1.4.RELEASE and Spring Cloud Greenwich.SR1 are used in this example.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.4.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
  </dependency>
  <dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>2.1.1.RELEASE</version>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

- iii. Develop a service gateway startup class that is named `GatewayApplication`.

```
@SpringBootApplication
@EnableDiscoveryClient
public class GatewayApplication {
    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }
}
```

- iv. To specify the address of Nacos Server as the registry address, add the following configuration in `application.yaml` .
- In the following configuration, `127.0.0.1:8848` specifies the IP address of Nacos Server. If your Nacos Server is deployed on another machine, change the value to the corresponding address. The `routes` setting specifies the routing and forwarding rules of the gateway. In this example, all requests prefixed with `/provider1/` are routed to the `service-provider` backend service.

```
server:
  port: 15012
  spring:
    application:
      name: spring-cloud-gateway-nacos
    cloud:
      gateway: # config the routes for gateway
      routes:
        - id: service-provider      # Forwards requests that are prefixed with /provider1/ to provider1.
          uri: lb://service-provider
          predicates:
            - Path=/provider1/**
          filters:
            - StripPrefix=1        # Indicates that nacos in the prefix /provider1 must be truncated.
      discovery:
        server-addr: 127.0.0.1:8848
```

- v. Run the main function of the `GatewayApplication` startup class to start the gateway.
- vi. Log on to the on-premises Nacos Server console at `http://127.0.0.1:8848/nacos`. The default username and password are both `nacos`. In the left-side navigation pane, choose **Service Management** > **Services**. You can find that `spring-cloud-gateway-nacos` appears in the service list. You can also query the detailed information about the service in Details. This indicates that the gateway has been started and registered. Then, create a downstream service to verify the request forwarding feature of the gateway.

2. Create a service provider.

Create a service provider application.

The following code provides an example on how to create a service provider:

```
@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication, args);
    }
}

@RestController
public class EchoController {
    @RequestMapping(value = "/echo/{string}", method = RequestMethod.GET)
    public String echo(@PathVariable String string) {
        return string;
    }
}
}
```

3. Verify the result.

- Verify the result in an on-premises environment.

Start the developed service gateway and service provider in an on-premises environment, and access Spring Cloud Gateway to forward the request to the backend service. You can see that the result returned indicates a successful call.

```
→ spring-cloud-gateway-nacos curl http://127.0.0.1:18012/provider1/echo/123456123456%
```

- o Verify the result in EDAS.

Deploy your application to EDAS and verify the result.

The EDAS registry provides a formal GA version of Nacos Server. When you deploy an application to EDAS, EDAS sets the following information based on the priority setting: the IP address and the service port of Nacos Server, namespace, AccessKey ID, AccessKey secret, and context path. You do not need to make additional configurations. You can retain or delete the original configurations.

## Build a service gateway based on Zuul

This section describes how to use Nacos as the registry to build service gateways based on Zuul for applications from scratch.

### 1. Create a service gateway.

- Create a Maven project that is named `spring-cloud-zuul-nacos`.
- Add the dependencies of Spring Boot, Spring Cloud, and Spring Cloud Alibaba in the `pom.xml` file.  
Add the dependencies of Spring Boot 2.1.4.RELEASE, Spring Cloud Greenwich.SR1, and Spring Cloud Alibaba 0.9.0.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.4.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
  </dependency>
  <dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>2.1.1.RELEASE</version>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Greenwich.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

- iii. Develop a service gateway startup class that is named `ZuulApplication` .

```
@SpringBootApplication
@EnableZuulProxy
@EnableDiscoveryClient
public class ZuulApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZuulApplication.class, args);
    }
}
```

- iv. Add the following configuration in `application.properties` to specify the address of Nacos Server as the registry address.  
In the following configuration, `127.0.0.1:8848` specifies the IP address of Nacos Server. If your Nacos Server is deployed on another machine, change the value to the corresponding address. The `routes` setting specifies the routing and forwarding rules of Zuul. In this example, all requests that are prefixed with `/provider1/` are routed to the `service-provider` backend service.

```
spring.application.name=spring-cloud-zuul-nacos
server.port=18022
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
zuul.routes.opensource-provider1.path=/provider1/**
zuul.routes.opensource-provider1.serviceId=service-provider
```

- v. Run the main function of `ZuulApplication` in `spring-cloud-zuul-nacos` to start the service.
  - vi. Log on to the on-premises Nacos Server console at `http://127.0.0.1:8848/nacos`. The default username and password are both `nacos`. In the left-side navigation pane, choose `Service Management > Services`. You can see that `spring-cloud-zuul-nacos` appears in the service list. You can also query the details about the service in `Details`. This indicates that the gateway has been started and registered. Then, create a downstream service to verify the request forwarding feature of the gateway.
2. Create a service provider. For more information about how to create a service provider, see [Implement service registration and discovery](#).

The following code provides an example on how to create a service provider startup class:

```
@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {
    public static void main(String[] args) {
        SpringApplication.run(ProviderApplication, args);
    }
    @RestController
    public class EchoController {
        @RequestMapping(value = "/echo/{string}", method = RequestMethod.GET)
        public String echo(@PathVariable String string) {
            return string;
        }
    }
}
```

3. Verify the result.
  - o Verify the result in an on-premises environment.  
Start the developed Zuul service gateway and service provider in an on-premises environment, and access Spring Cloud Netflix Zuul to forward the request to the backend service. You can see that the result returned indicates a successful call.

```
➔ spring-cloud-gateway-nacos curl http://127.0.0.1:18022/provider1/echo/123456
123456%
➔ spring-cloud-gateway-nacos
```

- o Verify the result in EDAS.  
Deploy your application to EDAS and verify the result. For more information, see [Deploy microservice applications in custom ECS environments](#).

The EDAS registry provides a formal GA version of Nacos Server. When you deploy an application to EDAS, EDAS sets the following information based on the priority setting: the IP address and the service port of Nacos Server, namespace, AccessKey ID, AccessKey secret, and context path. You do not need to make additional configurations. You can retain or delete the original configurations.

### 1.3.2.6. Implement job scheduling

Enterprise Distributed Application Service (EDAS) integrates SchedulerX into the console as a component to schedule jobs. This topic describes how to use SchedulerX to implement job scheduling in Spring Cloud applications. This topic also describes how to deploy these applications to EDAS in the *test* region and implement the job scheduling feature in *Simple Job Single-instance Edition*.

#### Context

SchedulerX is a distributed job scheduling service that is developed by Alibaba. It provides a scheduling service that is accurate, highly reliable, and highly available within seconds based on Cron expressions. It also provides models for implementing distributed jobs, such as grid jobs.

#### Procedure

1. Create a Maven project that is named `scx-example`.
2. In the following example, the dependencies of *Spring Boot 2.0.6.RELEASE* and *Spring Cloud Finchley.SR1* are added in the `pom.xml` file:

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.6.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-alicloud-schedulerx</artifactId>
    <version>0.2.1.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Finchley.SR1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

 **Note**

- If you need to use *Spring Boot 1.x*, use *Spring Boot 1.5.x* and *Spring Cloud Edgware*. The corresponding version of **Spring Cloud Alibaba** is *0.1.1.RELEASE*.
- *Spring Boot 1.x* expired in **August 2019**. Therefore, we recommend that you use the latest version of Spring Boot to develop your applications.

3. Create a startup class that is named `ScxApplication` for `scx-example`.

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class ScxApplication {
    public static void main(String[] args) {
        SpringApplication.run(ScxApplication.class, args);
    }
    ...
}
```

4. Create a simple class that is named `TestService` and use Spring to inject the class to the Inversion of Control (IoC) test job.

```
import org.springframework.stereotype.Service;
@Service
public class TestService {
    public void test() {
        System.out.println("-----IOC Success-----");
    }
}
```


5. Create a simple class that is named `SimpleTask` as the test job class and inject `TestService` to the class.

```
import com.alibaba.edas.schedulerx.ProcessResult;
import com.alibaba.edas.schedulerx.ScxSimpleJobContext;
import com.alibaba.edas.schedulerx.ScxSimpleJobProcessor;
import org.springframework.beans.factory.annotation.Autowired;
public class SimpleTask implements ScxSimpleJobProcessor {
    @Autowired
    private TestService testService;
    @Override
    public ProcessResult process(ScxSimpleJobContext context) {
        System.out.println("-----Hello world-----");
        testService.test();
        ProcessResult processResult = new ProcessResult(true);
        return processResult;
    }
}
```

6. Create a scheduled job and add the configuration.
  - i. Log on to the EDAS console. In the **test** region, create a scheduled job group and record the group ID.

- ii. In the job group that you created, create a scheduled job based on the following parameters:
  - **Job Group:** Select the *ID of the group* that you created in the *test* region.
  - **Job Processing Interface:** The full name of the class that implements the job processing interface. In this example, the value is *SimpleTask*, which is the same as the test job class in the application.
  - **Type:** *Simple Job Single-instance Edition*.
  - **Cron Expression:** `*0 * * * * ? *` is selected by default. It indicates that the job is executed once every minute.
  - **Job Description:** None.
  - **Custom Parameters:** None.
- iii. In the `src/main/resources` path of the local Maven project, create the `application.properties` file and add the following configuration to the file:

```
server.port=18033
# Specify the region (**regionName** of the test region is *cn-test*) and group ID (group-id) of the job.
spring.cloud.alicloud.scx.group-id=***
spring.cloud.alicloud.edas.namespace=cn-test
```

 **Note** In this topic, the test region is used and the test is performed over the Internet. You can verify the deployment result in an on-premises environment or in the cloud. No permission limits are imposed. If you want to deploy applications to other regions, such as *China (Hangzhou)*, you must perform the following steps when you create a scheduled job and schedule the job:

- i. Log on to the EDAS console. In the *China (Hangzhou)* region, create a job group and a scheduled job.
- ii. Go to the Security Management page and obtain the **AccessKey ID** and the **AccessKey secret**.
- iii. Configure the scheduled job in the `application.properties` file.
- iv. In the `application.properties` file, add the **AccessKey ID** and the **AccessKey secret** of your Apsara Stack tenant account.

```
spring.cloud.alicloud.access-key=xxxxx
spring.cloud.alicloud.secret-key=xxxxx
```

7. Run the main function in `ScxApplication` to start the service.

## Result

View the standard output in the IntelliJ IDEA console. You can find that the following test information is periodically displayed:

```
-----Hello world-----
-----IOC Success-----
```

## What's next

After your application is deployed to EDAS, you can use the SchedulerX component to implement more job scheduling features.

## 1.3.3. Use Dubbo to develop applications

### 1.3.3.1. Dubbo overview

Enterprise Distributed Application Service (EDAS) supports the Apache Dubbo (Dubbo) microservices framework. You can deploy Dubbo microservices to EDAS by adding dependencies and modifying configurations, without interfering with code. Then, you have access to the features of EDAS, such as hosting of enterprise-level microservice-oriented applications, microservice governance, monitoring and alerting, and application diagnostics.

#### Architecture

Two mainstream versions of open source Dubbo are available: 2.6.x and 2.7.x.

- Dubbo 2.6.x is widely used and will continue to be maintained, but will not be upgraded with new features.
- Dubbo 2.7.x is the latest version of Dubbo and will be upgraded with new features.

We recommend that you use Dubbo 2.7.x. If you are using Dubbo 2.6.x, we recommend that you migrate to Dubbo 2.7.x to use future new features.

The following process shows the workflow of the Dubbo microservices framework:

1. During startup, the provider registers with the registry.
2. During startup, the consumer subscribes to services from the registry as needed.
3. The registry returns a list of provider addresses to the consumer. When the provider changes, the registry pushes changed data to the consumer.
4. The consumer selects a provider from the list of provider addresses based on the software load balancing algorithm.

#### Meaning of hosting Dubbo applications to EDAS

The core of hosting a Dubbo application to EDAS is to host the registry, configuration center, and metadata center.

- Before:  
Before you host an application to EDAS, you must build and maintain the registry, configuration center, and metadata center. The registry is an open source component such as ZooKeeper or Nacos. The configuration center and metadata center are included in Dubbo Admin.
- After:
  - After you host an application to EDAS, EDAS provides the Dubbo service governance platform and Nacos that contains the registry, configuration center, and metadata center. You do not need to build or maintain these components or monitor their availability. You can use a microservice governance platform that is more powerful than the user-created Dubbo Admin.
  - You can also continue to use the user-created or MSE-hosted ZooKeeper, Nacos, or Eureka registry, and use the microservice governance capabilities provided by EDAS.

Type	Open source component	EDAS component	Hosting instruction
------	-----------------------	----------------	---------------------

Type	Open source component	EDAS component	Hosting instruction
Registry	<ul style="list-style-type: none"><li>• Nacos (recommended)</li><li>• ZooKeeper (recommended)</li><li>• etcd</li><li>• Consul</li><li>• Eureka</li></ul>	<ul style="list-style-type: none"><li>• Nacos (recommended)</li><li>• EDAS registry</li></ul>	After you deploy applications to EDAS, you are connected to the registry by default.
Configuration center	<ul style="list-style-type: none"><li>• Nacos (recommended)</li><li>• ZooKeeper (recommended)</li><li>• Apollo</li></ul>	Nacos (recommended)	After you deploy applications to EDAS, you are connected to the configuration center by default.
Metadata center	<ul style="list-style-type: none"><li>• Nacos (recommended)</li><li>• Redis (recommended)</li><li>• ZooKeeper</li></ul>	Nacos (recommended)	After you deploy applications to EDAS, you are connected to the metadata center by default.

## Benefits of hosting Dubbo applications to EDAS

By hosting Dubbo applications to EDAS, you need only to focus on how to build the logic of the applications. You do not need to pay attention to the creation or maintenance of the registry, configuration center, and metadata center. You can also take advantage of EDAS capabilities such as auto scaling, throttling and degradation, monitoring, and microservice governance for various management purposes. The entire hosting process is completely transparent to you. It does not require you to learn anything, or increase your development costs. The following information specifies the benefits of hosting:

- **Costs:** EDAS provides the service discovery and configuration management features, which save you from maintaining middleware components such as Eureka, ZooKeeper, and Consul.
- **Deployment:** EDAS provides flexible configuration of startup parameters, process visualization, graceful service connection and disconnection, and batch publishing, which allow you to configure, query, and manage your application deployment.
- **Service governance:** EDAS provides service query, conditional routing, blacklist and whitelist, label-based routing, dynamic configuration, load balancing configuration, weight configuration, and centralized configuration management, which allow you to comprehensively govern your services.
- **Auto scaling:** EDAS provides the auto scaling feature, which allows you to dynamically scale your applications in or out based on traffic peaks and valleys.
- **Throttling and degradation:** EDAS provides the throttling and degradation feature to ensure high availability of your applications.
- **Monitoring:** EDAS integrates some monitoring features of Application Real-Time Monitoring Service (ARMS). In addition to instance information query, EDAS also provides advanced monitoring features such as microservice trace query, service call topology query, and slow SQL query.

## 1.3.3.2. Use Spring Boot to develop Dubbo microservice applications

Spring Boot simplifies the configuration and deployment of microservice applications. You can select a registry on your own and manage configurations. This topic describes how to develop a demo Dubbo microservice application based on Nacos by using Spring Boot annotations. If you have a Dubbo application that is developed by using Spring Boot, you can skip this topic and deploy the application to Enterprise Distributed Application Service (EDAS).

### Prerequisites

Before you use Spring Boot to develop Dubbo microservice applications, complete the following tasks:

- Download **Maven** and set the environment variables.
- Download the latest version of **Nacos Server**.
- Start Nacos Server.
  - i. Decompress the downloaded Nacos Server package.
  - ii. Go to the `nacos/bin` directory and start Nacos Server.
    - On Linux, UNIX, or macOS, run the `sh startup.sh -m standalone` command.
    - On Windows, double-click the `startup.cmd` file to run the file.

### Demo project

You can perform the steps described in this topic to build a project. You can also download the **demo project** used in this topic, or clone the project by running the Git command `git clone https://github.com/aliyun/alibabacloud-microservice-demo.git`.

This project contains multiple demo projects. The demo project used in this topic can be found in `alibabacloud-microservice-demo/microservice-doc-demo/dubbo-samples-spring-boot`.

### Create a service provider

1. Create a Maven project named `spring-boot-dubbo-provider`.
2. Add the required dependencies to the `pom.xml` file.  
Spring Boot 2.0.6.RELEASE is used in the example.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>2.0.6.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-actuator</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
    <version>2.7.3</version>
  </dependency>
  <dependency>
    <groupId>com.alibaba.nacos</groupId>
    <artifactId>nacos-client</artifactId>
    <version>1.1.1</version>
  </dependency>
</dependencies>
```


3. Develop a Dubbo service provider. All Dubbo services are provided by using interfaces.

- i. Create a package named `com.alibaba.edas.boot` in `src/main/java`.
- ii. In `com.alibaba.edas.boot`, create an interface named `IHelloService` that contains a `SayHello` method.

```
package com.alibaba.edas.boot;
public interface IHelloService {
    String sayHello(String str);
}
```

- iii. Create a class named `IHelloServiceImpl` in `com.alibaba.edas.boot` to implement this interface.

```
package com.alibaba.edas.boot;
import com.alibaba.dubbo.config.annotation.Service;
@Service
public class IHelloServiceImpl implements IHelloService {
    public String sayHello(String name) {
        return "Hello, " + name + " (from Dubbo with Spring Boot)";
    }
}
```

 **Note** The service annotation is `com.alibaba.dubbo.config.annotation.Service`, which is an annotation class in Dubbo.

4. Configure a Dubbo service.

- i. In `src/main/resources`, create a file named `application.properties` or `application.yml` and open the file.
- ii. Add the following configuration items to `application.properties` or `application.yml`:

```
# Base packages to scan Dubbo Components (e.g @Service , @Reference)
dubbo.scan.basePackages=com.alibaba.edas.boot
dubbo.application.name=dubbo-provider-demo
dubbo.registry.address=nacos://127.0.0.1:8848
```

 **Note**

- You must specify values for the preceding three configuration items because they have no default values.
- The value of `dubbo.scan.basePackages` is the package whose code contains the `com.alibaba.dubbo.config.annotation.Service` and `com.alibaba.dubbo.config.annotation.Reference` annotations. Separate multiple packages with commas (,).
- The value of `dubbo.registry.address` must start with `nacos://`, which is followed by the IP address and port of Nacos Server. The IP address in the sample code is an on-premises address. If your Nacos Server is deployed on another machine, change it to the corresponding IP address.

5. Develop and start the Spring Boot entry class `DubboProvider`.

```
package com.alibaba.edas.boot;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DubboProvider {
    public static void main(String[] args) {
        SpringApplication.run(DubboProvider.class, args);
    }
}
```

6. Log on to the Nacos console at `http://127.0.0.1:8848`. In the left-side navigation pane, click **Services**. On the Services page, view the list of providers. `com.alibaba.edas.boot.IHelloService` is available in the list of service providers. You can also query the service group and provider IP


address of the service.

## Create a service consumer

1. Create a Maven project named `spring-boot-dubbo-consumer`.
2. Add the required dependencies to the `pom.xml` file. Spring Boot 2.0.6.RELEASE is used in the example.

```
<dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-dependencies</artifactId>
    <version>2.0.6.RELEASE</version>
    <type>pom</type>
    <scope>import</scope>
  </dependency>
</dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-actuator</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.dubbo</groupId>
    <artifactId>dubbo-spring-boot-starter</artifactId>
    <version>2.7.3</version>
  </dependency>
  <dependency>
    <groupId>com.alibaba.nacos</groupId>
    <artifactId>nacos-client</artifactId>
    <version>1.1.1</version>
  </dependency>
</dependencies>
```

If you want to use Spring Boot 1.x, use Spring Boot 1.5.x and `com.alibaba.boot:dubbo-spring-boot-starter` 0.1.0.

 **Note** Spring Boot 1.x reached the end of life in August 2019. We recommend that you use a later version to develop applications.

3. Develop a Dubbo consumer.
  - i. Create a package named `com.alibaba.edas.boot` in `src/main/java`.

- ii. In `com.alibaba.edas.boot` , create an interface named `IHelloService` that contains a `SayHello` method.

```
package com.alibaba.edas.boot;
public interface IHelloService {
    String sayHello(String str);
}
```

4. Develop a Dubbo service call.


For example, if you want to call a remote Dubbo service in a controller, use the following sample code:

```
package com.alibaba.edas.boot;
import com.alibaba.dubbo.config.annotation.Reference;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class DemoConsumerController {
    @Reference
    private IHelloService demoService;
    @RequestMapping("/sayHello/{name}")
    public String sayHello(@PathVariable String name) {
        return demoService.sayHello(name);
    }
}
```

 **Note** The `Reference` annotation is `com.alibaba.dubbo.config.annotation.Reference`.

5. Add the following configuration items to the `application.properties` or `application.yaml` file:

```
dubbo.application.name=dubbo-consumer-demo
dubbo.registry.address=nacos://127.0.0.1:8848
```

 **Note**

- You must specify values for the preceding two configuration items because they have no default values.
- The value of `dubbo.registry.address` must start with `nacos://` , which is followed by the IP address and port of Nacos Server. The IP address in the sample code is an on-premises address. If your Nacos Server is deployed on another machine, change it to the corresponding IP address.

6. Develop and start the Spring Boot entry class `DubboConsumer` .

```
package com.alibaba.edas.boot;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DubboConsumer {
    public static void main(String[] args) {
        SpringApplication.run(DubboConsumer.class, args);
    }
}
```

7. Log on to the Nacos console at `http://127.0.0.1:8848`. In the left-side navigation pane, click **Services**. On the Services page, view the list of consumers. `com.alibaba.edas.boot.IHelloService` is available in the list of consumers. You can also view the service group and consumer IP address of the service.

## Verify the result

```
` curl http://localhost:8080/sayHello/EDAS `
` Hello, EDAS (from Dubbo with Spring Boot) `
```

## Deploy an application to EDAS

If you use the console to deploy your application, perform the following steps in your on-premises program before you deploy the application:

1. Add the following configuration of the packaging plug-in to the `pom.xml` file:
  - Provider

```
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<executions>
<execution>
<goals>
<goal>repackage</goal>
</goals>
<configuration>
<classifier>spring-boot</classifier>
<mainClass>com.alibaba.edas.boot.DubboProvider</mainClass>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

- Consumer

```
<build>
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <executions>
      <execution>
        <goals>
          <goal>repackage</goal>
        </goals>
        <configuration>
          <classifier>spring-boot</classifier>
          <mainClass>com.alibaba.edas.boot.DubboConsumer</mainClass>
        </configuration>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

2. Run the `mvn clean package` command to package your on-premises program into a JAR package.

## 1.3.4. Use HSF to develop applications

### 1.3.4.1. HSF overview

High-speed Service Framework (HSF) is a distributed remote procedure call (RPC) service framework that is widely used by Alibaba Group. HSF connects different business systems and decouples the implementation dependencies between systems. HSF unifies service publishing and call methods for distributed applications. This helps you develop distributed applications in an easy manner and provide or use shared feature modules. This also frees you from various types of complex technical details in the distributed field. For example, you do not need to consider remote communication, the implementation of serialization, performance overhead, and the implementation of synchronous or asynchronous calls.

### HSF architecture

HSF is an RPC framework that is provided only for clients. Therefore, it does not have clusters at servers. All HSF services are called between service consumers and service providers in a point-to-point method. However, HSF must depend on the following external systems to implement a complete distributed service system.

- Address registry  
HSF depends on the registry for service discovery. If the registry is not used, HSF can only implement simple point-to-point calls. A service provider cannot publish its service information to others. A service consumer may know which services to call, but cannot obtain the information about the machines that provide these services. In this case, the registry can serve as a medium to discover services. ConfigServer assumes the role of the registry.
- Persistent configuration center

The persistent configuration center is used to store all types of governance rules of HSF services. In the startup process, HSF consumers subscribe to various service governance rules, such as routing rules, grouping rules, and weighting rules, from the persistent configuration center. This way, the HSF consumers can intervene in the address selection logic of the calling procedure based on the rules. Diamond assumes the role of the persistent configuration center.

- **Metadata storage center**

Metadata refers to the information related to HSF services, such as the method list and the parameter structure. Metadata does not affect the calling procedure of HSF. Therefore, the metadata storage center is optional. However, to ensure convenient service maintenance, in the startup process, HSF consumers report the metadata to the metadata storage center for further maintenance. Redis assumes the role of the metadata storage center.

## Features

HSF supports the following service call methods based on its distributed RPC framework:

- **Synchronous calls**

By default, an HSF consumer uses synchronous calls to consume services, and the consumer must wait for the returned results of each call.

- **Asynchronous calls**

For a consumer that calls HSF services, the consumer does not need to synchronously wait for the returned results of all service calls. For such services, HSF provides asynchronous calls so that consumers are not synchronously blocked in HSF call operations. HSF supports the following two types of asynchronous calls:

- **Future calls:** The HSF consumer calls `HSFResponseFuture.getResponse(int timeout)` based on needs to obtain the returned results of calls.
- **Callback calls:** The calls are made by the Callback method that is internally provided by HSF. After the specified HSF service is consumed and the results are returned, the HSF framework calls the `HSFResponseCallback` API operation. The consumer then obtains the call results based on callback notifications.

- **Generic calls**

For a general HSF call, the HSF consumer performs a programmatic call based on API programming in the second-party package of the service to obtain the returned results. However, a generic call initiates an HSF call and obtains returned results by using a method that is independent of the second-party package of the service. For some platform-based products, generic calls can reduce dependencies on second-party packages and realize lightweight system operation.

- **Trace filter extension**

HSF provides a built-in call filter. This can actively find the call filter extension for users and integrate it into HSF call traces. This helps extend HSF requests in a convenient manner.

## Application development methods

HSF allows you to develop applications based on Ali-Tomcat and Pandora Boot.

- **Ali-Tomcat:** This method supports complete HSF features based on Ali-Tomcat and Pandora. The features include service registration and discovery, implicit parameter passing, asynchronous calls, generic calls, and trace filter extension. Applications must be deployed in the format of WAR packages.
- **Pandora Boot:** This method supports comparatively complete HSF features, such as service registration and discovery and asynchronous calls. Applications can be deployed after they are packaged into JAR files that run independently.


## 1.3.4.2. Start the light-weight configuration registry

You can use the light-weight configuration registry in the local environment for application registration, discovery, and configuration management when you develop and test applications. These features are still available after you deploy applications to Enterprise Distributed Application Service (EDAS) Serverless App Engine (SAE). This topic describes how to download, start, and verify the light-weight configuration registry.

### Prerequisites


Before the light-weight configuration registry is used, the following tasks are completed:

- Java Development Kit (JDK) 1.8 or later is downloaded and the environment variable `JAVA_HOME` is specified.
- Ports 8080, 8848, and 9600 are not in use based on the check result.

 **Note** The light-weight configuration registry occupies ports 8080, 8848, and 9600. We recommend that you install and start the light-weight configuration registry on a dedicated machine. If you use the light-weight configuration registry on the local machine, use other ports.

### Step 1: Download the light-weight configuration registry


- Windows:
  - i. Download the [light-weight configuration registry package](#).
  - ii. Decompress the package to a local directory.
- UNIX:
  - i. Run the `wget http://edas.oss-cn-hangzhou.aliyuncs.com/edas-res/edas-lightweight-server-1.0.0.tar.gz` command to download the light-weight configuration registry package.
  - ii. Run the `tar -zxvf edas-lightweight-server-1.0.0.tar.gz` command to decompress the package.

 **Notice** The light-weight configuration registry can be used for only local development and testing. Do not use it in the production environment. If you expose the light-weight configuration registry to the Internet, use an IP access policy for access control.

### Step 2: Start the light-weight configuration registry

1. Go to the `edas-lightweight/bin` directory.
2. Start the light-weight configuration registry and view the start up result.
  - For a Windows system, double-click `startup.bat`.
  - For a UNIX system, run `sh startup.sh`.

```
1. /usr/java/jdk1.8.0/bin/java -server -Xms1g -Xmx1g -Xmn512m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/root/service/edas-lightweight/logs/java_heapdump.hprof -XX:-UseLargePages -Djava.ext.dirs=/usr/java/jdk1.8.0/jre/lib/ext:/usr/java/jdk1.8.0/lib/ext:/root/service/edas-lightweight/plugins/comdb:/root/service/edas-lightweight/plugins/mysql -Xloggc:/root/service/edas-lightweight/logs/nacos_gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=100M -Dnacos.home=/root/service/edas-lightweight -Dnacos.standalone=true -jar /root/service/edas-lightweight/target/edas-lightweight.jar --spring.config.location=classpath:/,classpath:/config/,file:/,file:/config/,file:/root/service/edas-lightweight/conf/ --logging.config=/root/service/edas-lightweight/conf/nacos-logback.xml --server.max-http-header-size=524288
2. edas lightweight is starting with standalone
3. edas lightweight is starting, you can check the /root/edas-lightweight/logs/start.out
```

 **Note** To adjust the values of Java virtual machine (JVM) startup parameters, specify the appropriate JVM parameters in the startup script based on your local environment.

3. (Optional) Check the started listening ports of the light-weight configuration registry. After startup, the local node listens to the following ports:
- Port 8848: used for configuration management and service registration of Nacos applications
  - Port 9600: used for HSF or Dubbo service registration and subscription
  - Port 8080: supports service registration and configuration management.

For Linux and MacOS environments, run `netstat -an | grep -E "8080|8848|9600" | grep -i listen` to check whether the three ports are in the listening state. For Linux, you can also run `netstat -nltp | grep -E "8080|8848|9600"` to check whether the three ports are enabled by the process of the light-weight configuration registry.

## Step 3: Configure hosts in the local development environment

Configure the hosts file of the light-weight configuration registry on the machine where you want to develop and test applications by using the light-weight configuration registry. To be specific, in the Domain Name System (DNS) or the hosts file, configure the `jmenv.tbsite.net` domain name to point to the IP address of the machine where the light-weight configuration registry is enabled.

- Open the hosts file.
  - Windows operating system: `C:\Windows\System32\drivers\etc\hosts`
  - UNIX operating system: `/etc/hosts`
- Add the configuration of the light-weight configuration registry.
  - If you enable the light-weight configuration registry on a machine whose IP address is `192.168.1.100`, add `192.168.1.100 jmenv.tbsite.net` in the hosts file.
  - If you enable the light-weight configuration registry on the local machine, change the preceding IP address to `127.0.0.1 jmenv.tbsite.net` in the hosts file.

## Verify the result

Verify the light-weight configuration registry in the following two aspects:

- Verify the availability of the light-weight configuration registry.

You can start the light-weight configuration registry on a local or independent machine. This results in two access modes.

- Local machine  
Enter the URL `http://127.0.0.1:8080` of the light-weight configuration registry in the web browser and press Enter.
- Independent machine  
Enter the URL `http://machine IP address:8080` of the light-weight configuration registry in the web browser and press Enter.

**Note** After the hosts file is bound, you can access the light-weight configuration registry by using the domain name and the port number `jmenv.tbsite.net:8080`.

The following figure shows the homepage of the light-weight configuration registry.  
If the homepage cannot appear, view the boot log `logs/start.log` in the installation directory to identify the cause of failed startup and fix the bug.

- Verify the feature availability.  
The light-weight configuration registry provides features such as service registration, discovery, configuration management, and namespaces. The configuration management and namespace features are available only if you previously used Nacos.  
Some users previously used the light-weight configuration center or Nacos, and some users use the light-weight configuration registry for the first time. Therefore, the feature availability is verified in two scenarios based on whether you are an original user or a new user.
  - If you previously used the light-weight configuration center or Nacos, you can directly verify the feature availability based on the business logic after you start the light-weight configuration registry.
  - If you did not previously use the light-weight configuration center or Nacos, you need to add and modify configurations in your application after you download and start the light-weight configuration registry. We recommend that you verify the feature availability by referring to the application development documents for specific features.

### 1.3.4.3. Use Ali-Tomcat to develop applications

#### 1.3.4.3.1. Ali-Tomcat overview

Ali-Tomcat is a container on which Enterprise Distributed Application Service (EDAS) depends to run services. It integrates core features, such as service publishing, subscription, and trace query. You can publish applications in Ali-Tomcat in both development and runtime environments.

Pandora is a lightweight isolation container. It is also known as taobao-hsf.sar. This container is used to isolate dependencies between applications and middleware modules and dependencies between middleware modules. EDAS Pandora integrates plug-ins that support the features of middleware products, such as service discovery, configuration push, and trace query. You can use these plug-ins to monitor, govern, track, and analyze EDAS applications to implement comprehensive operations and maintenance (O&M).

**Note** In EDAS, Ali-Tomcat is used only for High-speed Service Framework (HSF) applications that are created as WAR packages.

#### 1.3.4.3.2. Install Ali-Tomcat and Pandora

Ali-Tomcat and Pandora are containers on which Enterprise Distributed Application Service (EDAS) depends to run services. They integrate a series of core features, such as service publishing, subscription, and trace query. Applications must be published in the containers in both development and runtime environments.

## Procedure

1. Download the [Ali-Tomcat-8](#) or [Ali-Tomcat-7](#) package, save it, and then decompress it to a corresponding directory, such as `d:\work\tomcat\`.

 **Note** Use Java Development Kit (JDK) 1.7 or later.

2. Download the [Pandora](#) package, save it, and then decompress it to the deploy directory (`d:\work\tomcat\deploy\`) where Ali-Tomcat is saved.

The following examples illustrate the directory structure:

- In a Linux system, run the `tree -L 2 deploy/` command in the corresponding path to view the directory structure.

```
d:\work\tomcat > tree -L 2 deploy/
deploy/
├── taobao-hsf.sar
│   ├── META-INF
│   ├── lib
│   ├── log.properties
│   ├── plugins
│   ├── sharedlib
│   └── version.properties
```

- In a Windows system, go to the corresponding path to view the directory structure.



### 1.3.4.3.3. Configure startup for an IDE runtime environment

Startup configuration for an IDE runtime environment includes configuration of the Eclipse development environment and IntelliJ IDEA development environment.

#### 1.3.4.3.3.1. Configure an Eclipse development environment

This topic describes how to configure an Eclipse development environment.

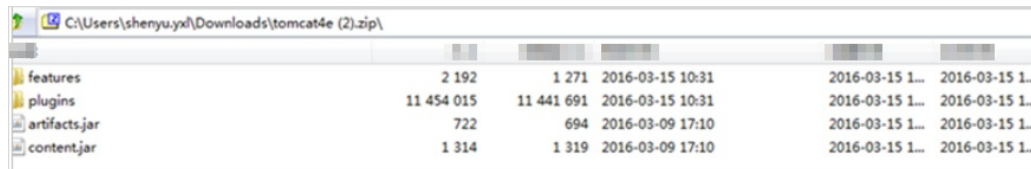
## Context

Before you configure Eclipse, download the Tomcat4E plug-in and select the Pandora directory that you download from [Install Ali-Tomcat and Pandora](#). After Eclipse is configured, you can publish and debug native code in Eclipse.

## Procedure

1. Download the package of the [Tomcat4E plug-in](#) and decompress it to an on-premises location, such as `d:\work\tomcat4e\`. The following figure shows the items that are included in the package.

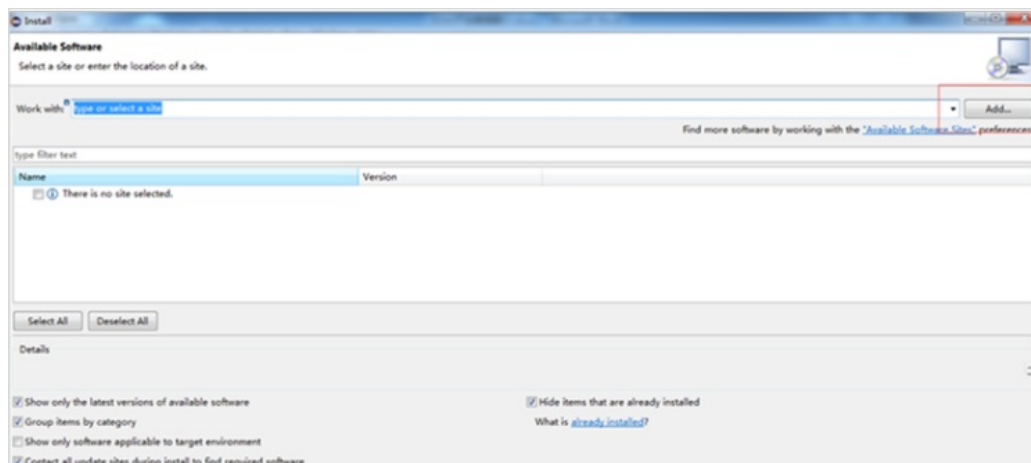
Tomcat4E directory structure



File Name	Size	Size	Modified	Modified	Modified
features	2 192	1 271	2016-03-15 10:31	2016-03-15 10:31	2016-03-15 10:31
plugins	11 454 015	11 441 691	2016-03-15 10:31	2016-03-15 10:31	2016-03-15 10:31
artifacts.jar	722	694	2016-03-09 17:10	2016-03-15 10:31	2016-03-15 10:31
content.jar	1 314	1 319	2016-03-09 17:10	2016-03-15 10:31	2016-03-15 10:31

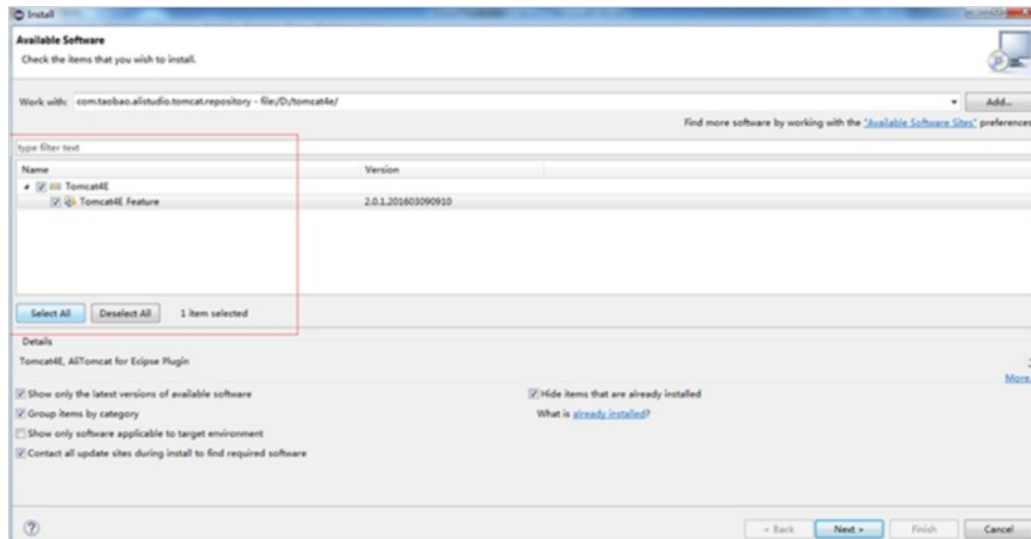
2. Open Eclipse. In the toolbar, choose **Menu > Help**. Then, click **Install New Software**. The following dialog box appears. Choose **Add > Local**, select the `d:\work\tomcat4e\` directory of the decompressed Tomcat4E, and then click **OK**.

Add the Tomcat4E plug-in to Eclipse



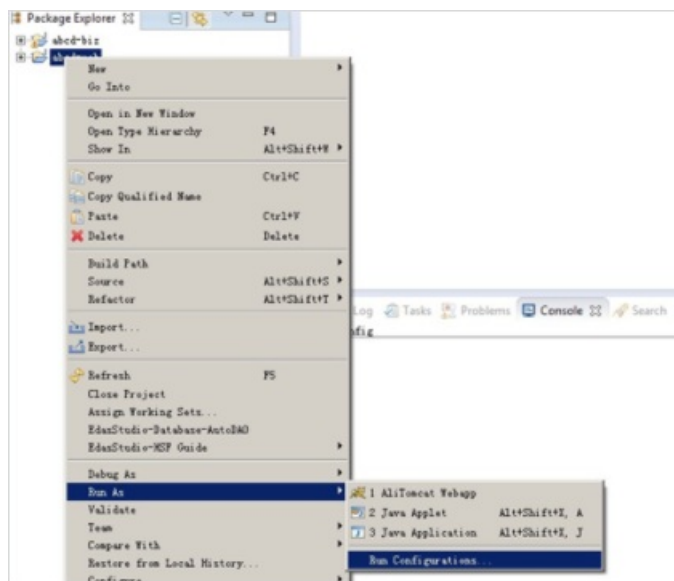
3. Click **Select All**, and then click **Next**. The plug-in is installed.

Install the Tomcat4E plug-in in Eclipse



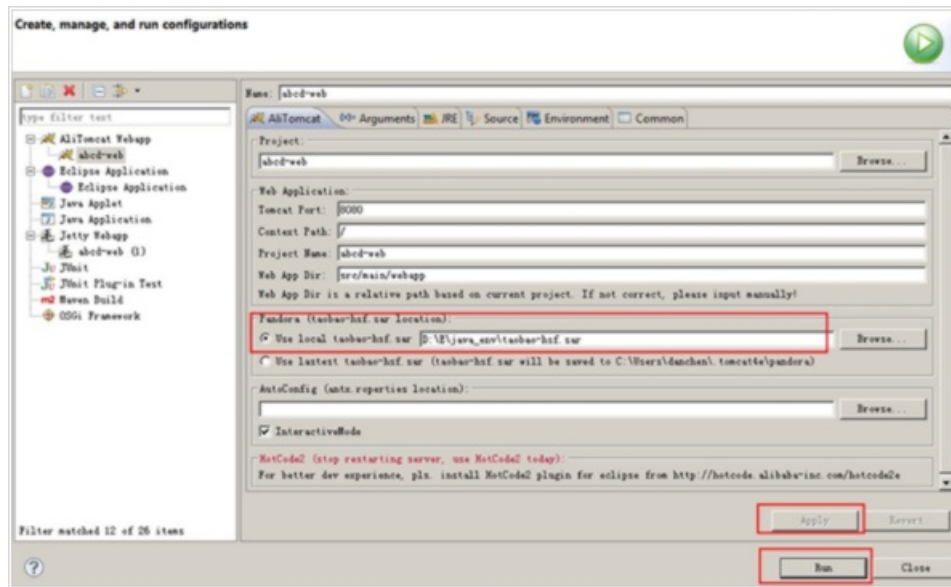
- Restart Eclipse.
- Configure the Eclipse project to activate Tomcat4E. Right-click the corresponding Eclipse project. In the short cut menu that appears, choose **Run As > Run Configurations**.

Eclipse runtime configurations



- In the left-side navigation pane, click **AliTomcat Webapp**, and click the **New Launch Configuration** icon at the top.
- On the page that appears, click the **AliTomcat** tab. In the Pandora (taobao-hsf.sar location) section, select **Use local taobao-hsf.sar** and click **Browse** to select the on-premises path for Pandora, such as `d:\work\tomcat\deploy\taobao-hsf.sar`.

Set the Pandora path



- Click **Apply** or **Run**. You need to configure a project only one time and can directly start it next time.

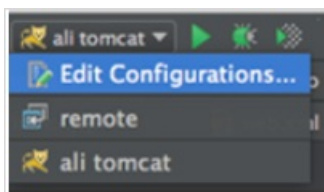
### 1.3.4.3.3.2. Configure the IntelliJ IDEA development environment

The IDEA configuration does not depend on additional plug-ins. You can use the JVM start up parameter -Dpandra.location. However, IntelliJ IDEA supports only the commercial version, and the community version does not support this method.

#### Procedure

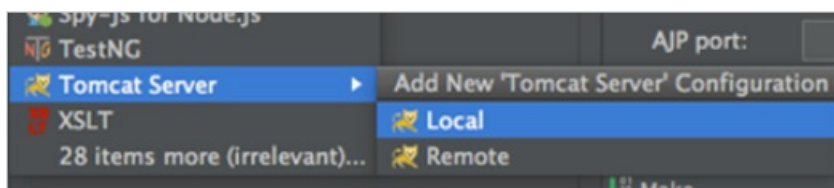
- In the menu or the toolbar, click **Run** and select **Edit Configurations**.

Edit configurations of IntelliJ IDEA



- On the page, click the plus sign (+) and choose **Tomcat Server > Local** to add a local Tomcat start up configuration.

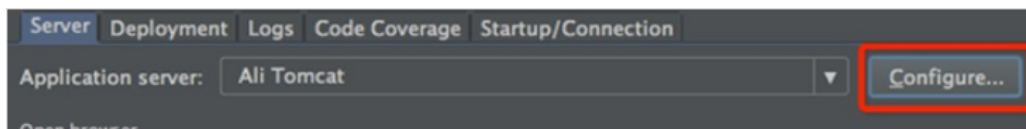
Add local Tomcat configuration



- Configure Ali-Tomcat by performing the following steps: Select the **Server** tab on the right. In the **Application Server** section, click **Configure**. Then, on the page that appears, select the path of downloading and installing Ali-Tomcat in **Install Ali-Tomcat and Pandora**, such as

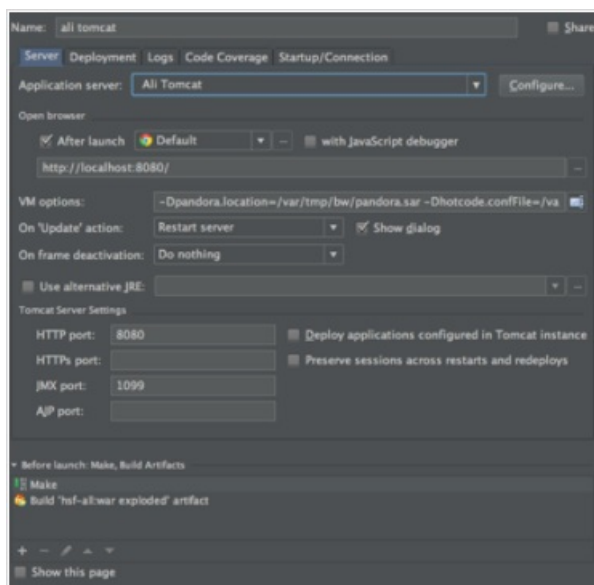
`d:\work\tomcat\.`

Specify the Ali-Tomcat path



4. Select the configured Ali-Tomcat instance from the Application Server drop-down list.

Add the Ali-Tomcat instance



5. In the VM Options field, enter the JVM startup parameter that points to the Pandora path, such as - `Dpandora.location=d:\work\tomcat\deploy\taobao-hsf.sar`.
6. Click **Apply** or **OK** to complete the configuration.

## 1.3.4.3.4. Develop HSF applications (EDAS-SDK)

### 1.3.4.3.4.1. Download a demo project

A demo project is a code sample that is provided for users as a reference. This topic describes how to download a demo project.

#### Context

**Note** Use Java Development Kit (JDK) 1.7 or later.

#### Procedure


1. Download the compressed package of a demo project.  
All the following code described are available in the official demo. You can click [Download demo projects](#) to download official demos.
2. Decompress the downloaded package.  
After you decompress the package, three Maven projects are available: itemcenter-api, itemcenter, and detail. Among the demo projects:

- The itemcenter-api project provides the interface definition.
- The detail project is a service consumer application.
- The itemcenter project is a service provider application.

### 1.3.4.3.4.2. Define service interfaces

HSF services are implemented based on interfaces. After an interface is defined, the provider can implement a specific service through this interface. The consumer also subscribes to the service over this interface.

#### Context

 **Notice** This topic only describes how to define a service interface. However, in actual application scenarios, you need to add routes in an interface and implement them through an instance because definition alone is inadequate.

#### Procedure

1. In the demo project, locate the *itemcenter-api* folder, and locate and open the *ItemService.java* file.  
In the file, the service interface `com.alibaba.edas.carshop.itemcenter.ItemService` is defined and has the following content:

```
public interface ItemService {  
    public Item getItemById( long id );  
    public Item getItemByName( String name );  
}
```

The interface provides two methods: `getItemById` and `getItemByName`, indicating that the `com.alibaba.edas.carshop.itemcenter.ItemService` service provides the `getItemById` and `getItemByName` methods.

2. Define new service interfaces based on planning or actual implementation.

### 1.3.4.3.4.3. Implement services as a provider

The provider implements an interface to provide specific services. Besides code implementation, you must define the XML file used for service publishing because HSF is implemented based on the Spring framework.

Implement service interfaces by code

The sample code of itemcenter in the demo project is as follows:

```
package com.alibaba.edas.carshop.itemcenter;
public class ItemServiceImpl implements ItemService {
    @Override
    public Item getItemById( long id ) {
        Item car = new Item();
        car.setItemId( 11);
        car.setItemName( "Mercedes Benz" );
        return car;
    }
    @Override
    public Item getItemByName( String name ) {
        Item car = new Item();
        car.setItemId( 11);
        car.setItemName( "Mercedes Benz" );
        return car;
    }
}
```

### Configure services

This topic describes how to configure provider services.

## Context

**Implement service interfaces by code** implements the service API `com.alibaba.edas.carshop.itemcenter.ItemService` and returns an `Item` object in two methods. After the code is developed, configure the required general Spring items and add Maven dependencies in the `web.xml` file. Then, use the `<hsf />` tag in the Spring profile to register and publish the service.

## Procedure

1. Add the following Maven dependencies to the *pom.xml* file:

```
<dependencies>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>servlet-api</artifactId>
<version>2.5</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>com.alibaba.edas.carshop</groupId>
<artifactId>itemcenter-api</artifactId>
<version>1.0.0-SNAPSHOT</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-web</artifactId>
<version>2.5.6 (or later)</version>
</dependency>
<dependency>
<groupId>com.alibaba.edas</groupId>
<artifactId>edas-sdk</artifactId>
<version>1.5.0</version>
</dependency>
</dependencies>
```

2. Add the HSF-specific Spring configurations. The following content of the HSF profile (*/resources/hsf-provider-beans.xml*) of the demo project is available:

```
<? xml version="1.0" encoding="UTF-8"? >
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:hsf="http://www.taobao.com/hsf"
xmlns="http://www.springframework.org/schema/beans"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.taobao.com/hsf
http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
<!-- Define the specific implementation of the service. -->
<bean id="itemService" class="com.alibaba.edas.carshop.itemcenter.ItemServiceImpl" />
<!-- Use the hsf:provider tag to define a service provider. -->
<hsf:provider id="itemServiceProvider"
<!-- Use the interface property to indicate that the service is an implementation of the interface class. -->
>
interface="com.alibaba.edas.carshop.itemcenter.ItemService"
<!-- The Spring object that is implemented by the service -->
ref="itemService"
<!-- The version number of the published service. The version number is user-defined and the default value is 1.0.0. -->
version="1.0.0"
</hsf:provider>
</beans>
```

#### Checklist of provider configuration properties

In addition to the properties mentioned in the preceding content, you can use the following properties of the HSF provider configuration:

Property	Description
interface	A required string-type property. It is the interface for providing the service.
version	An optional string-type property. It is the version of the service. The default value is 1.0.0.
clientTimeout	This property applies to all the methods in the interface. However, if the client specifies a time-out period for a method by using the MethodSpecial property, the time-out period configured on the client prevails for the method. Other methods are not affected by this property and still use the time-out period that is configured on the provider.
serializeType	Optional. It indicates the serialization type. Its value is in string format and can be Hessian or Java. Default value: Hessian.
corePoolSize	This property is used to specify a piece of the public thread pool as the core thread pool that is dedicated to this service.
maxPoolSize	This property is used to specify a piece of the public thread pool as the maximum thread pool that is dedicated to this service.
enableTXC	This property enables the distributed transaction middleware GTS.
ref	A required ref-style property. It indicates the ID of the Spring bean that you want to publish as an HSF service.
methodSpecials	Optional. It is used to configure a time-out period (unit: millisecond) for a method. If this property is used, methods in an interface can apply to different time-out periods. This timeout property takes precedence over clientTimeout and has a precedence that is lower than the precedence of methodSpecials on the client.

Tag configuration example:

```
<bean id="impl" class="com.taobao.edas.service.impl.SimpleServiceImpl" />
<hsf:provider id="simpleService" interface="com.taobao.edas.service.SimpleService"
  ref="impl" version="1.0.1" clientTimeout="3000" enableTXC="true"
  serializeType="hessian">
  <hsf:methodSpecials>
    <hsf:methodSpecial name="sum" timeout="2000" />
  </hsf:methodSpecials>
</hsf:provider>
```

Publish services in the development environment

After coding and configuration, you can directly publish the service in Eclipse or IntelliJ IDEA.

## Procedure

1. You can directly run the service by using Ali-Tomcat in Eclipse or IntelliJ IDEA. For more information, see [Startup configurations during IDE operation](#).
2. After the service runs properly, you can query the service you published in the configuration center. For more information, see [Query HSF services in a development environment](#) or [Query HSF services in an online environment](#).

Other JVM start up parameters

This topic describes additional JVM startup parameters.

The following table describes additional startup parameters in the service provider. The parameters are used to change the behavior of HSF.

Property	Description
-Dhsf.server.port	Specifies a port bound to the HSF startup service. Default value: 12200.
-Dhsf.serializer	Specifies the serialization method of HSF. Default value: Hessian.
-Dhsf.server.max.poolsize	Specifies the maximum size of the thread pool of the HSF provider. Default value: 600.
-Dhsf.server.min.poolsize	Specifies the minimum size of the thread pool of the HSF provider. Default value: 50.

### 1.3.4.3.4.4. Subscribe to services as a consumer

Service subscription for consumers is coded in two steps. In the first step, use the `<hsf:consumer/>` tag in the Spring profile to define a bean. In the second step, the bean can be retrieved from the Spring context when you use the bean. In the demo project, detail shows a consumer-specific example.

Configure consumers

This topic describes how to configure a consumer.

The consumer profile consists of the Maven dependency configuration and Spring configuration. This is the same as the provider profile. The Maven dependency configuration for consumers is same as that for providers. For more information, see [Configure services](#).

In addition to the required Spring configuration, you must add the consumer definition to the Spring profile. The HSF subscribes to services from the service center based on the configuration file. The following example shows the configuration content (`/resource/hsf-consumer-beans.xml`) and the meaning:

```
<? xml version="1.0" encoding="UTF-8"? >
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:hsf="http://www.taobao.com/hsf"
  xmlns="http://www.springframework.org/schema/beans"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.taobao.com/hsf
http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
<!-- Example of consuming a service -->
<hsf:consumer
  <!-- The Bean ID that is used to retrieve the consumer object by code injection -->
    id="item"
  <!-- The name of the service. It corresponds to the service name of the service provider. The HSF consumer queries and subscribes to services based on the interface and the version. -->
    interface="com.alibaba.edas.carshop.itemcenter.ItemService"
  <!-- The version number that corresponds to the version number of the service provider. The HSF consumer queries and subscribes to services based on the interface and the version number. -->
    version="1.0.0"
  </hsf:consumer>
</beans>
```

Act as a consumer to use services

This topic provides sample code to describe how to act as a consumer to use services.

The following example shows the sample code in a demo:

```
public class StartListener implements ServletContextListener{
    @Override
    public void contextInitialized( ServletContextEvent sce ) {
        ApplicationContext ctx = WebApplicationContextUtils.getWebApplicationContext( sce.getServletContext() );
        // Retrieve subscribed services based on the bean ID "item" in the Spring configuration: final ItemService itemService = ( ItemService ) ctx.getBean( "item" );
        .....
        // Call the getItemById method of ItemService: System.out.println( itemService.getItemById( 1111 ) );
        // Call the getItemByName method of the ItemService: System.out.println( itemService.getItemByName( "myname is le" ) );
        .....
    }
}
```

Consumer configuration properties

This topic describes consumer configuration properties.

In addition to the properties in the sample code, such as interface and version, you can use other properties listed in the following table.

Property	Description
interface	A required string-type property. It is the interface for calling the target service.
version	An optional string-type property. It is the version of the target service. The default value is 1.0.0.

Property	Description
methodSpecials	Optional. It is used to configure the timeout period (unit: milliseconds) for each method separately. In this way, methods in an interface can apply different timeout periods. The timeout period specified by this property takes precedence over that of the provider.
target	This property is used in the unit testing environment and development environment where hsf.runmode is set to 0. In the runtime environment, this property is invalid, and the target service address pushed by the configuration center is used instead.
connectionNum	Optional. It is the maximum number of connections to the provider. The default value is 1. If you transmit a small amount of data and require a shorter delay, set this property to a larger value to improve TPS.
clientTimeout	It indicates that the consumer sets the same timeout period (unit: milliseconds) for all methods in an interface. Timeout settings are sorted in descending order of priority as follows: consumer MethodSpecial, consumer interface level, provider MethodSpecial, and provider interface level.
asynccallMethods	An optional list-type property. It indicates that the asynchronously called method name list and asynchronous calls are required for calling the service. This property is an empty set by default, which indicates that all methods are called synchronously.
maxWaitTimeForCsAddress	This property indicates the time during which the thread is blocked to wait for address push when a service is subscribed. Otherwise, the address may not be found due to an empty address when the service is called. If the address is not pushed before the blocking time expires, the thread no longer waits and proceeds with initialization.

Tag configuration example:

```
<hsf:consumer id="service" interface="com.taobao.edas.service.SimpleService"
  version="1.1.0" clientTimeout="3000"
  target="10.1.6.57:12200?_TIMEOUT=1000" maxWaitTimeForCsAddress="5000">
  <hsf:methodSpecials>
    <hsf:methodSpecial name="sum" timeout="2000" />
  </hsf:methodSpecials>
</hsf:consumer>
```

Consume a service in a development environment

After you complete coding and configuration tasks, you can directly consume a service in Eclipse or IntelliJ IDEA.

## Procedure

1. You can use Ali-Tomcat to run the service in Eclipse or IntelliJ IDEA. For more information, see [Perform startup configurations for an IDE runtime environment](#).
2. After the service runs, query the service to be consumed in the configuration center. For more information, see the topic about how to build development environments.

## 1.3.4.3.4.5. Use HSF features

This topic describes the procedures and notes for using High-speed Service Framework (HSF) features. You can download [demos](#) to use all features.

Add the edas-sdk dependency to the pom.xml file

To use features related to High-speed Service Framework (HSF), add the following edas-sdk dependency to the pom.xml file:

```
<dependency>
  <groupId>com.alibaba.edas</groupId>
  <artifactId>edas-sdk</artifactId>
  <version>1.5.1</version>
</dependency>
```

Implicit parameter passing

In most cases, implicit parameter passing is used to pass simple key-value (KV) data based on a method that does not call API operations. This method is similar to cookies. Only strings can be passed in parameters.

You can pass a single parameter or multiple parameters in implicit mode.

- **Pass a single parameter**

- Service consumer:

```
RpcContext.getContext().setAttachment("key", "args test");
```

- Service provider:

```
String keyVal=RpcContext.getContext().getAttachment("key");
```

- **Pass multiple parameters**

- Service consumer:

```
Map<String,String> map=new HashMap<String,String>();
map.put("param1", "param1 test");
map.put("param2", "param2 test");
map.put("param3", "param3 test");
map.put("param4", "param4 test");
map.put("param5", "param5 test");
RpcContext rpcContext = RpcContext.getContext();
rpcContext.setAttachments(map);
```

- Service provider:


```
Map<String,String> map=rpcContext.getAttachments();
Set<String> set=map.keySet();
for (String key : set) {
  System.out.println("map value:"+map.get(key));
}
```

Asynchronous calls

You can implement asynchronous calls by using Callback or Future methods.

- **Callback methods**

If a Callback method is used in the consumer, you must configure a listener that has implemented the HSFResponseCallback API operation. After the result is returned, High-speed Service Framework (HSF) calls the method in HSFResponseCallback.

 **Notice** The listener of the HSFResponseCallback API operation cannot be an internal class. Otherwise, the ClassLoader of Pandora reports an error when it is loaded.

The following code provides an example on how to configure a Callback method in XML:

```
<hsf:consumer id="demoApi" interface="com.alibaba.demo.api.DemoApi"
  version="1.1.2">
  <hsf:asyncallMethods>
    <hsf:method name="ayncTest" type="callback"
      listener="com.alibaba.ifree.hsf.consumer.AsyncABTestCallbackHandler" />
    </hsf:asyncallMethods>
  </hsf:consumer>
```

In the preceding configuration, the AsyncABTestCallbackHandler class implements the HSFResponseCallback API operation. In the DemoApi API operation, an ayncTest method exists. The following sample code is used:

```
public void onAppResponse(Object appResponse) {
    // Obtain the value after an asynchronous call is implemented. String msg = (String)appResponse;
    System.out.println("msg:"+msg);
}
```

 **Note**

- The methods are identified only by name. Therefore, repeatedly loaded methods are not differentiated. Methods that have the same name are set to use the same call method.
- HSF calls cannot be repeatedly initiated in a call. If HSF calls are repeatedly initiated in a call, the I/O thread may stop responding and cannot be resumed.

#### • Future methods

If a Future method is used in the consumer, the result is returned based on public static Object getResponse(long timeout) in the HSFResponseFuture API operation after the API operation is called. The following code provides an example on how to configure a Future method in XML:

```
<hsf:consumer id="demoApi" interface="com.alibaba.demo.api.DemoApi" version="1.1.2">
  <hsf:asyncallMethods>
    <hsf:method name="ayncTest" type="future" />
  </hsf:asyncallMethods>
</hsf:consumer>
```

The following sample code is used:

- Asynchronous processing of a single call

```
// Call demoApi.ayncTest();
// Process the business.
...
// Directly obtain the message. If the result is not required, you can skip this step.
String msg=(String) HSFResponseFuture.getResponse(3000);
```

- Concurrent processing of multiple calls  
If multiple calls require concurrent processing, you can first obtain the futures, store them, and then use them after the call is complete.

```
// Define an array of futures. List<HSFFuture> futures = new ArrayList<HSFFuture>();
```

- Concurrent call within a method

```
// Call demoApi.asyncTest();  
// Query a future object first. HSFFuture future=HSFResponseFuture.getFuture();  
futures.add(future);  
// Continue to call other services by using asynchronous calls.  
HSFFuture future=HSFResponseFuture.getFuture();  
futures.add(future);  
// Process the business.  
...  
// Query data and process it. for (HSFFuture hsfFuture : futures) {  
String msg=(String) hsfFuture.getResponse(3000);  
// Process the corresponding data.  
...  
}
```

#### Generic calls

Generic calls do not depend on service APIs and can combine interfaces, methods, and parameters for remote procedure calls (RPCs).

## Procedure

1. Add the generic property to the service consumer configuration.

```
<hsf:consumer id="demoApi" interface="com.alibaba.demo.api.DemoApi" generic="true"/>
```

#### Note

The generic property indicates generic parameters. The value true indicates that generic parameters are supported, and the value false indicates that generic parameters are not supported. The default value is false.

The following code describes the DemoApi method:

```
public String dealMsg(String msg);  
public GenericTestDO dealGenericTestDO(GenericTestDO testDO);
```

2. Retrieve DemoApi to enforce conversion to a generic service.

- i. Import the generic service interface.

```
import com.alibaba.dubbo.rpc.service.GenericService
```

- ii. Retrieve generic objects.

```
// In a web project, you can use Spring beans for injection and then enforce service conversion. In this example, a unit test is performed. Therefore, you must load the configuration file. ClassPathXmlApplicationContext consumerContext = new ClassPathXmlApplicationContext("hsf-generic-consumer-beans.xml");  
GenericService svc = (GenericService) consumerContext.getBean("demoApi");
```

3. Perform a generic operation.

```
Object $invoke(String methodName, String[] parameterTypes, Object[] args) throws GenericException;
```

 Note

- methodName: the name of the method that you want to call
- parameterTypes: the type of the parameters of the method that you want to call
- args: the parameter value that you want to pass.

4. Make a generic call.

- String-type parameters

```
svc.$invoke("dealMsg", new String[] { "java.lang.String" }, new Object[] { "hello" })
```

- Object parameters

```
// First, construct the GenericTestDO entity object. This entity has the ID and name properties. GenericTestDO genericTestDO = new GenericTestDO();
genericTestDO.setId(1980l);
genericTestDO.setName("genericTestDO-tst");
// Use PojoUtils to generate the pojo description of the second-party package. Object comp = PojoUtils.generalize(genericTestDO);
// Call the service in generic mode. svc.$invoke("dealGenericTestDO", new String[] { "com.alibaba.demos.generic.domain.GenericTestDO" }, new Object[] { comp });
```

### 1.3.4.3.4.6. Query services

Enterprise Distributed Application Service (EDAS) allows you to register Dubbo and High-speed Service Framework (HSF) services. This topic only describes how to query HSF services. If your Dubbo services are published to the original registry, such as ZooKeeper, you cannot query the services in the EDAS console.

Query HSF services in a development environment


During development and debugging, you can register and discover services by using the lightweight configuration center. In this case, you can query the services that are provided or called by a specific application in the administration console of the lightweight configuration center.

#### Context

Assume that you start the Enterprise Distributed Application Service (EDAS) configuration center on an Elastic Compute Service (ECS) instance whose IP address is 192.168.1.100. This topic describes how to query High-speed Service Framework (HSF) services in the configuration center.

#### Procedure

1. Open your browser and enter `http://192.168.1.100:8080/` in the address bar to log on to the EDAS configuration center.
2. In the left-side navigation pane, click **Services**. On the page that appears, specify a **service name**, a **service group name**, or an **IP address**, and then click **Search**.

 **Notice** After the configuration center is started, the address of the first network interface card (NIC) is used as the service discovery address by default. If your ECS instance has multiple NICs, configure the `SERVER_IP` variable in the start up script to bind an address.

3. View the corresponding service provider and service consumer.


- Providers tab
  - In the search bar, enter an IP address, and click **Search**. You can then query the services that are provided by the ECS instance at the specified IP address.
  - In the search bar, enter a service name or a service group name. You can then query the IP addresses that provide the specified service.
- Consumers tab
  - In the search bar, enter an IP address, and click **Search**. You can then query the services that are called by the ECS instance at the specified IP address.
  - In the search bar, enter a service name or a service group name. You can then query the IP addresses that have called the specified service.

#### Query HSF services in an online environment

After developed services are packaged and deployed in the Enterprise Distributed Application Service (EDAS) console and the applications are confirmed to start properly, you can query the services in the EDAS console.

1. Log on to the EDAS console. In the left-side navigation pane, select **Application Management > Applications**.
2. On the **Applications** page, click the name of a specific application.
3. In the left-side navigation pane, choose **Services**.

The Services page consists of the **Published services** and **Services consumed** tabs. **Published services** display the service providers that you have configured for the application. **Services consumed** display the service consumers that you have configured for the application.

 **Notice** If you log on to the console by using a Resource Access Management (RAM) user, check whether the RAM user has the permission to view **Services**. In the left-side navigation pane of the console, select **System Management > All Permissions**. On the permission management page, click **Application Management**. On the page that appears, check whether you can view **Services**.

## FAQ

If the service that you want to query does not appear in the corresponding service list, perform the following steps to troubleshoot the issue:

- Check whether the service configurations in the code are correct.
- Check whether the Tomcat process of the service is properly started and whether errors are reported in the logs that are stored in `TOMCAT_HOME/logs/catalina.out` and `$TOMCAT_HOME/logs/localhost.log`.
- Check whether the latest version of the software is used. To view the software version, choose **Software Version** from the left-side navigation pane on the corresponding service information page. If the latest software version is not used, check whether the corresponding High-speed Service Framework (HSF) group is created.

- Check whether the host of the corresponding machine has special network bindings. In normal cases, online machines are not bound to hosts.
- Check whether the network and Elastic Compute Service (ECS) security group configurations of the machine have obvious limits.

## 1.3.4.3.5. Migrate Dubbo applications to HSF (not recommended)

### 1.3.4.3.5.1. Notes for developing Dubbo applications

This topic describes notes for developing Dubbo applications.

1. A single Dubbo configuration file allows you to define multiple groups of consumers. However, Enterprise Distributed Application Service (EDAS) allows you to specify only one group by using the group property.
2. In Dubbo, you must specify the version information when you consume a service. For example, set the version field to 1.0.0. In EDAS, when you consume a service, you can optionally specify this field. The version is 1.0.0 by default.
3. The remote procedure call (RPC) framework of Dubbo supports various protocols, such as remote method invocation (RMI) and Hessian. However, EDAS supports only the Dubbo protocol. For example, you can specify `<dubbo:protocol name="dubbo" port="20880">`. If you specify other protocols, an error message that is similar to the following message may occur:  
"com.alibaba.dubbo.config.ServiceConfig service [xx.xx.xxx] contain xx protocol, HSF not supported".
4. The methods to obtain the RPC context information are different. Dubbo uses the `RpcContext.getContext()` method to obtain the RPC context information. However, High-speed Service Framework (HSF) in EDAS uses the `com.taobao.hsf.util.RequestCtxUtil` method to obtain the RPC context information. After a Dubbo application is migrated to HSF, if HSF still calls `RpcContext.getContext()` to obtain the RPC context information, the "Caused by: java.lang.UnsupportedOperationException: not support getInvocation method in HSF" error is reported.

### 1.3.4.3.5.2. Modify Dubbo application configurations


You can migrate applications developed by using Dubbo to HSF by steps such as modifying the application configuration, configuring multiple registries, and converting JAR to WAR. However, we recommend that new users do not use this method because EDAS supports applications in the native Dubbo framework.

For more information about how to develop applications in the native Dubbo framework, see [Use Spring Boot to develop Dubbo applications](#).

You can configure Dubbo applications that include service providers and consumers in one of the two methods in EDAS: creating XML configuration files and adding annotations. This topic describes the two configuration methods by using examples.

- Configure a service provider in an XML file

```
<? xml version="1.0" encoding="UTF-8"? >
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com
/schema/dubbo"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.o
rg/schema/beans/spring-beans.xsd
  http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
  <dubbo:application name="edas-dubbo-demo-provider" ></dubbo:application>
  <bean id="demoProvider" class="com.alibaba.edas.dubbo.demo.provider.DemoProvider" ></bean>
  <dubbo:registry address="zookeeper://127.0.0.1:2181" ></dubbo:registry>
  <dubbo:protocol name="dubbo" port="20880" threadpool="cached"
    threads="100" ></dubbo:protocol>
  <dubbo:service delay="-1" interface="com.alibaba.edas.dubbo.demo.api.DemoApi"
    ref="demoProvider" version="1.0.0" group="dubbogroup" retries="3" timeout="3000"></dubbo:servi
ce>
</beans>
```

 **Note** The following parameters are optional: threadpool, threads, delay, version, retries, and timeout. Other options are required. You can change the parameter locations in the XML file as needed.

- Configure a service consumer in an XML file

```
<? xml version="1.0" encoding="UTF-8"? >
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com
/schema/dubbo"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.o
rg/schema/beans/spring-beans-2.5.xsd
  http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
  <dubbo:application name="edas-dubbo-consumer" />
  <dubbo:registry address="zookeeper://127.0.0.1:2181" />
  <dubbo:reference id="demoProviderApi"
    interface="com.alibaba.edas.dubbo.demo.api.DemoApi" version="1.0.0" group="dubbogroup" lazy="
true" loadbalance="random">
    <!-- Define a method that does not wait for the return value -->
    <dubbo:method name="sayMsg" async="true" return="false" />
  </dubbo:reference>
  <bean id="demoConsumer" class="com.alibaba.edas.dubbo.demo.consumer.DemoConsumer"
    init-method="reviceMsg">
    <property name="demoApi" ref="demoProviderApi"></property>
  </bean>
</beans>
```

**Note**

- The following parameters are optional: version, group, lazy, loadbalance, async, and return. Other options are required. You can change the parameter locations in the XML file as needed.
- The registry does not take effect in EDAS. All the Dubbo services are automatically registered in the EDAS configuration center. You do not need to concern yourself with this issue.

### 1.3.4.3.5.3. Convert JAR packages to WAR packages

Enterprise Distributed Application Service (EDAS) supports only web projects that are packaged in the WAR format. Therefore, if your project is released in the JAR format, you must convert it to the WAR format first.

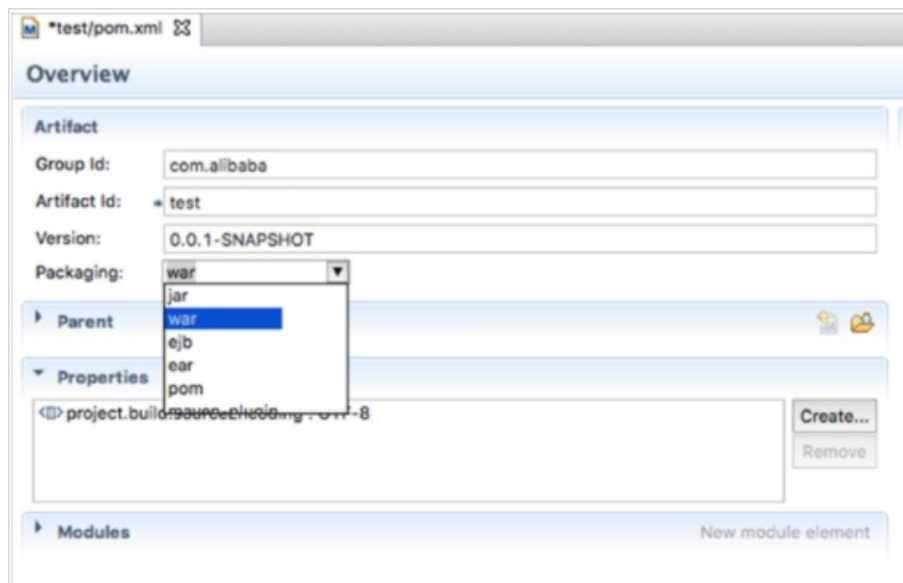
#### Context

**Note** This topic provides an example on how to convert JAR packages to WAR packages based on Maven projects.

#### Procedure

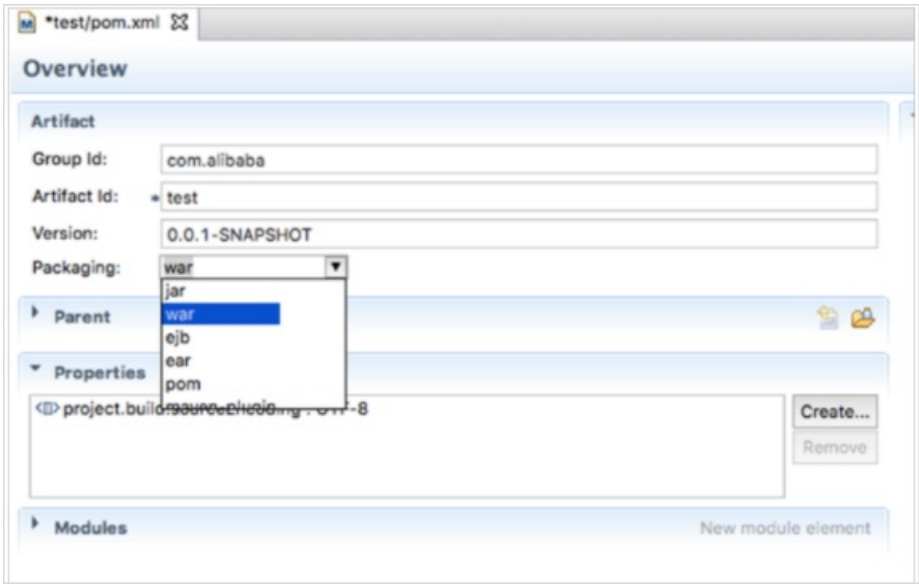
1. Convert the format of a pom.xml file from a JAR package to a WAR package.

Modify the pom.xml file



2. If no web.xml files exist, add a web.xml file configuration.

Add the web.xml file configuration



3. Configure the web.xml file to load the configuration file.  
Configure the web.xml file to load the configuration file

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:hsf-provider-beans.xml</param-value>
</context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener
</listener-class>
</listener>
<listener>
  <listener-class>
    org.springframework.web.context.request.RequestContextListener
  </listener-class>
</listener>
```

1.3.4.3.5.4. Run programs

This topic describes two methods to run programs.

- Right-click to directly start the Tomcat4E plug-in to start a web project. In most cases, this method is used in test environments where the project directly runs in an integrated development environment (IDE). This method does not require excessive configurations. If you have multiple projects, make sure that their Tomcat ports are unique. For more information about how to configure Tomcat4E, see [Install Ali-Tomcat and Pandora](#).
- Access the Enterprise Distributed Application Service (EDAS) console to publish WAR projects.

1.3.4.3.5.5. Compatibility between Dubbo and HSF

To check the compatibility between service properties in the Dubbo and High-speed Service Framework (HSF) configuration files, reference the following table.

Feature	Dubbo parameter	Compatibility description	Error message	Supported by EDAS
---------	-----------------	---------------------------	---------------	-------------------

Feature	Dubbo parameter	Compatibility description	Error message	Supported by EDAS
Timeout	timeout	None	None	Yes
Delayed exposure	delay	None	None	Yes
Thread model	dispatcher="all" threadpool="fixed" threads="100"	None	None	Yes
Echo testing	None	None	None	Yes
Delayed connection	lazy="true"	It is enabled by default.	None	Yes
On-premises calls	protocol="invm"	None	None	Yes
Implicit parameter passing	None	None	None	Yes
Concurrency control	actives="10" /executes="10"	This parameter can be configured in the EDAS console. You do not need to configure it in the Dubbo configuration file.	None	Yes
Connection control	accepts="10" /connections="2"	This parameter can be configured in the EDAS console. You do not need to configure it in the Dubbo configuration file.	None	Yes

Feature	Dubbo parameter	Compatibility description	Error message	Supported by EDAS
Graceful service degradation	None	This parameter can be configured in the EDAS console. You do not need to configure it in the Dubbo configuration file.	None	Yes
Cluster fault tolerance	retries/cluster	Retries are supported.	None	Partially supported
Load balancing	loadbalance	The default value is random.	None	Partially supported
Service grouping	group	Wildcard configuration (*) is not supported.	<code>java.lang.IllegalStateException: HSF2 does not support consuming multiple groups at the same time.</code>	Partially supported
Multi-version	version	Wildcard configuration (*) is not supported.	[HSF-Consumer] Cannot find the address of the service that you want to call.	Partially supported
Asynchronous calls	<code>async="true"</code> <code>return="false"</code>	The return parameter is invalid.	None	Partially supported
Check upon startup	check	Check upon startup is disabled by default in EDAS.	None	Check upon startup is disabled by default.
Multi-protocol	None	Only the Dubbo protocol is supported.	<code>com.alibaba.dubbo.config.ServiceConfig [com.alibaba.demo.api.DemoApi]</code> The remote method invocation (RMI) protocol is configured. This is not supported by HSF2.	Partially supported

Feature	Dubbo parameter	Compatibility description	Error message	Supported by EDAS
Routing rules	None	This parameter can be configured in the EDAS console. You do not need to configure it in the Dubbo configuration file.	None	Yes
Configuration rules	None	This parameter can be configured in the EDAS console. You do not need to configure it in the Dubbo configuration file.	None	Yes
Multi-registry	None	None	None	No
Group aggregation	group="aaa,bbb" merger="true"	Errors are reported.	<a href="#">java.lang.IllegalStateException</a> : HSF2 does not support consuming multiple groups at the same time.	No
Context information	None	Errors are reported.	Caused by: <a href="#">java.lang.UnsupportedOperationException</a> : not support getInvocation method in hsf2	No

After you check the configuration compatibility, you can debug and publish applications based on the preceding chapters.

## 1.3.4.4. Use Pandora Boot to develop applications

Pandora Boot is a more light-weight service that is developed based on Pandora.

- Pandora Boot allows you to directly start a Pandora environment in IDE based on Pandora and FatJar technologies. This greatly improves your development and debugging efficiency.
- Pandora Boot deeply integrates with Spring Boot AutoConfigure. This enables you to enjoy the convenience of the Spring Boot framework.

Spring Boot users who need to use HSF and users who previously used Pandora Boot can use Pandora Boot to develop EDAS applications.

### 1.3.4.4.1. Pandora Boot overview

Pandora Boot is a lightweight service that is developed based on Pandora.

- Pandora Boot allows you to directly start a Pandora environment in an integrated development environment (IDE) based on Pandora and FatJar technologies. This improves your development and debugging efficiency.
- Pandora Boot deeply integrates with Spring Boot AutoConfigure. This allows you to benefit from the Spring Boot framework.


Spring Boot users who need to use High-speed Service Framework (HSF) and users who already use Pandora Boot can use Pandora Boot to develop applications in Enterprise Distributed Application Service (EDAS).

### 1.3.4.4.2. Configure the local repository path and the lightweight configuration center of EDAS

Before you use Pandora Boot to develop High-speed Service Framework (HSF) applications, you must configure the local repository path and the lightweight configuration center of Enterprise Distributed Application Service (EDAS).

- Third-party packages of Spring Cloud for Aliware are released only in the local repository of EDAS. Therefore, you must configure the local repository path of EDAS in Maven.
- The lightweight configuration center must be started for on-premises development and debugging. The lightweight configuration center provides the lightweight version of EDAS service discovery and configuration management.

#### Configure the local repository path of EDAS in Maven


 **Note** Only Maven 3.x and later are supported. Add the local repository path of EDAS in the settings.xml configuration file of Maven.

1. In the Maven configuration file, add the local repository path of EDAS. In most cases, the path of the configuration file is `~/.m2/settings.xml`. The following code provides an example of the configuration:

```
<profiles>
<profile>
  <id>nexus</id>
  <repositories>
    <repository>
      <id>central</id>
      <url>http://repo1.maven.org/maven2</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>central</id>
      <url>http://repo1.maven.org/maven2</url>
      <releases>
```

```
<enabled>true</enabled>
</releases>
<snapshots>
  <enabled>true</enabled>
</snapshots>
</pluginRepository>
</pluginRepositories>
</profile>
<profile>
  <id>edas.oss.repo</id>
  <repositories>
    <repository>
      <id>edas-oss-central</id>
      <name>taobao mirror central</name>
      <url>http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>edas-oss-plugin-central</id>
      <url>http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </pluginRepository>
  </pluginRepositories>
</profile>
</profiles>
<activeProfiles>
  <activeProfile>nexus</activeProfile>
  <activeProfile>edas.oss.repo</activeProfile>
</activeProfiles>
```

2. In the CLI, run the `mvn help:effective-settings` command to check whether the settings are added. Take note of the following issues during verification:
  - No errors are reported. This indicates that the file format of settings.xml is correct.
  - The edas.oss.repo profile is included in profiles. This indicates that the local repository settings have been added to the profiles.
  - The edas.oss.repo profile is included in activeProfiles. This indicates that the edas.oss.repo local repository is activated.

 **Note** If no errors occur when you run the Maven packaging command in the CLI but the integrated development environment (IDE) still cannot download the dependencies, close the IDE and restart it. You can also search for related documentation to obtain information about how to configure Maven in an IDE.

## Configure the lightweight configuration center

For information about how to configure the lightweight configuration center, see [Start the lightweight configuration registry](#).

### 1.3.4.4.3. Use Pandora Boot to develop HSF applications

Enterprise Distributed Application Service (EDAS) fully supports Spring Cloud applications. Therefore, you can directly deploy Spring Cloud applications to EDAS.

- The concept behind Spring Boot is to build anything. It helps resolve complicated XML configuration issues.
- The concept behind Spring Cloud is to coordinate anything. It helps simplify the development of distributed microservices by providing a large number of spring-cloud-starters that feature convenient component access.

EDAS also implements its own Spring Cloud Starter HSF. Therefore, you can use Spring Cloud to develop High-speed Service Framework (HSF) applications.

This topic describes how to use Spring Cloud to develop HSF applications.

#### 1.3.4.4.3.1. Example of HSF application development

This topic provides an example to demonstrate how to develop an HSF application based on Spring Cloud. To develop an HSF application, you must create a service provider and a service consumer.

Create a service provider

This topic describes how to create a service provider and provide services through an interface.

#### Procedure

1. Create a Spring Cloud project named sc-hsf-provider.
2. Add required dependencies to pom.xml.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.8.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hsf</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-pandora</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Dalston.SR4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Although the HSF service framework is independent of the web environment, web-related features are required when EDAS is used to manage the lifecycle of applications. Therefore, you must add the spring-boot-starter-web dependency.

If you do not want to configure the parent of the project as spring-boot-starter-parent, you can add dependencyManagement and set scope=import as follows to manage dependency versions:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>1.5.8.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

3. Define a service interface, and create an interface class `com.aliware.edas.EchoService`.

```
public interface EchoService {  
    String echo(String string);  
}
```

The HSF service framework enables service communication over interfaces. When an interface is defined, providers implement and release specific services by using this interface, and consumers subscribe to and consume services also by using this interface.

The interface `com.aliware.edas.EchoService` provides an `echo` method, which also means that the service `com.aliware.edas.EchoService` provides an `echo` method.

4. Add the implementation class `EchoServiceImpl` of the service provider, and publish the service by using annotations.

```
@HSFProvider(serviceInterface = EchoService.class, serviceVersion = "1.0.0")  
public class EchoServiceImpl implements EchoService {  
    @Override  
    public String echo(String string) {  
        return string;  
    }  
}
```

In addition to the interface specified by `serviceInterface`, HSF also requires the version specified by `serviceVersion` to uniquely identify a service. In this example, the `serviceVersion` property in the `HSFProvider` annotation is set to "1.0.0". Then, the service you want to publish can be identified by the combination of `serviceInterface com.aliware.edas.EchoService` and `serviceVersion 1.0.0`. The configuration in the `HSFProvider` annotation has the highest priority. If it is not configured in the `HSFProvider` annotation, the global configuration of these properties is searched in the `resources/application.properties` file when the service is published. If neither is configured, the default values in the `HSFProvider` annotation are used.

5. Configure the application name and the listener port number in the `application.properties` file in the `resources` directory.

```
spring.application.name=hsf-provider  
server.port=18081  
spring.hsf.version=1.0.0  
spring.hsf.timeout=3000
```

**Best practices:** We recommend that you configure both the **service version** and **service timeout** in the `application.properties` file.

6. Add the main function handler for starting the service.

```
@SpringBootApplication  
public class HSFProviderApplication {  
    public static void main(String[] args) {  
        // Start Pandora Boot for loading the Pandora container.  
        PandoraBootstrap.run(args);  
        SpringApplication.run(ServerApplication.class, args);  
        // Indicate that the service has been started, and a thread waiting time is set. This prevents the container from exiting due to users who exit after running the service code.  
        PandoraBootstrap.markStartupAndWait();  
    }  
}
```

Create a service consumer

In this example, we create a service consumer that calls the service provider by using the interface provided by HSFProvider.

## Procedure

1. Create a Spring Cloud project named sc-hsf-consumer.
2. Add required dependencies to pom.xml.  
The Maven dependencies for HSFConsumer and HSFProvider are exactly the same.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.8.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hsf</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-pandora</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Dalston.SR4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

3. Copy the service interface com.aliware.edas.EchoService, including the package name, published by the service provider to a local instance.

```
public interface EchoService {
    String echo(String string);
}
```

4. Use annotations to inject the **service consumer** instance to the Spring context.

```
@Configuration
public class HsfConfig {
    @HSFConsumer(clientTimeout = 3000, serviceVersion = "1.0.0")
    private EchoService echoService;
}
```

**Best practices:** Configure `@HSFConsumer` once in the Config class, and then inject and use it in multiple steps through `@Autowired`. Usually, `HSFConsumer` is used in multiple places, but you do not have to mark each place where it is used with `@HSFConsumer`. You can write a unified Config class and directly inject it wherever it is needed through `@Autowired`.

5. To facilitate testing, an HTTP method of `/hsf-echo/*` is exposed through SimpleController. Calls to the HSF service provider are internally implemented in the API `/hsf-echo/*`.

```
@RestController
public class SimpleController {
    @Autowired
    private EchoService echoService;
    @RequestMapping(value = "/hsf-echo/{str}", method = RequestMethod.GET)
    public String echo(@PathVariable String str) {
        return echoService.echo(str);
    }
}
```

6. Configure the application name and the listener port number in the application.properties file in the resources directory.

```
spring.application.name=hsf-consumer
server.port=18082
spring.hsf.version=1.0.0
spring.hsf.timeout=1000
```

**Best practices:** We recommend that you configure both the **service version** and **service timeout** in the application.properties file.

7. Add the main function handler for starting the service.

```
@SpringBootApplication
public class HSFConsumerApplication {
    public static void main(String[] args) {
        PandoraBootstrap.run(args);
        SpringApplication.run(HSFConsumerApplication.class, args);
        PandoraBootstrap.markStartupAndWait();
    }
}
```

### 1.3.4.4.3.2. Advanced HSF features

This topic describes how to use Pandora Boot to develop a number of advanced features of High-speed Service Framework (HSF), including unit testing and asynchronous calls.

#### Context

Download the demo source code: [sc-hsf-provider](#) and [sc-hsf-consumer](#).

#### Procedure

## Unit testing

The implementation of spring-cloud-starter-hsf depends on Pandora Boot. Unit testing of Pandora Boot can be enabled through PandoraBootRunner and can be seamlessly integrated with SpringJUnit4ClassRunner.

1. Add a spring-boot-starter-test dependency in Maven.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
</dependency>
```

2. Compile the test classes.

```
@RunWith(PandoraBootRunner.class)
@DelegateTo(SpringJUnit4ClassRunner.class)
// Add the class required for unit testing. It must contain the Spring Boot startup class and can contain
the service class.
@SpringBootTest(classes = {HSFProviderApplication.class, EchoServiceTest.class })
@Component
public class EchoServiceTest {
    /**
     * If you use @HSFConsumer, you must add the service class to the @SpringBootTest annotation and
     use the service class to inject objects. This can prevent abnormal class conversion during generic calls.
     */
    @HSFConsumer(generic = true)
    EchoService echoService;
    // Common calls
    @Test
    public void testInvoke() {
        TestCase.assertEquals("hello world", echoService.echo("hello world"));
    }
    // Generic calls
    @Test
    public void testGenericInvoke() {
        GenericService service = (GenericService) echoService;
        Object result = service.$invoke("echo", new String[] { "java.lang.String" }, new Object[] { "hello world
    " });
        TestCase.assertEquals("hello world", result);
    }
    // Returns the value of Mock.
    @Test
    public void testMock() {
        EchoService mock = Mockito.mock(EchoService.class, AdditionalAnswers.delegatesTo(echoService
    ));
        Mockito.when(mock.echo("")).thenReturn("beta");
        TestCase.assertEquals("beta", mock.echo(""));
    }
}
```

## Asynchronous calls

HSF implements asynchronous calls by using Future or Callback methods.


3. To demonstrate an asynchronous call, you must release the following new service first:  
com.aliware.edas.async.AsyncEchoService.

```
public interface AsyncEchoService {  
    String future(String string);  
    String callback(String string);  
}
```

4. The service provider implements AsyncEchoService and uses annotations to release it.

```
@HSFProvider(serviceInterface = AsyncEchoService.class, serviceVersion = "1.0.0")  
public class AsyncEchoServiceImpl implements AsyncEchoService {  
    @Override  
    public String future(String string) {  
        return string;  
    }  
    @Override  
    public String callback(String string) {  
        return string;  
    }  
}
```

Release from the service provider is consistent with common release, as shown in the preceding two calls. Subsequent configurations are also the same as application startup processes.

 **Note** The logic of asynchronous calls is modified on the service consumer rather than the service provider.

#### Future

5. To enable Future asynchronous calls for the service consumer, use annotations to inject service consumer instances into the Spring context. Configure the method name of asynchronous calls in the futureMethods property of the @HSFConsumer annotation. In this example, the Future method of com.aliware.edas.async.AsyncEchoService is marked as Future asynchronous calls.

```
@Configuration  
public class HsfConfig {  
    @HSFConsumer(serviceVersion = "1.0.0", futureMethods = "future")  
    private AsyncEchoService asyncEchoService;  
}
```

6. After the method is marked as Future asynchronous calls, the actual return value of the method during synchronous execution is null. The call result must be obtained through HSFResponseFuture. In the following sample code, TestAsyncController is used for demonstration:

```
@RestController
public class TestAsyncController {
    @Autowired
    private AsyncEchoService asyncEchoService;
    @RequestMapping(value = "/hsf-future/{str}", method = RequestMethod.GET)
    public String testFuture(@PathVariable String str) {
        String str1 = asyncEchoService.future(str);
        String str2;
        try {
            HSFFuture hsfFuture = HSFFutureResponseFuture.getFuture();
            str2 = (String) hsfFuture.getResponse(3000);
        } catch (Throwable t) {
            t.printStackTrace();
            str2 = "future-exception";
        }
        return str1 + " " + str2;
    }
}
```

In the preceding code, after /hsf-future/123 is called, the str1 value is null and the str2 value is 123. This indicates that the str2 value is the actual return value.

7. If a series of return values of operations are required for service processing, see the following call method:

```
@RequestMapping(value = "/hsf-future-list/{str}", method = RequestMethod.GET)
public String testFutureList(@PathVariable String str) {
    try {
        int num = Integer.parseInt(str);
        List<String> params = new ArrayList<String>();
        for (int i = 1; i <= num; i++) {
            params.add(i + "");
        }
        List<HSFFuture> hsfFutures = new ArrayList<HSFFuture>();
        for (String param : params) {
            asyncEchoService.future(param);
            hsfFutures.add(HSFFutureResponseFuture.getFuture());
        }
        ArrayList<String> results = new ArrayList<String>();
        for (HSFFuture hsfFuture : hsfFutures) {
            results.add((String) hsfFuture.getResponse(3000));
        }
        return Arrays.toString(results.toArray());
    } catch (Throwable t) {
        return "exception";
    }
}
```

Callback

8. To enable Callback asynchronous calls for the service consumer, create a class to implement the HSFFutureCallback API operation and configure it through the @Async annotation.

```
@AsyncOn(interfaceName = AsyncEchoService.class,methodName = "callback")
public class AsyncEchoResponseListener implements HSFResponseCallback{
    @Override
    public void onAppException(Throwable t) {
        t.printStackTrace();
    }
    @Override
    public void onAppResponse(Object appResponse) {
        System.out.println(appResponse);
    }
    @Override
    public void onHSFException(HSFException hsfEx) {
        hsfEx.printStackTrace();
    }
}
```

AsyncEchoResponseListener implements the HSFResponseCallback API operation and sets interfaceName to AsyncEchoService.class and methodName to callback in the @Async annotation. The Callback method of com.aliware.edas.async.AsyncEchoService is marked as Callback asynchronous calls.

9. In this following sample code, TestAsyncController is used for demonstration:

```
@RequestMapping(value = "/hsf-callback/{str}", method = RequestMethod.GET)
public String testCallback(@PathVariable String str) {
    String timestamp = System.currentTimeMillis() + "";
    String str1 = asyncEchoService.callback(str);
    return str1 + " " + timestamp;
}
```

After the service consumer sets the Callback method to Callback asynchronous calls, the return value for synchronous execution is null.

After the result is returned, HSF calls the methods in AsyncEchoResponseListener and obtains the actual return value of the call in the onAppResponse method.

10. Use CallbackInvocationContext to transmit the contextual information of the call to the Callback method. The following code provides an example for the call:

```
CallbackInvocationContext.setContext(timestamp);
String str1 = asyncEchoService.callback(str);
CallbackInvocationContext.setContext(null);
```

The code of AsyncEchoResponseListener is described in the following example:

```
@Override
public void onAppResponse(Object appResponse) {
    Object timestamp = CallbackInvocationContext.getContext();
    System.out.println(timestamp + " " + appResponse);
}
```

In the console, 1513068791916 123 is displayed. This proves that the onAppResponse method in AsyncEchoResponseListener obtains the content of timestamp that is transmitted through CallbackInvocationContext before the call.

### 1.3.4.4.3.3. Local debugging

After a HSF application is developed, you need to locally debug its code before the application is published.

## Procedure


1. Start the light-weight configuration center.

The light-weight configuration center must be used for local code development and debugging. The light-weight configuration center includes a light-weight version of the EDAS service registry for service providers. For more information, see [Start the light-weight configuration registry](#).

2. Start the application.

You can locally start the application by using the following two methods:


- o Start the application in an integrated development environment (IDE)  
Configure the startup parameter `-Djenv.tbsite.net={IP}` in VM options and use the main method to directly start the application. {IP} is the address of the machine where the light-weight configuration center service is started. For example, the value of {IP} is `127.0.0.1` for the light-weight configuration center that is locally started.  
If you do not configure JVM parameters, modify the hosts file to attach `jenv.tbsite.net` to the IP address of the machine where the light-weight configuration center service is started. For more information, see [Start the light-weight configuration registry](#).
- o Start the application by using FatJar  
Add a FatJar packaging plug-in.  
To use Maven to package the `pandora-boot` project into a FatJar package, add the following plug-in to the `pom.xml` file:

 **Note** To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.

```
<build>
  <plugin>
    <groupId>com.taobao.pandora</groupId>
    <artifactId>pandora-boot-maven-plugin</artifactId>
    <version>2.1.7.8</version>
    <executions>
      <execution>
        <phase>package</phase>
        <goals>
          <goal>repackage</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</build>
```

3. After you add the plug-in, in the home directory of the project, run the Maven command `mvn clean package` for packaging. Then, you can view the FatJar package in the target directory.
4. Start the application by running the Java command.

```
java -Djenv.tbsite.net=127.0.0.1 -Dpandora.location=/Users/{username}/.m2/repository/com/taobao/pandora/taobao-hsf.sar/dev-SNAPSHOT/taobao-hsf.sar-dev-SNAPSHOT.jar -jar sc-hsf-provider-0.0.1-SNAPSHOT.jar
```

 **Note** The path specified by `-Dpandora.location` must be a full path and it must be placed before `sc-hsf-provider-0.0.1-SNAPSHOT.jar`.

## Result

Start the service, call the service, and check whether the service can be called.

### 1.3.4.4.3.4. Publish applications to EDAS

After the code for service discovery and calling is debugged in an on-premises environment, it can be published to Enterprise Distributed Application Service (EDAS).

## Procedure

1. When you create an application, choose a container of the latest version. For more information about how to create applications, see the **Application management** section in *User Guide*.
2. In the directory of the project, run the `mvn clean package` command to create the application as a FatJar package. Then, add the FatJar package.
3. To deploy the application to EDAS, upload the FatJar package in the target directory. You do not need to configure the application. For more information about how to deploy applications, see the **Application management** section in *User Guide*.

### 1.3.4.4.4. Develop RESTful applications (not recommended)

This topic describes how to develop RESTful applications based on Spring Cloud in Enterprise Distributed Application Service (EDAS).

#### 1.3.4.4.4.1. Terms

This topic introduces the concepts that you may encounter when you develop applications based on Spring Cloud.

##### ***Pandora***

Pandora is a lightweight container that is used in Alibaba to provide isolation services. Aliware uses Pandora to isolate and load classes. Pandora is a **required** dependency to use Spring Cloud for Aliware.

##### ***VIPServer***

VIPServer is the server that is used for service registry when you develop RESTful applications in Enterprise Distributed Application Service (EDAS). The corresponding client is VIPClient.

##### ***Lightweight configuration center***

The EDAS lightweight configuration center can run in an on-premises environment. It provides service discovery and configuration management features.

##### ***FatJar***

FatJar (also known as the executable IAR) is a JAR file that contains dependencies used by compiled classes and code. You can run the `java -jar` command to use this program.

##### ***EagleEye***

EagleEye is a distributed tracing system that is developed by Alibaba. It traces information based on the distributed tracing system of Google Dapper.

EagleEye processes trillions of distributed trace data each day. It implements automatic tracking of logs by collecting and analyzing the data in various network calls to obtain trace relationships between systems for one request. This helps sort out the application request portals, service call sources, and dependencies. This also helps analyze system call bottlenecks, estimate the link capacities, and locate exceptions.

#### HSF

High-speed Service Framework (HSF) is a distributed service framework that is used in Alibaba. HSF provides support for developers on distributed applications and unified publishing and call methods. Developers can develop distributed applications in an easy method and provide or use public feature modules. This frees the developers from considering various technical details in the distributed field, such as remote communication, performance overhead, call transparency, and the implementation of synchronous and asynchronous calls.

### 1.3.4.4.2. Service registration and discovery

This topic describes how to implement service registration and discovery by using Spring Cloud.


Preparations

Before you use Spring Cloud to implement service registration and discovery, you must make relevant preparations.

#### Configure the EDAS local repository path in Maven

Third-party packages of Spring Cloud for Aliware are released only in the local repository of Enterprise Distributed Application Service (EDAS). Therefore, you must configure the local repository path. For more information about Maven and the Maven local repository path, see [Official documentation](#).

The local repository path of EDAS is <http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository>.

 **Note** You must use Maven 3.x or later, and add the local repository path of EDAS in the settings.xml configuration file of Maven. [Download](#) the sample file.

#### Lightweight configuration center

The lightweight configuration center must be started for on-premises development and debugging. The lightweight configuration center provides a lightweight version of EDAS service discovery and configuration management. For more information about its documentation and download URLs, see [Start the light-weight configuration registry](#).

Implement service registration and discovery

This topic uses a simple example to describe how to discover and call services.

#### Context

The following two roles are involved in this process:

- Service provider: It provides a simple echo service and registers itself to the service discovery center.
- Service consumer: It calls services through the following three clients: RestTemplate, AsyncRestTemplate, and FeignClient.

#### Procedure

Service providers implement services and register them with the service discovery center.

1. Create a Spring Cloud project named sc-vip-client.
2. Add required dependencies to the pom.xml file.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.8.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-vipclient</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-pandora</artifactId>
    <version>1.2</version>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Dalston.SR4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

If you do not need to set the parent of the project to spring-boot-starter-parent, you can use the following method to add dependencyManagement and specify scope=import to manage dependencies.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>1.5.8.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

3. Add the code of the service provider. In the code, the annotation @EnableDiscoveryClient indicates that service registration and discovery feature is required for the application.

```
@SpringBootApplication
@EnableDiscoveryClient
public class ServerApplication {
    public static void main(String[] args) {
        PandoraBootstrap.run(args);
        SpringApplication.run(ServerApplication.class, args);
        PandoraBootstrap.markStartupAndWait();
    }
}
```

4. Create an EchoController to provide simple echo services.

```
@RestController
public class EchoController {
    @RequestMapping(value = "/echo/{string}", method = RequestMethod.GET)
    public String echo(@PathVariable String string) {
        return string;
    }
}
```

5. Configure the application name and the listener port number in the application.properties file in resources.

```
spring.application.name=service-provider
server.port=18081
```

Service consumers call services through the following three clients: RestTemplate, AsyncRestTemplate, and FeignClient.

6. Create a Spring Cloud project named sc-vip-client.
7. Add required dependencies to the pom.xml file.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.8.RELEASE</version>
  <relativePath/>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-vipclient</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-pandora</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-feign</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Dalston.SR4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

spring-cloud-starter-feign is added to the dependency in the pom.xml file compared with that of the service provider. This operation is performed to demonstrate how to use FeignClient.

8. Compared with the service provider, you must add the following two configurations in addition to service enabling and registration to use the following three clients: RestTemplate, AsyncRestTemplate, and FeignClient:
  - i. Add the annotation @LoadBalanced to combine RestTemplate, AsyncRestTemplate, and service discovery.
  - ii. Use the annotation @EnableFeignClients to activate FeignClient.

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class ConsumerApplication {
    @LoadBalanced
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
    @LoadBalanced
    @Bean
    public AsyncRestTemplate asyncRestTemplate(){
        return new AsyncRestTemplate();
    }
    public static void main(String[] args) {
        PandoraBootstrap.run(args);
        SpringApplication.run(ConsumerApplication.class, args);
        PandoraBootstrap.markStartupAndWait();
    }
}
```

9. Before you use FeignClient of EchoService, complete the configuration of this client. Configure the service name and the HTTP request that corresponds to the method. The service name is service-provider that is configured in the sc-vip-server project. The following code is provided:

```
@FeignClient(name = "service-provider")
public interface EchoService {
    @RequestMapping(value = "/echo/{str}", method = RequestMethod.GET)
    String echo(@PathVariable("str") String str);
}
```

10. Create a Controller for call test.
- o /echo-rest/\* calls the service from the service provider through RestTemplate.
  - o /echo-async-rest/\* calls the service from the service provider through AsyncRestTemplate.
  - o /echo-feign/\* calls the service from the service provider through FeignClient.

```
@RestController
public class Controller {
    @Autowired
    private RestTemplate restTemplate;
    @Autowired
    private AsyncRestTemplate asyncRestTemplate;
    @Autowired
    private EchoService echoService;
    @RequestMapping(value = "/echo-rest/{str}", method = RequestMethod.GET)
    public String rest(@PathVariable String str) {
        return restTemplate.getForObject("http://service-provider/echo/" + str, String.class);
    }
    @RequestMapping(value = "/echo-async-rest/{str}", method = RequestMethod.GET)
    public String asyncRest(@PathVariable String str) throws Exception {
        ListenableFuture<ResponseEntity<String>> future = asyncRestTemplate.
            getForEntity("http://service-provider/echo/"+str, String.class);
        return future.get().getBody();
    }
    @RequestMapping(value = "/echo-feign/{str}", method = RequestMethod.GET)
    public String feign(@PathVariable String str) {
        return echoService.echo(str);
    }
}
```

11. Configure the application name and the listener port number.

```
spring.application.name=service-consumer
server.port=18082
```

#### On-premises debugging

After service discovery and calling are enabled, you must debug the code in an on-premises environment before you deploy it to EDAS.

## Procedure

1. Start the lightweight configuration center.  
The lightweight configuration center must be started before on-premises development and debugging. It provides a lightweight version of EDAS service registration and server discovery. For more information, see [Start the light-weight configuration registry](#).
2. Start an application.  
You can start the application in an on-premises environment by using one of the following two methods:
  - Start the application in an integrated development environment (IDE).  
To start the application in an IDE, specify the `-Dvipserver.server.port=8080` startup parameter in VM options, and use the main method to directly start the application.  
If your lightweight configuration center and application are deployed on different machines, perform hosts binding. For more information, see [Start the light-weight configuration registry](#).
  - Start the application by using FatJar.

- a. Add a FatJar packaging plug-in.

To create the pandora-boot project as a FatJar package in Maven, you must add the following plug-in to pom.xml. To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.

```
<build>
  <plugin>
    <groupId>com.taobao.pandora</groupId>
    <artifactId>pandora-boot-maven-plugin</artifactId>
    <version>2.1.7.8</version>
    <executions>
      <execution>
        <phase>package</phase>
        <goals>
          <goal>repackage</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</build>
```

- b. After you add the plug-in, run the `mvn clean package` command in the home directory of the project to create a FatJar package. Then, you can find the FatJar package in the target directory.
- c. Start the application by running the Java command.

```
java -Dvipserver.server.port=8080 -Dpandora.location=/Users/{$username}/.m2/repository/com/taobao/pandora/taobao-hsf.sar/dev-SNAPSHOT/taobao-hsf.sar-dev-SNAPSHOT.jar -jar sc-vip-server-0.0.1-SNAPSHOT.jar
```

 **Note** The path specified by `-Dpandora.location` must be a full path and must be placed before `sc-vip-server-0.0.1-SNAPSHOT.jar`.

3. Start the service and run the `curl` command to call the consumer and the provider. Each call is successful. You can also paste the corresponding URL in the address bar of your browser and observe the response for verification.

#### Publish applications to EDAS

After the code for service discovery and calling is debugged in an on-premises environment, it can be published to Enterprise Distributed Application Service (EDAS).

## Procedure

1. When you create an application, choose a container of the latest version. For more information about how to create applications, see the **Application management** section in *User Guide*.
2. In the directory of the project, run the `mvn clean package` command to create the application as a FatJar package. Then, add the FatJar package.
3. To deploy the application to EDAS, upload the FatJar package in the target directory. You do not need to configure the application. For more information about how to deploy applications, see the **Application management** section in *User Guide*.

#### Migrate from Eureka

For the connected Eureka application for service registration and discovery, you can perform two steps to connect services to the EDAS service registration and discovery center.

## Procedure

1. Modify the source code.

Add two lines in the main function. The original main function has the following content :

```
public static void main(String[] args) {  
    SpringApplication.run(ServerApplication.class, args);  
}
```

The modified main function has the following content :

```
public static void main(String[] args) {  
    PandoraBootstrap.run(args);  
    SpringApplication.run(ServerApplication.class, args);  
    PandoraBootstrap.markStartupAndWait();  
}
```

2. Modify the dependency in the pom.xml file.

Replace `spring-cloud-starter-eureka` with `spring-cloud-starter-vipclient` .  
Before the replacement :

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-starter-eureka</artifactId>  
</dependency>
```

After the replacement :

```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-starter-vipclient</artifactId>  
  <version>1.1</version>  
</dependency>
```

## FAQ

This topic describes possible issues and solutions for scenarios where service registration and discovery are implemented based on Spring Cloud.

1. What can I do if I am unable to enable service discovery for AsyncRestTemplate?  
AsyncRestTemplate provides service discovery only in Spring Cloud Dalston and later versions.
2. What can I do when the FatJar packaging plug-in conflicts with other plug-ins?  
To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.
3. Can taobao-hsf.sar be included during packaging?  
Yes, but this is not recommended.  
You can modify pandora-boot-maven-plugin and set excludeSar to false so that taobao-hsf.sar is automatically added to the package file during packaging.

```
<plugin>
  <groupId>com.taobao.pandora</groupId>
  <artifactId>pandora-boot-maven-plugin</artifactId>
  <version>2.1.7.8</version>
  <configuration>
    <excludeSar>false</excludeSar>
  </configuration>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>repackage</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

This way, the package can be started even if the Pandora address is not configured.

```
java -jar -Dvipserver.server.port=8080 sc-vip-server-0.0.1-SNAPSHOT.jar
```

Before you deploy an application to Enterprise Distributed Application Service (EDAS), restore the default configuration to exclude taobao-hsf.sar.

### 1.3.4.4.3. Distributed tracing

To reduce development costs and improve development efficiency, EDAS provides the EagleEye component for distributed tracing. After the EagleEye tracking point is configured in the code, you can use the distributed tracing function of EDAS without considering log collection, analysis, and storage. This topic describes how to enable the distributed tracing feature of EDAS.

Connect to EagleEye

Before you use distributed tracing, you must connect to EagleEye.

#### Prerequisites

Before you connect to EagleEye, add the local repository path of Enterprise Distributed Application Service (EDAS) in the settings.xml configuration file of Maven.

Third-party packages of Spring Cloud for Aliware are released only in the local repository of EDAS. Therefore, you must configure the local repository path. For more information about Maven and the Maven local repository path, see [Official documentation](#). [Download](#) the sample file.

The local repository path is <http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository>.

 **Note** The Maven version must be 3.x or later.

#### Procedure

1. Add the following public dependencies in the pom.xml file:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-eagleeye</artifactId>
  <version>1.1</version>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-pandora</artifactId>
  <version>1.2</version>
</dependency>
```

2. Add two lines of code to the main function.

Assume that the main function is declared in the following format before modification:

```
public static void main(String[] args) {
    SpringApplication.run(ServerApplication.class, args);
}
```

The main function is declared in the following format after modification:

```
public static void main(String[] args) {
    PandoraBootstrap.run(args);
    SpringApplication.run(ServerApplication.class, args);
    PandoraBootstrap.markStartupAndWait();
}
```

3. Add a FatJar packaging plug-in.

To create the pandora-boot project as a FatJar package in Maven, you must add the following plug-in to pom.xml.

To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.

```
<build>
  <plugins>
    <plugin>
      <groupId>com.taobao.pandora</groupId>
      <artifactId>pandora-boot-maven-plugin</artifactId>
      <version>2.1.7.8</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

## Result

After you complete the preceding three steps, you do not need to set up a collection and analysis system and can directly use the EDAS distributed tracing feature.

## Distributed tracing example

This topic provides an example to describe how to perform distributed tracing for applications.

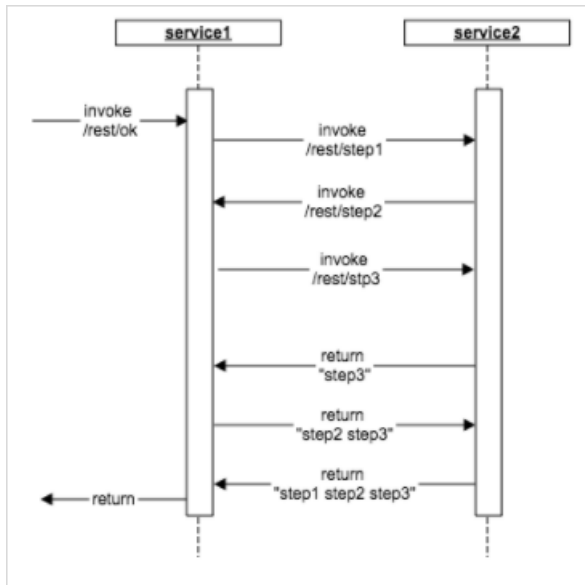
### Context

To demonstrate how to use the distributed tracing feature, two application code demos are used in this example: **service1** and **service2**.

**service1** works as an entrance and provides services for the following three demonstration scenarios:

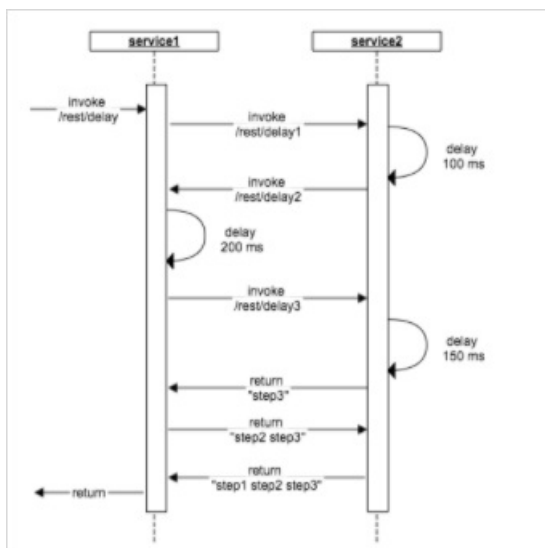
- **/rest/ok** for normal calls

Normal calls



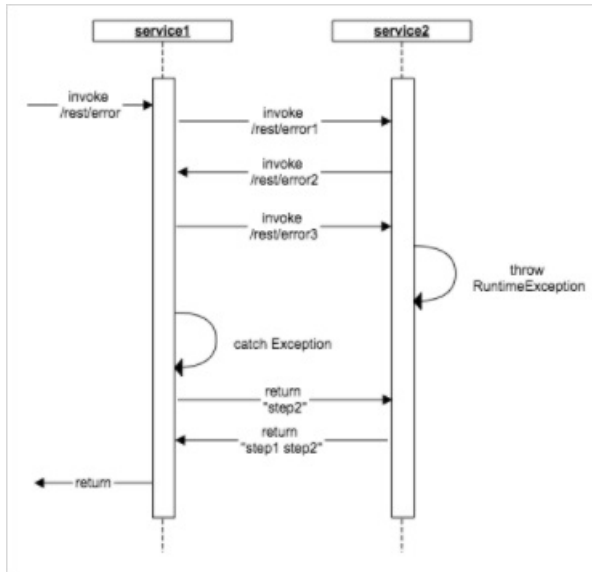
- **/rest/delay** for calls that have a large delay

Calls that have a large delay



- **/rest/error** for calls that have exceptions

Calls that have exceptions



## Procedure

### Deploy applications

The collection and analytics features of EagleEye are built on Enterprise Distributed Application Service (EDAS). To demonstrate how to perform trace queries, deploy service1 and service2 to EDAS first.

1. When you create the applications, choose a container of the latest version. For more information about how to create applications, see the **Application management** section in *User Guide*.
2. Add a FatJar packaging plug-in. Then, in the directory of the project, run the `mvn clean package` command to create applications as FatJar packages.
3. To deploy the applications to EDAS, upload the FatJar packages in the target directory. For more information about how to deploy applications, see the **Application management** section in *User Guide*.

### Call EagleEye

To view the trace information after the applications are deployed, you must familiarize yourself with the corresponding methods for calling service1 in the three demonstration scenarios.

4. You can run the `curl http://{ip:port}/rest/ok` command. You can also use tools, such as Postman, or directly call the methods in your browser.  
To observe the response, we recommend that you call the methods in script mode multiple times.

### View a trace

5. Log on to the EDAS console and find the application that you have deployed.
6. In the left-side navigation pane of the application details page, choose **Application Monitoring > Service Monitoring**.
7. On the Service Monitoring page, click the **RPC Services Provided** tab, and then click **View Trace**.  
For more information about how to use this feature, see *Service monitoring*.  
For more information about service monitoring, see the **Application management** section in *User Guide*.

## FAQ

This topic describes issues that may occur during distributed tracing and their solutions.

## Automatic tracking

EagleEye of Enterprise Distributed Application Service (EDAS) automatically tracks requests of the RestTemplate, AsyncRestTemplate, and FeignClient calls. In the future, more components will support automatic tracking.

## AsyncRestTemplate

AsyncRestTemplate must modify the automatic tracking configuration during the class instantiation phase. Therefore, the eagleEyeAsyncRestTemplate object must be injected to enable the distributed tracing feature. This object supports service discovery by default.

```
@Autowired
private AsyncRestTemplate eagleEyeAsyncRestTemplate;
```


## FatJar packaging plug-ins

To create the pandora-boot project as a FatJar package in Maven, you must add the packaging plug-in of pandora-boot-maven-plugin to pom.xml. To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.

### 1.3.4.4.5. Migrate Dubbo applications to HSF (not recommended)

You can migrate applications developed by using Dubbo to HSF by adding Maven dependencies, adding or modifying the Maven packaging plug-in, and modifying the configurations. However, we recommend that new users do not use this method because EDAS already supports applications in the native Dubbo framework.

For more information about how to develop applications in the native Dubbo framework, see [Use Spring Boot to develop Dubbo applications](#).

 **Note** This topic describes how to modify the configuration. The application development process is not described in detail in this topic. For more information about how to develop applications, download the [Demos for converting Dubbo applications to HSF applications](#).


## Add Maven dependencies

In the `pom.xml` file of the application project, add the `spring-cloud-starter-pandora` dependency.

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-pandora</artifactId>
  <version>1.3</version>
</dependency>
```

## Add or modify the Maven packaging plug-in

In the `pom.xml` file of the application project, add or modify the Maven packaging plug-in.

 **Note** To prevent conflicts with other packaging plug-ins, do not add other FatJar plug-ins to the plugin field in build.

```
<build>
  <plugins>
    <plugin>
      <groupId>com.taobao.pandora</groupId>
      <artifactId>pandora-boot-maven-plugin</artifactId>
      <version>2.1.9.1</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

## Modify the configuration

In the Spring Boot startup class, add the following two lines of code to load Pandora:

```
import com.taobao.pandora.boot.PandoraBootstrap;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class ServerApplication {
    public static void main(String[] args) {
        PandoraBootstrap.run(args);
        SpringApplication.run(ServerApplication.class, args);
        PandoraBootstrap.markStartupAndWait();
    }
}
```


### 1.3.4.5. One call process

This topic describes how to make a HSF call.

In one HSF call, the HSF consumer initiates a request to the HSF provider over the network, and returns the result returned by the HSF provider over the network to the user. This process involves multi-thread interaction, and different domain objects in HSF.

The following figure shows the HSF call process.

Step	Description
------	-------------

Step	Description
1	<p>In the client thread, the user request parameters that are request objects are serialized and then stored in the request communication object.</p> <div>  <b>Note</b> The request communication object uses the HSF protocol and contains multiple items, such as the request ID, that are irrelevant to the request object.         </div>
2	The system submits the request communication object to the I/O thread and encodes the object in the I/O thread.
3	After the encoding is complete, the system delivers the content to the I/O thread of the HSF provider. The client thread waits for the results.
4	The I/O thread of the HSF provider receives the binary content, decodes the binary content into the request communication object, and submits the object to the HSF server thread.
5	The HSF server thread deserializes the request communication object and restores it to the request object.
6	The HSF server thread initiates a reflection call and obtains the result that is the response object.
7	The HSF server thread serializes the response object and stores the result in the communication response object.
8	The HSF server thread submits the response communication object to the I/O thread and encodes the object in the I/O thread.
9	After the HSF provider encodes the object in the I/O thread, the HSF provider returns the object to the HSF consumer.
10	The HSF consumer receives the binary content, decodes the binary content into the response communication object in the I/O thread, and wakes up the client thread.
11	The client thread deserializes the response communication object. When the user receives the response object, a remote call ends.

## 1.3.4.6. Asynchronous calls

This topic describes how to perform asynchronous calls in High-speed Service Framework (HSF).

### Synchronous calls

I/O operations in HSF are performed in an asynchronous manner. In synchronous calls, the consumer executes the `future.get(timeout)` operation and waits for the response from the provider. In this operation, `timeout` indicates the timeout period that is used by the consumer and is 3,000 ms by default. The following figure shows the sequence diagram of a synchronous call.

For the consumer, not all HSF services need to synchronously wait for the response from the provider. For these services, HSF provides an asynchronous method so that the consumer is not blocked due to synchronous calls in HSF. In an asynchronous call, the default value is returned for all calls of HSF services. For example, if the data type of the return value is `int`, 0 is returned, and if the data type of the return value is `object`, null is returned. To obtain the actual result, call the `HSFResponseFuture` or callback function.

### Future asynchronous calls

After you initiate an HSF call, you can obtain the `HSFFuture` object that is associated with the returned result in the context. Then, call `HSFFuture.getResponse(timeout)` to obtain the result that is returned from the provider. The following figure shows the sequence diagram of a Future asynchronous call.

- Configure an HSF service based on API programming.

HSF allows you to call methods to configure asynchronous calls in the format of `name:${methodName};type:future`. The methods are identified only by name. Therefore, repeatedly loaded methods are not differentiated. Methods that have the same name are set to use the same call method.

```
HSFApiConsumerBean hsfApiConsumerBean = new HSFApiConsumerBean();
hsfApiConsumerBean.setInterfaceName("com.alibaba.middleware.hsf.guide.api.service.OrderService");
hsfApiConsumerBean.setVersion("1.0.0");
hsfApiConsumerBean.setGroup("HSF");
// [Set] Future asynchronous calls. List<String> asyncallMethods = new ArrayList<String>();
// Format: name:{methodName};type:future
asyncallMethods.add("name:queryOrder;type:future");
hsfApiConsumerBean.setAsyncallMethods(asyncallMethods);
hsfApiConsumerBean.init(true);
// [Agent] Get the HSF agent. OrderService orderService = (OrderService) hsfApiConsumerBean.getObject();
// ----- Call -----//
// [Call] Initiate an HSF asynchronous call and return null.
OrderModel orderModel = orderService.queryOrder(1L);
// Get the future object in the context of the current call at the earliest opportunity. The future object will
// be overwritten during subsequent calls in the same thread because the object is placed in ThreadLocal. T
// herefore, you must get the object at the earliest opportunity. HSFFuture hsfFuture = HSFResponseFuture
// .getFuture();
// do something else
// The result is obtained in this step. If the call is not complete, the call request is blocked and the consum
// er must wait for the result. The maximum waiting time is 5,000 ms. try {
//     System.out.println(hsfFuture.getResponse(5000));
// } catch (InterruptedException e) {
//     e.printStackTrace();
// }
```

In HSF, the default timeout period is 3,000 ms. If the service object is not returned after the timeout period expires, an error is thrown when you call `HSFFuture.getResponse()`. `HSFFuture.getResponse(timeout)` specifies the timeout period. If the service result is not returned before the specified timeout period expires and the request does not time out, you can call `getResponse` multiple times to obtain the result.

- Configure an HSF service based on Spring.  
Spring is a framework that is widely used in applications. If you do not want to configure HSF services based on API programming, you can configure them based on Spring XML. The following XML configuration has the same effect as the preceding API configuration.

```
<bean id="orderService" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean">
  <property name="interfaceName" value="com.alibaba.middleware.hsf.guide.api.service.OrderService" />
  <property name="version" value="1.0.0"/>
  <property name="group" value="HSF"/>
  <!--Set the interface for service subscription. -->
  <property name="asynccallMethods">
    <list>
      <value>name:queryOrder;type:future</value>
    </list>
  </property>
</bean>
```

- Configure an HSF service based on annotations.  
Spring Boot is widely used. You can use annotations to configure Spring beans and subscribe to HSF services.  
Add the starter dependency to the project.

```
<dependency>
  <groupId>com.alibaba.boot</groupId>
  <artifactId>pandora-hsf-spring-boot-starter</artifactId>
</dependency>
```

In most cases, an HSF consumer is used in multiple places, but you do not need to use `@HSFConsumer` to mark each place where the consumer is used. You need only to write a unified Config class and use `@Autowired` to inject the class when it is used. The following annotation configuration has the same effect as the preceding API configuration:

```
@Configuration
public class HsfConfig {
  @HSFConsumer(serviceVersion = "1.0.0", serviceGroup = "HSF", futureMethods = "sayHelloInFuture")
  OrderService orderService;
}
```

Inject the class at the place where the HSF consumer is used.

```
@Autowired
OrderService orderService;
```

## Callback asynchronous calls

If a Callback method is used in the consumer, you must configure a listener that has implemented the `HSFResponseCallback` API operation. After the result is returned, HSF calls the method in `HSFResponseCallback`. The following figure shows the call sequence diagram.

- Configure an HSF service based on API programming.

#### Call context for callbacks

Before you initiate a call, you can use `CallbackInvocationContext.setContext(Object obj)` to set context information for the current call and save the information to `ThreadLocal`. In the callback function of a listener, you can use `CallbackInvocationContext.getContext()` to obtain the object.

#### Sample callback function

```
public class CallbackHandler implements HSFResponseCallback {
    // It is triggered when a service error occurs.
    @Override
    public void onAppException(Throwable t) {
        t.printStackTrace();
    }
    // The returned result.
    @Override
    public void onAppResponse(Object result) {
        // Get the context that is set for the callback. Object context = CallbackInvocationContext.getContext();
        ;
        System.out.println(result.toString() + context);
    }
    // HSF exceptions.
    @Override
    public void onHSFException(HSFException e) {
        e.printStackTrace();
    }
}
```

#### Configure a callback method based on API programming

```
HSFApiConsumerBean hsfApiConsumerBean = new HSFApiConsumerBean();
hsfApiConsumerBean.setInterfaceName("com.alibaba.middleware.hsf.guide.api.service.OrderService");
hsfApiConsumerBean.setVersion("1.0.0");
hsfApiConsumerBean.setGroup("HSF");
// [Set] Asynchronous callbacks. List<String> asyncallMethods = new ArrayList<String>();
asyncallMethods.add("name:queryOrder;type:callback;listener:com.alibaba.middleware.hsf.CallbackHandler");
hsfApiConsumerBean.setAsyncallMethods(asyncallMethods);
hsfApiConsumerBean.init(true);
// [Agent] Get the HSF agent. OrderService orderService = (OrderService) hsfApiConsumerBean.getObject();
// Optional. Set the context. You can obtain the context from CallbackHandler based on API programming.
CallbackInvocationContext.setContext("in callback");
// Initiate a call. orderService.queryOrder(1L); // The actual return value is null.
// Clear the context. CallbackInvocationContext.setContext(null);
// do something else
```


You can set the context in the calling thread, and then obtain and use it in the listener. Compared with Future asynchronous calls, Callbacks allow you to immediately obtain the returned result.

- Configure an HSF service based on Spring.

```
<bean id="CallHelloWorld" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean">
  <!--Set the interface for service subscription. -->
  <property name="interfaceName" value="com.alibaba.middleware.hsf.guide.api.service.OrderService" />
  <!--Set the version of the service. -->
  <property name="version" value="1.0.0"/>
  <!--Set the group to which the service belongs. -->
  <property name="group" value="HSF"/>
  <property name="asyncallMethods">
    <list>
      <!--future indicates that the Future method is used to obtain the request execution result. For example, a remote API operation is called first, other operations are performed in the same thread, and then Future is called to obtain the result in the same thread. -->
      <!--name:methodName;type:future|callback-->
      <value>name:queryOrder;type:callback;listener:com.alibaba.middleware.hsf.CallbackHandler</value>
    </list>
  </property>
</bean>
```

- Configure a callback asynchronous call for HSF services based on annotations.

```
@AsyncOn(interfaceName = OrderService.class, methodName = "queryOrder")
public class CallbackHandler implements HSFResponseCallback {
  @Override
  public void onAppException(Throwable t) {
    t.printStackTrace();
  }
  @Override
  public void onAppResponse(Object result) {
    // Get the context that is set for the callback. Object context = CallbackInvocationContext.getContext();
    System.out.println(result.toString() + context);
  }
  @Override
  public void onHSFException(HSFException e) {
    e.printStackTrace();
  }
}
```

 **Notice** The callback function is called by a separate thread pool (LinkedBlockingQueue infinite queue). Do not perform time-consuming operations that may affect the onAppResponse callback of other requests. The default corePoolSize and maxPoolSize settings of the callback thread are equal to the number of instance CPUs. You can customize the -D parameters that are described in the following configuration:

- Minimum size of the callback thread pool: -Dhsf.callback.min.poolsize
- Maximum size of the callback thread pool: -Dhsf.callback.max.poolsize

### 1.3.4.7. Generic calls

Normal calls depend on the JAR packages of business clients. Generic calls do not depend on second-party packages. A generic call passes the name of the method to be called, the method signature, and parameter settings to the GenericService interface for service calling. Generic calls are applicable to some gateway applications that cannot depend on all the service-specific second-party packages. hsf-ops service testing also depends on generic calls.

## Configure HSF services by using the API programming method

You can set `generic` to `true` in `HSFConsumerBean` so that the HSF consumer ignores failed interface loading.

```
HSFApiConsumerBean hsfApiConsumerBean = new HSFApiConsumerBean();
hsfApiConsumerBean.setInterfaceName("com.alibaba.middleware.hsf.guide.api.service.OrderService");
hsfApiConsumerBean.setVersion("1.0.0");
hsfApiConsumerBean.setGroup("HSF");
// [Specify] generic settings: hsfApiConsumerBean.setGeneric("true");
hsfApiConsumerBean.init(true);
// Perform generic operations to obtain the proxy: GenericService genericOrderService = (GenericService) hsfApiConsumerBean.getObject();
// ----- Call -----//
// [Call] Initiate an HSF generic call and return Map-type result.
Map orderModelMap = (Map) genericOrderService.$invoke("queryOrder",
    // Array-type input parameters of the method (xxx.getClass().getName())
    new String[] { Long.class.getName() },
    // Parameter. If it is of the plain old Java object (POJO) type, it must be converted to the Map type.
    new Object[] { 1L});
```

The `$invoke` method provided by `GenericService` contains the name of the actually called method, input parameter type, and parameters. The provider can locate the method based on the contained information. Provider-dependent API JAR packages are not used here. An input parameter of the custom data transfer object (DTO) type must be converted to the `Map` type that can be serialized by the consumer.

Call method and parameter description

- Methods do not have input parameters and support only `methodName : service.$invoke("sayHello", null, null)`.
- The generic method type is supported, such as `List<String>`. Pass only `java.util.List` that is the value of `List.class.getName()`. Do not pass `java.util.List<String>`. Otherwise, an error appears to indicate that the method cannot be located.
- If the format of the call method is uncertain, the caller can write a unit testing that depends on the second-party package for generic call. A POJO bean of the `Map` type can be created by using the `generalize()` method of the HSF-provided `com.taobao.hsf.util.PojoUtils` utility class.

```
Map pojoMap = (Map) PojoUtils.generalize(new OrderModel());
```

- Pass the demo that has POJO-type parameters.

```
class User {
    private String name;
    private int age;
    // Use the standard POJO format. The getter and setter methods are omitted here.
}
// Directly use the Map type to construct generic parameters of the POJO type. Map param = new HashMap<String, Object>();
param.put("age", 11);
param.put("name", "Miles");
// If the input parameter is a subclass of the declared parameter type, pass the "class" field to indicate the real type of the POJO parameter. The provider must have this type.
param.put("class", "com.taobao.User");
```

## Configure HSF services by using Spring

The Spring Framework is a component that is widely used in applications. If you do not need to configure HSF services by using the API programming method, you can configure them by using the Spring XML method. The following XML configuration has the same effect as the preceding API programming:

```
<bean id="CallHelloWorld" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean">
    <!--[Specify] the interface for service subscription. -->
    <property name="interfaceName" value="com.alibaba.middleware.hsf.guide.api.service.OrderService"/>
    <!--[Specify] the version of the service. -->
    <property name="version" value="1.0.0"/>
    <!--[Specify] the group to which the service belongs. -->
    <property name="group" value="HSF"/>
    <property name="generic" value="true"/>
</bean>
```

## Notes

- By default, the routing rule does not take effect if the consumer does not include the interface class in a generic call.
- The generic call performs worse than a normal call.
- Configure `-Dhsf.generic.throw.exception=true` (the default value is false and indicates that exceptions are generalized into returned results of the Map type) to throw service exceptions.  
If an exception class locally exists and the exception class is not of the `RuntimeException` type or its subclass, the `UndeclaredThrowableException` exception is thrown. This is because the exception is not declared on the `com.taobao.hsf.remoting.service.GenericService`. The real exception can be retrieved through `getCause`.  
If the exception class does not locally exist, `com.taobao.hsf.util.GenericInvocationException` is thrown.

### 1.3.4.8. Call context

The request context includes call properties and transmitted custom data other than call-defined parameters. The call properties include the called address, name of the application that initiates the call, and time-out period.

## Specify and retrieve the ongoing call context

com.taobao.hsf.util.RequestCtxUtil provides a static method to specify and retrieve the call context. Based on ThreadLocal, getXXX removes the XXX property from the ThreadLocal variable. This method takes effect for only a single call of the current thread. The following table describes the methods that are used to specify and retrieve specific properties.

#### Consumer

Method	Description
setRequestTimeout()	Specifies the time-out period for a single call.
setUserId()	Specifies the userId of the unit service in the current call. This method is required for generic call.
getProviderIp()	Retrieves the IP address of the provider that was latest called.
setTargetServerIp(String ip)	Specifies the IP address of the required server in the next call of the current thread. This IP address must be included in the address list for memory-provided services.
setDirectTargetServerIp(String targetIp)	Specifies the IP address of the server in the next call of the current thread. This method bypasses the registry and ignores the address list in the memory.

#### Provider

Method	Description
getClientIp()	Retrieves the IP address of the caller.
getAppNameOfClient()	Retrieves the name of the application that is used by the caller.
isHttpRequest()	Specifies whether the call is an HTTP call.
getHttpRequest(String key)	Retrieves the header property of the HTTP request.

## Transmit the custom request context

RpcContext provides a method to transmit additional data to the provider without interface modification. The parameter type can be the custom domain object (DO) type or basic types. Ensure that the peer end can serialize parameters of the same type.

Specify the context before the consumer initiates a call.

```
//setup context before rpc call
RPCContext rpcContext = RPCContext.getClientContext();
rpcContext.putAttachment("tenantId", "123");
//rpc call and context is also transmitted to the remote end. orderService.queryOrder(1L);
```

Retrieve the context by using the methods of the provider.

```
//get context data
RPCContext rpcContext = RPCContext.getServerContext();
String myContext = (String)rpcContext.getAttachment("tenantId");
```

### 1.3.4.9. Select a serialization method

The serialization process converts a Java object into a byte array that is to be transmitted over a network. The deserialization process converts a byte array into a Java object. When you select a serialization method, you must consider factors including compatibility and performance. High-Speed Service Framework (HSF) provides a serialization method based on Java and hessian2. By default, HSF uses hessian2 as the method. The following table describes the differences and configurations of the two serialization methods. Configure HSFApiProviderBean only for service providers.

Serialization method	Maven dependency	Configuration	Compatibility	Performance
hessian2	<code>&lt;artifactId&gt;hsf-io-serialize-hessian2&lt;/artifactId&gt;</code>	<code>setPreferSerializeType("hessian2")</code>	Excellent	Excellent
java	<code>&lt;artifactId&gt;hsf-io-serialize-java&lt;/artifactId&gt;</code>	<code>setPreferSerializeType("java")</code>	Optimal	Moderate

### Configure an HSF service based on API programming

```
HSFApiProviderBean hsfApiProviderBean = new HSFApiProviderBean();
hsfApiProviderBean.setPreferSerializeType("hessian2");
```

### Configure an HSF service based on Spring

Spring is a framework that is widely used in applications. If you do not want to configure HSF services based on API programming, you can configure them based on Spring XML. The following XML configuration has the same effect as the preceding API configuration.

```
<bean class="com.taobao.hsf.app.spring.util.HSFSpringProviderBean" init-method="init">
  <!--Set the interface for service publishing. -->
  <property name="serviceInterface" value="com.alibaba.middleware.hsf.guide.api.service.OrderService"/>
</bean>

  <!--Set the target of the object in service implementation to [ref]. This specifies the Spring bean ID of the HSF service to be published.-->
  <property name="target" ref="Referenced bean ID"/>
  <!--Set the version of the service. -->
  <property name="serviceVersion" value="1.0.0"/>
  <!--Set the group to which the service belongs. -->
  <property name="serviceGroup" value="HSF"/>
  <!--Set the service response time. -->
  <property name="clientTimeout" value="3000"/>
  <!--Sets the serialization type for transmitting the object in service implementation. -->
  <property name="preferSerializeType" value="hessian2"/>
</bean>
```

## 1.3.4.10. Configure a timeout period

This topic describes how to configure a timeout period when you develop High-Speed Service Framework (HSF) applications.

### Context

A timeout period must be configured for requests that are initiated for network calls. The default HSF timeout period is 3,000 ms. You can configure a timeout period for the provider and the consumer. By default, the timeout period configured for the consumer has higher priority. If no timeout period is configured for the consumer, the timeout period configured for the provider is used. When you configure a timeout period for the provider, consider the total duration of service execution, serialization, and network communication. We recommend that you configure a default timeout period for each service on the provider. You can configure a timeout period for the consumer based on actual business scenarios. For example, you can set the timeout period to a small value for frontend applications that need to return results to users within a short period of time.

The following table describes the objects and scopes of related API operations. The operations are sorted by priority in descending order.

Priority	API	Object	Scope
0	com.taobao.hsf.util.RequestCtxUtil#setRequestTimeout	Consumer	Single calls
1	HSFApiConsumerBean#setMethodSpecials	Consumer	Methods
2	HSFApiConsumerBean#setClientTimeout	Consumer	API operations
3	-DdefaultHsfClientTimeout	Consumer	All API operations

Priority	API	Object	Scope
4	HSFApiProviderBean#setMethodSpecials	Provider	Methods
5	HSFApiProviderBean#setClientTimeout	Provider	API operations

**Note** Consumer-specific configurations take precedence over provider-specific configurations, and method-specific configurations take precedence over API operation-specific configurations.

## Configure a timeout period for the consumer

- Configure an HSF service based on API programming.  
Set the clientTimeout property of HSFApiConsumerBean. Unit: ms. The following code sets the timeout period to 1,000 ms for API operations and 100 ms for the queryOrder method:

```
HSFApiConsumerBean consumerBean = new HSFApiConsumerBean();
// Configure the timeout period at the API operation level. consumerBean.setClientTimeout(1000);
//xxx
MethodSpecial methodSpecial = new MethodSpecial();
methodSpecial.setMethodName("queryOrder");
// Configure the timeout period at the method level. This setting takes precedence over the timeout setting that is configured at the API operation level. methodSpecial.setClientTimeout(100);
consumerBean.setMethodSpecials(new MethodSpecial[]{methodSpecial});
```

- Configure an HSF service based on Spring.  
Spring is a framework that is widely used in applications. If you do not want to configure HSF services based on API programming, you can configure them based on Spring XML. The following XML configuration has the same effect as the preceding API configuration.

```
<bean id="CallHelloWorld" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean">
    ...
    <property name="clientTimeout" value="1000" />
    <property name="methodSpecials">
        <list>
            <bean class="com.taobao.hsf.model.metadata.MethodSpecial">
                <property name="methodName" value="queryOrder" />
                <property name="clientTimeout" value="100" />
            </bean>
        </list>
    </property>
    ...
</bean>
```

- Configure an HSF service based on annotations.  
Spring Boot is widely used. You can use annotations to configure Spring beans and subscribe to HSF services.
  - Add the starter dependency to the project.

```
<dependency>
  <groupId>com.alibaba.boot</groupId>
  <artifactId>pandora-hsf-spring-boot-starter</artifactId>
</dependency>
```

ii. Add annotations in the code.

In most cases, an HSF consumer is used in multiple places, but you do not need to use `@HSFConsumer` to mark each place where the consumer is used. You need only to write a unified Config class and use `@Autowired` to directly inject the class when the consumer is used. The following annotation configuration has the same effect as the preceding API configuration:

```
@HSFConsumer(clientTimeout = 1000, methodSpecials = @HSFConsumer.ConsumerMethodSpecial(
    methodName = "queryOrder", clientTimeout = "100"))
private OrderService orderService;
```

- Configure a global timeout period for the consumer.
  - Add `-DdefaultHsfClientTimeout=100` to the startup parameters.
  - Add `System.setProperty("defaultHsfClientTimeout", "100")` to the code.

## Configure a timeout period for the provider

- Configure an HSF service based on API programming.

Set the `clientTimeout` property of `HSFApiProviderBean`. Unit: ms. The following code provides an example on how to configure the property:

```
HSFApiProviderBean providerBean = new HSFApiProviderBean();
// Configure the timeout period at the API operation level. providerBean.setClientTimeout(1000);
//xxx
MethodSpecial methodSpecial = new MethodSpecial();
methodSpecial.setMethodName("queryOrder");
// Configure the timeout period at the method level. This setting takes precedence over the timeout setting that is configured at the API operation level. methodSpecial.setClientTimeout(100);
providerBean.setMethodSpecials(new MethodSpecial[]{methodSpecial});
```

- Configure an HSF service based on Spring.

Spring is a framework that is widely used in applications. If you do not want to configure HSF services based on API programming, you can configure them based on Spring XML. The following XML configuration has the same effect as the preceding API configuration.

```
<bean class="com.taobao.hsf.app.spring.util.HSFSpringProviderBean" init-method="init">
  ...
  <property name="clientTimeout" value="1000" />
  <property name="methodSpecials">
    <list>
      <bean class="com.taobao.hsf.model.metadata.MethodSpecial">
        <property name="methodName" value="queryOrder" />
        <property name="clientTimeout" value="2000" />
      </bean>
    </list>
  </property>
  ...
</bean>
```

- Configure an HSF service based on annotations.

Add the annotation. The following annotation configuration has the same effect as the preceding API configuration:

```
@HSFProvider(serviceInterface = OrderService.class, clientTimeout = 3000)
public class OrderServiceImpl implements OrderService {
    @Autowired
    private OrderDAO orderDAO;
    @Override
    public OrderModel queryOrder(Long id) {
        return orderDAO.queryOrder(id);
    }
}
```

### 1.3.4.11. Configure a provider thread pool

This topic describes how to configure a provider thread pool when you develop HSF applications.

#### Schematic diagram of a service thread pool

An HSF provider thread pool includes I/O threads and business threads. The I/O thread model is used in Reactor or Netty. This topic describes how to configure a business thread pool. Business thread pools are classified into default business thread pools and service thread pools. Service thread pools are a subcategory of default business thread pools.

#### Configure a default thread pool

A provider thread pool is used to run business logic. By default, the core size of the thread pool is 50, the maximum size is 720, and keepAliveTime is 500s. SynchronousQueue is used. User requests are not accumulated because queues are not cached. When all the threads (720 in total) of a provider thread pool are processing requests, new requests are immediately denied and the error message "Thread pool is full" is returned. You can configure the pool by specifying the following VM parameters (-D parameters):

- Minimum thread pool size: `-Dhsf.server.min.poolsize`
- Maximum thread pool size: `-Dhsf.server.max.poolsize`
- Keepalive time of thread convergence: `-Dhsf.server.thread.keepalive`

#### Configure a service thread pool

You can configure thread pools that exclusively process slow services and high concurrency. This prevents excessive usage of business threads and protects service calls by applications.

- Configure HSF services by using the API programming method

```
HSFApiProviderBean hsfApiProviderBean = new HSFApiProviderBean();
//...
hsfApiProviderBean.setCorePoolSize("50");
hsfApiProviderBean.setMaxPoolSize("200");
```

- Configure HSF services by using Spring  
The Spring Framework is a component that is widely used in applications. If you do not need to configure HSF services by using the API programming method, you can configure them by using the Spring XML method. The following XML configuration has the same effect as the preceding API programming:

```
<bean class="com.taobao.hsf.app.spring.util.HSFSpringProviderBean" init-method="init">
  <!--[Specify] the interface for service publishing. -->
  <property name="serviceInterface" value="com.alibaba.middleware.hsf.guide.api.service.OrderService"/>
  <property name="corePoolSize" value="50" />
  <property name="maxPoolSize" value="200" />
</bean>
```

- Configure HSF services by using annotations  
Spring Boot is widely used. You can use annotations to configure Spring beans and configure HSF services for publishing.

i. Add the starter dependency to the project.

```
<dependency>
  <groupId>com.alibaba.boot</groupId>
  <artifactId>pandora-hsf-spring-boot-starter</artifactId>
</dependency>
```

- ii. Configure `@HSFProvider` to the implementation type.  
The following annotation method has the same effect as the preceding API programming method:

```
@HSFProvider(serviceInterface = OrderService.class, corePoolSize = 50, maxPoolSize = 200)
public class OrderServiceImpl implements OrderService {
  @Autowired
  private OrderDAO orderDAO;
  @Override
  public OrderModel queryOrder(Long id) {
    return orderDAO.queryOrder(id);
  }
}
```

## 1.3.4.12. API manual

Among the API operations of HSF applications, the key API operations are the operations that are related to creating provider beans and consumer beans.

### Context

Four operation classes are provided based on use cases.

- `com.taobao.hsf.app.api.util.HSFApiProviderBean`: creates provider beans by using the API programming method.
- `com.taobao.hsf.app.api.util.HSFApiConsumerBean`: creates consumer beans by using the API programming method.
- `com.taobao.hsf.app.spring.util.HSFSpringProviderBean`: creates provider beans by using the Spring configuration method.
- `com.taobao.hsf.app.spring.util.HSFSpringConsumerBean`: creates consumer beans by using the Spring configuration method.

The configuration property of `HSFSpringXxxBean` corresponds to the setter method of `HSFApiXxxBean`. The following content describes the four API operation classes from the aspects of provider beans and consumer beans.

## ProviderBean

- API programming method - HSFApiProviderBean

You can publish HSF services by configuring and initializing `com.taobao.hsf.app.api.util.HSFApiProviderBean`.

The `com.taobao.hsf.app.api.util.HSFApiProviderBean` configuration and initialization process is implemented only once for the same service. We recommend that you cache the `HSFApiProviderBean` object because the object is complex.

- Sample code

```
// Instantiate and configure a provider bean.
HSFApiProviderBean hsfApiProviderBean = new HSFApiProviderBean();
hsfApiProviderBean.setServiceInterface("com.taobao.hsf.test.HelloWorldService");
hsfApiProviderBean.setTarget(target); // target indicates the implementation object of the operation that is specified by serviceInterface.
hsfApiProviderBean.setServiceVersion("1.0.0");
hsfApiProviderBean.setServiceGroup("HSF");
// Initialize the provider bean for service publishing. hsfApiProviderBean.init();
```

- Configurable property table

In addition to the properties specified in the preceding sample code, `HSFApiProviderBean` includes many other configurable properties. You can use the corresponding setter methods to configure these properties.

Property name	Type	Required	Default value	Description
serviceInterface	String	Yes	None	Specifies the operation used to provide an HSF service. The consumer subscribes to the service by using this property.
target	Object	Yes	None	Specifies the service implementation object of the operation that is specified by serviceInterface.
serviceVersion	String	No	1.0.0	Specifies the version number of the service. The consumer subscribes to the service by using this property.

Property name	Type	Required	Default value	Description
serviceGroup	String	No	HSF	Specifies the group of the service. The consumer subscribes to the service by using this property.
serviceDesc	String	No	null	Specifies the description of the service for easy management.
clientTimeout	int	No	3000	Specifies the response time-out period. Unit: millisecond. If the provider returns no response in the time-out period, HSFTimeOutException is thrown.
methodSpecials	MethodSpecial[]	No	null	Specifies the response time-out period for some methods in the service. Specify the MethodSpecial.methodName parameter as a method name and specify the MethodSpecial.clientTimeout parameter as a time-out period for the current method. This property takes precedence over clientTimeout for the current provider.

Property name	Type	Required	Default value	Description
preferSerializeType	String	No	hessian2	Specifies the serialization method for the service request parameters and the service response result for HSF2. Valid values: java, hessian, hessian2, json, and kryo.
corePoolSize	String whose value is an integer	No	0	Configures a separate thread pool for the service and specifies the minimum number of active threads. By default, the public thread pool of the HSF provider is used if this property is not specified.
maxPoolSize	String whose value is an integer	No	0	Configures a separate thread pool for the service and specifies the maximum number of active threads. By default, the public thread pool of the HSF provider is used if this property is not specified.

- Spring configuration method - HSFSpringProviderBean  
You can publish HSF services by configuring a bean of the class `com.taobao.hsf.app.spring.util.HSFSpringProviderBean` in the Spring profile.  
Sample code

```
<bean id="helloWorldService" class="com.taobao.hsf.test.HelloWorldService" />
<bean class="com.taobao.hsf.app.spring.util.HSFSpringProviderBean" init-method="init">
  <!-- [Required] Specify the operation used to externally provide an HSF service. -->
  <property name="serviceInterface" value="com.taobao.hsf.test.HelloWorldService" />
  <!-- [Required] Specify the service implementation object of the operation that is specified by serviceInterface. To be specific, the object is the Spring bean ID of the HSF service to be published. -->
  <property name="target" ref="helloWorldService" />
  <!-- [Optional] Specify the version number of the service. The default value is 1.0.0. -->
  <property name="serviceVersion" value="1.0.0" />
  <!-- [Optional] Specify the group to which the service belongs. The default value is HSF. -->
  <property name="serviceGroup" value="HSF" />
  <!-- [Optional] Specify the description of the service for easy management. The default value is null. -->
  <property name="serviceDesc" value="HelloWorldService provided by HSF" />
  <!-- [Optional] Specify the response time-out period in milliseconds. If the provider returns no response in the specified period, HSFTIMEOUTException is thrown. -->
  <!-- The default value is 3000 ms. -->
  <property name="clientTimeout" value="3000" />
  <!-- [Optional] Specify the response time-out period for some methods in the service. This property takes precedence over clientTimeout. -->
  <!-- Specify the MethodSpecial.methodName parameter as a method name and specify the MethodSpecial.clientTimeout parameter as a time-out period for the current method. -->
  <property name="methodSpecials">
    <list>
      <bean class="com.taobao.hsf.model.metadata.MethodSpecial">
        <property name="methodName" value="sum" />
        <property name="clientTimeout" value="2000" />
      </bean>
    </list>
  </property>
  <!-- [Optional] Specify the serialization method for the request parameters and the response result of the service. Valid values: java, hessian, hessian2, json, and kryo. -->
  <!-- The default value is hessian2. -->
  <property name="preferSerializeType" value="hessian2" />
  <!-- [Optional] Configure a separate thread pool for the service and specify the minimum number of active threads. By default, the public thread pool of the HSF provider is used if this property is not specified. -->
  <property name="corePoolSize" value="10" />
  <!-- [Optional] Configure a separate thread pool for the service and specify the maximum number of active threads. By default, the public thread pool of the HSF provider is used if this property is not specified. -->
  <property name="maxPoolSize" value="60" />
</bean>
```

## ConsumerBean

- API programming method - HSFApiConsumerBean  
You can subscribe to HSF services by configuring and initializing com.taobao.hsf.app.api.util.HSFApiConsumerBean. The com.taobao.hsf.app.api.util.HSFApiConsumerBean configuration and initialization process is implemented only once for the same service. We recommend that you cache the HSFApiConsumerBean object and the retrieved HSF agent because the object size is large.

**Note** In the HSF, HSFApiConsumerBean caches service configuration. If a subscribed service is configured for multiple times, only the first configuration takes effect.

- Sample code

```
// Instantiate and configure a consumer bean.
HSFApiConsumerBean hsfApiConsumerBean = new HSFApiConsumerBean();
hsfApiConsumerBean.setInterfaceName("com.taobao.hsf.test.HelloWorldService");
hsfApiConsumerBean.setVersion("1.0.0");
hsfApiConsumerBean.setGroup("HSF");
// Initialize the consumer bean for service subscription.
// true indicates waiting for address push (time-out period: 3000 ms). The default value is false and indicates an asynchronous call.
hsfApiConsumerBean.init(true);
// Retrieve the HSF agent. HelloWorldService helloWorldService = (HelloWorldService) hsfApiConsumerBean.getObject();
// Initiate an HSF call. String helloStr = helloWorldService.sayHello("Li Lei");
```

- Configurable property table

In addition to the properties specified in the preceding sample code, HSFApiConsumerBean includes many other configurable properties. You can use the corresponding setter methods to configure these properties.

Property name	Type	Required	Default value	Description
interfaceName	String	Yes	None	Specifies the name of the interface for service subscription. The consumer subscribes to the service by using this property.
version	String	Yes	None	Specifies the version number of the subscribed service. The consumer subscribes to the service by using this property.
group	String	Yes	None	Specifies the group of the subscribed service. The consumer subscribes to the service by using this property.

Property name	Type	Required	Default value	Description
clientTimeout	int	No	None	Specifies the request time-out period. Unit: millisecond. If the consumer receives no response from the provider, HSFTimeOutException is thrown. clientTimeout that is specified for the consumer takes precedence over clientTimeout that is specified for the provider. If clientTimeout is not specified for the consumer, clientTimeout that is specified for the provider is used during a remote service call.
methodSpecials	MethodSpecial[]	No	null	Specifies the request time-out period for some methods in the service. Specify the MethodSpecial.methodName parameter as a method name and specify the MethodSpecial.clientTimeout parameter as a time-out period for the method. This property takes precedence over clientTimeout for the current consumer.

Property name	Type	Required	Default value	Description
maxWaitTimeForCsAddress	int	No	None	Specifies the time for the synchronous wait for Config Server to push an address. Unit: millisecond. This property prevents HSFAddressNotFoundException from occurring when the service is called before the address is pushed. We recommend that you set this property to 5000 ms to meet the requirement for wait time.

Property name	Type	Required	Default value	Description
asynccallMethods	List	No	null	<p>Specifies a list of methods to be asynchronously called. Each string in the list is in the following format:</p> <ul style="list-style-type: none"> <li>■ name: the name of the method.</li> <li>■ type: the type of the asynchronous call.</li> <li>■ listener: the listener.</li> </ul> <p>In this list, the listener field takes effect for only asynchronous calls of the callback type. Valid values of the type field:</p> <ul style="list-style-type: none"> <li>■ future: The Future method is used to retrieve the request execution result.</li> <li>■ callback: After the remote service call is completed, HSF uses the response to call back the configured listener. This listener must implement HSFResponseCallback.</li> </ul>

Property name	Type	Required	Default value	Description
proxyStyle	String	No	jdk	Specifies the agent mode of the service. Generally, this property does not need to be configured. To block the consumer bean, set this property to javassist.

- Spring configuration method - HSFSpringConsumerBean  
To subscribe to services, you can configure a bean of the class `com.taobao.hsf.app.api.util.HSFSpringConsumerBean` in the Spring profile.  
Sample code

```
<bean id="helloWorldService" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean">
  <!-- [Required] Specify the name of the operation for service subscription. -->
  <property name="interfaceName" value="com.taobao.hsf.test.HelloWorldService" />
  <!-- [Required] Specify the version number of the subscribed service. -->
  <property name="version" value="1.0.0" />
  <!-- [Required] Specify the group of the subscribed service. -->
  <property name="group" value="HSF" />
  <!-- [Optional] Specify the request time-out period in milliseconds. If the consumer receives no response from the provider, HSFTIMEOUTException is thrown. -->
  <!-- clientTimeout that is specified for the consumer takes precedence over clientTimeout that is specified for the provider. If clientTimeout is not specified for the consumer, clientTimeout that is specified for the provider is used during a remote service call. -->
  <property name="clientTimeout" value="3000" />
  <!-- [Optional] Specify the request time-out period for some methods in the service. This property takes precedence over clientTimeout for the consumer. -->
  <!-- Specify the MethodSpecial.methodName parameter as a method name and specify the MethodSpecial.clientTimeout parameter as a time-out period for the current method. -->
  <property name="methodSpecials">
    <list>
      <bean class="com.taobao.hsf.model.metadata.MethodSpecial">
        <property name="methodName" value="sum" />
        <property name="clientTimeout" value="2000" />
      </bean>
    </list>
  </property>
  <!-- [Optional] Specify the time (unit: millisecond) to synchronously wait for Config Server to push an address. -->
  <!-- This prevents HSFAddressNotFoundException from occurring when the service is called before the address is pushed. -->
  <!-- We recommend that you set this property to 5000 ms to meet the requirement for wait time. -->
  <property name="maxWaitTimeForCsAddress" value="5000"/>
  <!-- [Optional] Specify a list of methods to be asynchronously called. Each string in the list is in the following format: -->
  <!-- name: the name of the method. -->
  <!-- type: the type of the asynchronous call. -->
  <!-- listener: the listener. -->
  <!-- The listener field takes effect for only asynchronous calls of the callback type. -->
  <!-- Valid values of the type field: -->
  <!-- future: The Future method is used to retrieve the request execution result. -->
  <!-- callback: HSF uses the response to call back the configured listener. The listener must implement HSFResponseCallback. -->
  <property name="asyncallMethods">
    <list>
      <value>name:sayHello;type:callback;listener:com.taobao.hsf.test.service.HelloWorldServiceCallbackHandler</value>
    </list>
  </property>
  <!-- [Optional] Specify the proxy mode of the service. Generally, this property does not need to be configured. To block the consumer bean, set this property to javassist. -->
  <property name="proxyStyle" value="jdk" />
</bean>
```

### 1.3.4.13. Set JVM -D startup configuration parameters

This topic describes how to set JVM -D start up parameters when developing High-Speed Service Framework (HSF) applications.

**-D hsf.server.port**

Specifies the port for starting HSF services. The default value is 12200. Use another port than the default port if you start multiple HSF providers locally.

**-D hsf.server.max.poolsize**

Specifies the maximum size of the thread pool of the HSF provider. The default value is 720 .

**-D hsf.server.min.poolsize**

Specifies the minimum size of the thread pool of the HSF provider. The default value is 60 .

**-D hsf.client.localcall**

Enables or disables the precedence of calling local HSF clients. The default value is true .

**-D pandora.qos.port**

Specifies the Pandora monitoring port. The default value is 12201 . Use another port than the default port if you start multiple HSF providers locally.

**-D hsf.http.enable**

Specifies whether to enable the HTTP port. The default value is true .

**-D hsf.http.port**

Specifies the HTTP port used by the HSF application to provide services externally. The default value is 12220 . Use another port than the default port if you start multiple HSF providers locally.

**-D hsf.run.mode**

Specifies whether the HSF consumer performs a targeted call, that is, bypassing Config Server. The value 1 indicates that a targeted call is disallowed, and the value 0 indicates a targeted call is allowed. The default value is 1 . Do not set this parameter to 0 unless necessary.

**-D hsf.shuthook.wait**

The wait time for gracefully disconnecting an HSF application, in ms. The default value is 10000 .

**-D hsf.publish.delayed**

Specifies whether to delay publishing all services. The default value is false, indicating not to delay service publishing.

**-D hsf.server.ip**

Specifies the IP address to be bound. By default, the IP address of the first network interface controller (NIC) is bound when multiple NICs exist.

**-D HsfBindHost**

Specifies the host to be bound. By default, the HSF server binds the IP address of the first NIC and reports it to the address registry when multiple NICs exist. If you set this parameter to -DHsfBindHost=0.0.0.0 , the HSF server port is bound to all NICs of the local device.

**-D hsf.publish.interval=400**

Specifies the time interval between the publishing of two services. HSF services are instantly exposed when being published. You can set this parameter to mitigate the burden on starting applications during service exposure. The default value is 400, in ms.

-D `hsf.client.low.water.mark=32` -D `hsf.client.high.water.mark=64` -  
D `hsf.server.low.water.mark=32` -D `hsf.server.high.water.mark=64`

Specifies the write buffer limit for each channel of the consumer or provider.

- The unit is KB. When the consumer exceeds the upper limit, the channel forbids writing new requests and returns an error. Writing is resumed when the write buffer drops below the lower limit.
- When the provider exceeds the upper limit, the channel forbids writing new responses, and the consumer times out because no response is received. Writing is resumed when the write buffer drops below the lower limit.
- The upper and lower limits must be set as a pair, and the upper limit must be greater than the lower limit.

-D `hsf.generic.remove.class=true`

Retrieves the result of a generic call, without output of the `class` field.

-D `defaultHsfClientTimeout`

Specifies the global time-out period of the consumer.

-D `hsf.invocation.timeout.sensitive`

Determines whether the HSF call duration includes the time consumption logic such as connection creation and address selection. The default value of `hsf.invocation.timeout.sensitive` is false.

## 1.4. Resource management

This topic describes Enterprise Distributed Application Service (EDAS) resources and how to use and manage the resources.

In the EDAS console, you can view and use resources, such as Elastic Compute Service (ECS) and Server Load Balancer (SLB) instances. The EDAS resource management feature allows you to use the resources at the application level. EDAS also provides the resource group management feature. When EDAS is used by multiple users or departments, you can configure access control on resources by using Alibaba accounts and Resource Access Management (RAM) users.

### 1.4.1. VPC

After synchronizing your VPCs to the Enterprise Distributed Application Service (EDAS) console, you can view VPC details in the EDAS console.

#### Background

Alibaba Cloud provides two network types:

- Classic network  
The classic network is suitable for customers who require high network availability.
- Virtual Private Cloud (VPC)  
A VPC network supports custom isolation settings. You can define a custom VPC topology and IP address. The VPC network is suitable for customers with high cybersecurity requirements and network management capability.

#### View VPCs

After synchronizing your VPCs to the Enterprise Distributed Application Service (EDAS) console, you can view VPC details in the EDAS console.

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Resources > VPC**.
3. On the **VPC** page, select a **region** to view the VPC instances in the region.
  - **VPC ID**: the ID that is automatically generated by the system when the VPC is created. Click a VPC ID to go to the VPC console.
  - **Name**: the custom name that is defined for the created VPC.
  - **CIDR**: the CIDR block that is defined for the created VPC.
  - **Status**: the VPC status, which can be Running or Stopped. Expired VPCs do not appear.
  - **ECS Instances**: the number of Elastic Compute Service (ECS) instances created in this VPC. Click a number to go to the ECS page, where you can view all the ECS instances in this VPC.

## 1.4.2. Manage a namespace

Namespaces provide applications with isolated runtime environments, including the development, testing, and production environments. You can use namespaces to isolate resources and services. You are not allowed to create two namespaces that have the same name in the same region.

### Scenario

Assume that you have three sets of environments in a network and use them for application development, testing, and production. You can separately create the Dev, Test, and Prod namespaces for the three environments. If you create clusters and deploy applications in the three namespaces, the resources, applications, and services in a namespace are isolated from the resources, applications, and services in the other namespaces. Therefore, you cannot call services or push configurations across namespaces.

### Default namespace

When you create an application in Enterprise Distributed Application Service (EDAS), you can select the **Default** namespace. This indicates that the application does not use a namespace to isolate resources and services.

If you select the default namespace of a region in the application list, the selected applications do not belong to namespaces.

### Create a namespace

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Namespaces**.
3. On the **Namespaces** page, click **Create Namespace** in the upper-right corner.
4. In the **Create Namespace** dialog box, configure the namespace parameters and click **Create**.

### Create Namespace

\* Namespace Name

Enter the namespace of a specific environment, such as the namespace of the development environment

\* Namespace ID

cn-wulan-env116-d01

It can only contain letters or digits.

Region

cn-wulan-env116-d01

Description

Enter description

0/64

Create

Cancel

Parameter	Description
Namespace Name	Enter a name for the namespace that you create.
Namespace ID	Enter a custom string to specify the namespace ID. The namespace ID can contain only letters and digits.
Region	The region to which the current namespace belongs. This parameter cannot be modified.
Description	Enter a string of text to describe the namespace.

## Modify a namespace

1. Find a namespace in the namespace list, and click **Edit** in the **Actions** column.
2. In the **Edit Namespace** dialog box, change **Namespace Name** and **Description**. You can also enable or disable the remote debugging feature. After the namespace is modified, click **OK**.

## Delete a namespace

You can delete a namespace if this namespace meets the following conditions:

- The namespace does not contain a cluster.
  - The namespace does not contain an Elastic Compute Service (ECS) instance.
1. Find a namespace in the namespace list, and click **Delete** in the **Actions** column.
  2. In the message that appears, click **Delete**.

## 1.4.3. Manage ECS resources

### 1.4.3.1. Import ECS instances

Before you deploy applications by using Enterprise Distributed Application Service (EDAS), you need to import Elastic Compute Service (ECS) instances to a specified cluster and install EDAS Agent.

### Prerequisites

EDAS Agent must be installed on each target ECS instance. Before you install EDAS Agent, ensure that

the RAM user is authorized. For information about how to perform authorization, see the "RAM" topic in *ASCM Console User Guide*.

## Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Resources > ECS**.
3. On the **ECS** page, click **Import ECS** in the upper-right corner.
4. In the **Select Cluster and ECS** step, select a namespace from the **Namespace** drop-down list and select a cluster from the **Select Cluster to Import** drop-down list. In the instance list, select an ECS instance and click **Next**.
5. In the **Ready to Import** step, select **I agree to convert the above instances, and fully understand that the data in the original systems will be lost after conversion**. Then, enter a **new password** for the root user, confirm the new password, and click **Confirm and Import**.
6. In the **Import** step, view the import progress. In the **Import** step, the import progress of the ECS instance is **Converting now**. This conversion might take 5 minutes. If you click "Click to return to the Cluster Details page" before the import is completed, the health check status shows **Converting** and the conversion progress is shown as a percentage. When the health check status changes to **Running**, the instance is imported.

## Result

Click **Click to return to the Cluster Details page** to go to the Cluster Details page. In the **ECS Instance** section, view the import status and progress.

### 1.4.3.2. Create an ECS cluster

Create a cluster before publishing applications.

## Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Resource Management > Clusters**.
3. On the **Clusters** page, click **EDAS Cluster**. On the **EDAS Cluster** tab page, click **Create Cluster** in the upper-right corner.
4. In the **Create Cluster** dialog box, set the cluster parameters and click **Create**. Cluster parameters

Name	Description
Cluster Name	Enter a name for the cluster. The name can only contain letters, numbers, underscores (_), and periods (.), with a length up to 64 characters.
Cluster	The options are <b>Alibaba Cloud</b> and <b>Non-Alibaba Cloud</b> . Select <b>Alibaba Cloud</b> in this case. Select <b>Non-Alibaba Cloud</b> when creating a hybrid cloud cluster.
Cluster Type	Currently, only ECS clusters are supported.
Network Type	Only VPC is supported.

Name	Description
VPC	Select a specific VPC.
Namespace	A namespace has been selected on the <b>Clusters</b> page, so this parameter cannot be set here.

## Result

After the cluster is created, the message **Cluster created successfully** appears in the upper-right corner of the page, and the cluster appears in the cluster list and is in the **Normal** state.

## What's next

Add ECS instances after the cluster is created.

1. On the Cluster Details page, click **Add ECS Instance** in the upper-right corner.
2. On the **Add ECS Instance** page, click **Import ECS** or **From Existing Cluster** to add ECS instances.
  - **Import ECS**: See [Import ECS instances](#).
  - **From Existing Cluster**: In the current region, select a **namespace** and **source cluster**. In the ECS instance list, select ECS instances and click **>** to add them to the field on the right. Then, click **Next**. The subsequent procedure is the same as that for importing ECS instances.
3. After ECS instances are added, return to the Cluster Details page to view the health status of the ECS instances. The ECS instances are successfully added if the health status is **Normal**.

## 1.4.4. Manage Kubernetes resources

### 1.4.4.1. Import Container Service Kubernetes clusters

Container Service for Kubernetes provides enterprise-level high-performance and flexible management for Kubernetes containerized applications throughout the application lifecycle. Container Service for Kubernetes simplifies the deployment and scale-out operations of Kubernetes clusters and integrates Alibaba Cloud capabilities of virtualization, storage, networking, and security. Based on these capabilities, Container Service for Kubernetes provides an improved running environment for Kubernetes containerized applications.

## Prerequisites

Before importing a Container Service Kubernetes cluster, make sure that you have created the Container Service Kubernetes cluster.

## Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Resource Management > Clusters**.
3. On the **Clusters** page, click **Container Service K8s Cluster**.
4. On the **Container Service K8s Cluster** tab, select a **region** and **namespace**. Find the Container Service Kubernetes cluster you created, and click **Import** in the **Actions** column.
5. In the **Import Kubernetes Cluster** dialog box, confirm the **namespace** for import, and then click **Import**.

## Result

When the option in the **Actions** column changes to **Cancel import** and the **cluster status** is **Running**, the cluster is imported to EDAS.

## 1.4.5. Manage resource groups

Resource groups are groups of Enterprise Distributed Application Service (EDAS) resources. You can use resource groups to control account permissions. You can grant resource group access permissions to Resource Access Management (RAM) users, and each RAM user has the permission to operate on all the resources in the specified group.

### Typical scenarios

- A company uses EDAS to create business applications. Department A is responsible for user-related applications and Department B for goods-related ones.
- The company registers an EDAS account (the Apsara Stack tenant account) to activate EDAS and creates two RAM users for Departments A and B.
- Departments A and B have dedicated Elastic Compute Service (ECS) and Server Load Balancer (SLB) instances for deploying user-related applications and goods-related applications, respectively.
- You have created two resource groups in EDAS and bound them to the resources of Departments A and B, respectively. Then, you grant the RAM users of Departments A and B the permissions to access the two resource groups, respectively.
- Department A uses its RAM user only to operate the resources in the authorized resource group. Department B does the same for its resource group. There is no conflict between Departments A and B during resource management.

### Create a resource group

1. In the left-side navigation pane of the console, choose **Resources > Resource Groups**.
2. On the **Resource Groups** page, click **Create Resource Group** in the upper-right corner.
3. In the **Create Resource Group** dialog box, enter **Resource Group Name** and **Resource Group Description**, and click **Confirm**.

After the resource group is created, you can edit or delete it as needed.

### Bind resources to resource groups

You can bind ECS instances, SLB instances, and clusters to resource groups. The procedures for binding different types of resources are similar. This topic describes how to bind ECS instances.

1. On the **Resource Groups** page, find the target resource group, and click **Bind ECS** in the **Actions** column.
2. In the **Bind ECS** dialog box, select one or more ECS instances and click **Confirm**.

### Grant RAM users the permissions to access resource groups

You can grant RAM users the permissions to access specified resource groups.

1. Log on to the EDAS console by using your Apsara Stack tenant account.
2. In the left-side navigation pane, choose **System Management > Sub-Accounts**.
3. Find the target user and click **Resource Group Permission** in the **Actions** column.
4. In the **Resource Group Permission** dialog box, select a resource group and click **Confirm**.

## 1.5. Application management for ECS clusters

### 1.5.1. Application deployment

#### 1.5.1.1. Deploy applications in the console

##### 1.5.1.1.1. Deploy web applications in ECS clusters

In an ECS cluster, an ECS instance can only deploy one application. This topic describes how to create a Java web application that only contains a welcome page, and use a WAR package to deploy, update, view, and manage the application in the Enterprise Distributed Application Service (EDAS) console.


### Prerequisites

Prerequisites

- You have activated EDAS.
- You have created a VPC.
- (Optional) You have created a namespace.
- You have created an ECS cluster.

### Procedure

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and a **namespace** (optional). Click **Create Application**.
4. On the **Application Information** page that appears, enter the application information. After setting the parameters, click **Create an Empty Application**. This creates an application without any instance. Click **Next**. On the Application Configuration page, set the required parameters.
  - **Namespace**: Select a **region** from the left-side drop-down list and select a **namespace** from the right-side drop-down list. If you do not select a namespace, the **default** one is used.
  - **Cluster Type**: Select **ECS Cluster** from the left-side drop-down list, and select a specific ECS cluster from the right-side drop-down list.
  - **Application Name**: The name of the application.
  - **Deployment Method**: After selecting an ECS cluster, you can deploy the application through a **WAR package** or a **JAR package**.
  - **Application Runtime Environment**:
    - **Deploy applications by using a WAR package**:
      - If you want to create a native Spring Cloud or Dubbo application, select **apache-tomcat**.
      - If you want to create an HSF application, select **EDAS-Container**.

- **Deploy applications by using a JAR package:**
    - If you want to create a native Spring Cloud or Dubbo application, select **Default Environment**.
    - If you want to create an HSF application, select **EDAS-Container**.
  - **Java Environment:** Select **Open JDK 8** or **Open JDK 7**.
  - **Application Description:** Enter the basic information of the application. The maximum length of the description is 128 characters.
5. On the Application Configuration page, add an instance and configure the instance as instructed. After configuring the instance, click **Create**.
- **Instance Source:** In ECS clusters, you can add an instance in any of the following three methods. If you do not select any instance, click **Create an Empty Application** to create an application that contains no instances. Then, add instances to the application through **Scale out and scale in applications (ECS clusters)** and deploy the application.
    - **Select an instance from the cluster:** Click **Add** next to **Available Instances**. On the **Instances** page, select an idle instance from the cluster of the application, click **>** to add the instance to the **Selected Instances** area, and then click **OK**.
    - **Create an instance based on the existing instance specifications:**
      - a. Click **Host Selection** next to **Template Host**.
      - b. In the **Template Host** dialog box, select any instance in the cluster and use it as the template. Click **Recycling Mode**, and then click **OK** in the lower-right corner.
      - c. On the **Application Configuration** tab page, configure the **password** and **purchase quantity**. Then, select **ECS Service Terms | Image Service Terms**.
    - **Create an instance from a template:**
      - a. Click **Select Template** next to **Launch Template**.
      - b. In the **Select Template to Be Launched** dialog box, select the template based on which the instance is created and the template version, select **Recycling Mode**, and then click **OK** in the lower-right corner.
      - c. On the **Application Configuration** tab page, configure the **purchase quantity** of the instance, and select **ECS Service Terms | Image Service Terms**.
  - **Deploy Now:** This option is available only after you have selected an instance. Turn on **Deploy Now** and configure the instance as instructed.
  - **File Uploading Method:** Select **Upload WAR Package** or **WAR Package Location**.
    - **Upload WAR Package:** Click **Select File** and select the target WAR package.
    - **WAR Package Location:** Copy the storage path of the WAR package and paste the path to the WAR package location bar.
-  **Note** The name of the application deployment package can only contain letters, numbers, hyphens (-), and underscores (\_). The JAR package can be uploaded only when the JAR package deployment method is selected. Otherwise, you can only deploy the application by using the WAR package.
- **Version:** Specify the version, for example, *1.1.0*. We recommend that you do not use the timestamp as the version number.

- **Application Health Check:** (Optional) Specify a URL for application health check. The system checks the health of the application after the container is started or is running. Then, it performs a service routing task based on the health check result. A sample URL is [http://127.0.0.1:8080/\\_etc.html](http://127.0.0.1:8080/_etc.html).
- **Batch:** Specify the number of batches. You can specify the number of batches and publish the application to the selected instances in batches only when two or more instances are selected.
- **Batch Mode:** Select **Automatic** or **Manual**. When you select Automatic, you need to specify **Batch Wait Time**, which is the interval between different application deployment batches.

## Result

Wait several minutes until the application is created. After the application is created, you can view the application information on the Application Details page. On the Application Details page, click the **Instance Information** tab. On the Instance Information tab page, view the instance running status. If Running Status/Time is **Running**, the application is published.

## 1.5.1.2. Use CLI to deploy applications

### 1.5.1.2.1. Use toolkit-maven-plugin to automatically deploy applications

Previously, EDAS applications had to be deployed according to the step-by-step instructions in the console. To improve the developer experience, toolkit-maven-plugin has been provided for automatic application deployment. You can use toolkit-maven-plugin to automatically deploy applications that are developed based on the HSF, Dubbo, or Spring Cloud framework in ECS or Swarm clusters.

## Automatically deploy applications

1. Add the following plug-in dependencies to the pom.xml file in your packaged project.

```
<build>
  <plugins>
    <plugin>
      <groupId>com.alibaba.cloud</groupId>
      <artifactId>toolkit-maven-plugin</artifactId>
      <version>1.0.2</version>
    </plugin>
  </plugins>
</build>
```

2. Create a file named `.edas_config.yaml` in the root directory of the packaged project. If the packaged project is a Maven submodule, create the file in the submodule directory.


```
env: region_id: cn-beijingapp: app_id: eb20****-e6ee-4f6d-a36f-5f6a5455**** endpoint: xxxxx
```

In the preceding configuration, **region\_id** indicates the ID of the region where the ECS instance that hosts the application is located. **app\_id** indicates the ID of the application. **endpoint** indicates the point of presence (POP) of EDAS in Apsara Stack. The preceding parameter values are for reference only. Replace them with your actual application parameters. For example, to obtain an **endpoint**, contact EDAS Customer Services. For more information about configuration items, see [More configuration items](#).

To obtain the values of these configuration items, perform the following steps:

- i. Log on to the EDAS console.
  - ii. In the left-side navigation pane, choose **Application Management**. On the Applications page, locate the row that contains the target application and click the name of the application. On the Application Management page, click **Deploy Application**.
  - iii. On the **Deploy Application** page, click **Generate Maven Plug-in Configuration** to obtain the parameter values.
3. Create an account file and configure the AccessKey ID and AccessKey Secret in yaml format. Obtain the AccessKey ID and AccessKey Secret on the [User Info](#) page in the Alibaba Cloud console. We recommend that you use a RAM user that has been granted **application management** permissions to improve the application security. The following provides a configuration example:

```
access_key_id: abcaccess_key_secret: 1234567890
```

 **Note** In the preceding configuration, abc and 1234567890 are for reference only. Replace them with your actual AccessKey ID and AccessKey Secret. In this configuration, the AccessKey ID and AccessKey Secret are only used to generate request signatures and not for any other purposes, such as network transfers.

4. Go to the root directory (or the submodule directory if multiple Maven modules exist) and run the following packaging command:

```
mvn clean package toolkit:deploy -Daccess_key_file={account file path}
```

The preceding parameters are described as follows:

- o **toolkit:deploy**: Use toolkit-maven-plugin to deploy the application after it is packaged successfully. The application is deployed only when this parameter is configured.
  - o **access\_key\_file**: The file of the Alibaba Cloud account. For more information about how to specify a key pair, see [Account configuration](#).
5. After you run the preceding command, you have successfully deployed the application with toolkit-maven-plugin.

## More configuration items

Configuration items for deploying applications are classified as follows:

- Basic environment variables (env)
- Application configuration items (app)
- Storage configuration items (oss)

The configuration items currently supported are listed in the following table.

Type	Parameter	Required	Note
env	region_id	Yes	The ID of the region where the application is located.
	endpoint	No	The POP of the application.

Type	Parameter	Required	Note
app	app_id	Yes	The ID of the application.
	package_version	No	The version of the deployment package. The default value is the string of the pom.xml file version plus the instance creation time, for example, "1.0 (2018-09-27 19:00:00)".
	desc	No	The deployment description.
	group_id	No	The ID of the group to which the application is deployed. The default value is All Groups.
	batch	No	The number of deployment batches. The default value is 1 and the maximum value is 5.
	batch_wait_time	No	The waiting time (in minutes) between deployment batches. The default value is 0.
	stage_timeout	No	The timeout period (in minutes) for each change stage. The default value is 5. If batch_wait_time is set, it is automatically counted with this parameter during calculation. During runtime, if a stage waits for a time longer than this threshold value, the plug-in automatically exits.
	region_id	No	The ID of the region where the target bucket is located. The default value is the ID of the region where the application is located.

Type	Parameter	Required	Note
OSS	bucket	No	The name of the target bucket. The default value is the free OSS bucket provided by EDAS. If OSS configuration items are specified, you must specify the bucket parameter. Otherwise, the instances use the free OSS bucket automatically allocated by EDAS.
	key	No	The custom path used to upload the application package to OSS. The instances use the free OSS bucket provided by EDAS by default. If you use a specified OSS bucket, specify the package storage path in this parameter and use the {region_id}, {app_id}, and {version} variables to set the path through parameters, for example, pkgs/petstore/{version}/store.war. The default value is {region_id}/{app_id}/{version}.
	access_key_id	No	The custom account ID that is used to upload the application package to OSS.
	access_key_secret	No	The custom account key that is used to upload the application package to OSS.

### Configuration example 1: Specify the group and the deployment package version

Assume that you want to deploy application `eb20dc8a-e6ee-4f6d-a36f-5f6a545****` to group `06923bb9-8c5f-4508-94d8-517b692f****` in China (Beijing). The version of the deployment package is 1.2. In this case, the configuration is as follows:

```
env: region_id: cn-beijing app: app_id: eb20dc8a-e6ee-4f6d-a36f-5f6a5455**** package_version: 1.2 group_id: 06923bb9-8c5f-4508-94d8-517b692f****
```

### Configuration example 2: Specify an OSS bucket

Assume you want to deploy an application whose ID is `eb20dc8a-e6ee-4f6d-a36f-5f6a5455****` and upload the deployment package to your own bucket named `release-pkg` in China (Beijing). The file object name is `my.war`, the ID of the OSS account is `ABC`, and the key of the OSS account is `1234567890`. In this case, the configuration is as follows:

```
env: region_id: cn-beijing app: app_id: eb20dc8a-e6ee-4f6d-a36f-5f6a5455**** oss: region_id: cn-beijing bucket: release-pkg key: my.war access_key_id: ABC access_key_secret: 1234567890
```

## Configuration file

- When no configuration file is specified, the plug-in uses the `.edas_config.yaml` file in the root directory of the packaged project as the configuration file by default. If the packaged project is a submodule of the Maven project, the configuration file is in the root directory of the submodule by default but not the root directory of the entire Maven project.
- You can also specify a configuration file by setting the `-Dedas_config=xxx` parameter.
- If the default configuration file exists but another configuration file is specified using the parameter, the plug-in uses the latter.

## Account configurations and priorities

When using this plug-in to deploy applications, you must provide the AccessKey ID and AccessKey Secret of an Alibaba Cloud account for application deployment. Currently, the plug-in supports multiple configuration methods. If duplicate configurations exist, the configuration method with the higher priority overrides that with the lower priority. Configuration methods are listed as follows in descending order of priority:

- **Specify the AccessKey ID and AccessKey Secret in the CLI:** You can specify the AccessKey ID and AccessKey Secret in either of the following ways:
  - If you package the project by running Maven commands, specify both parameters with `-Daccess_key_id=xx -Daccess_key_secret=xx`.
  - When you configure this plug-in in the `pom.xml` file, configure both parameters as follows:

```
<plugin> <groupId>com.alibaba.cloud</groupId> <artifactId>toolkit-maven-plugin</artifactId> <version>1.0.2</version><configuration> <accessKeyId>abc</accessKeyId> <accessKeySecret>1234567890</accessKeySecret></configuration></plugin>
```

- **Specify the account file in the CLI (recommended):** When you package the project by running Maven commands, specify the account file in yaml format with `-Daccess_key_file={account file path}`. For example:

```
access_key_id: abc access_key_secret: 1234567890
```

- **Use the default Alibaba Cloud account file:** If you choose not to specify an account in either of the preceding ways, the plug-in uses the Alibaba Cloud account you set previously to deploy the application.

- **aliyuncli**: If you have used the latest Alibaba Cloud CLI and configured your Alibaba Cloud account, Alibaba Cloud generates the `.aliyuncli` directory in the current Home directory and creates the `credentials` file in the `.aliyuncli` directory to store your account information. Here, the MacOS system is used as an example. Assume that the system user is jack. Then, the following information is stored in the `/Users/jack/.aliyuncli/credentials` file:

```
[default]aliyun_access_key_secret = 1234567890aliyun_access_key_id = abc
```

This plug-in uses this account file as the account for deploying the application.

- **aliyun**: If you have used a legacy Alibaba Cloud CLI and configured the Alibaba Cloud account, the Alibaba Cloud CLI generates the `.aliyun` directory in the current Home directory and creates the `config.json` file in the `.aliyun` directory. Here, the MacOS system is used as an example. Assume that the system user is jack. Then, the following information is stored in the `/Users/jack/.aliyun/config.json` file:

```
{ "current": "", "profiles": [{ "name": "default", "mode": "AK", "access_key_id": "", "access_key_secret": "", "sts_token": "", "ram_role_name": "", "ram_role_arn": "", "ram_session_name": "", "private_key": "", "key_pair_name": "", "expired_seconds": 0, "verified": "", "region_id": "", "output_format": "json", "language": "en", "site": "", "retry_timeout": 0, "retry_count": 0 }, { "name": "", "mode": "AK", "access_key_id": "abc", "access_key_secret": "xxx", "sts_token": "", "ram_role_name": "", "ram_role_arn": "", "ram_session_name": "", "private_key": "", "key_pair_name": "", "expired_seconds": 0, "verified": "", "region_id": "cn-hangzhou", "output_format": "json", "language": "en", "site": "", "retry_timeout": 0, "retry_count": 0 }], "meta_path": "" }
```

- **System environment variables**: Then, the plug-in attempts to retrieve the values of `access_key_id` and `access_key_secret` from system environment variables. In other words, the plug-in retrieves the values from `System.getenv("access_key_id")` and `System.getenv("access_key_secret")`.

## 1.5.1.2.2. Use CLI to deploy applications in EDAS

The command line interface (CLI) was the most widely used type of user interface before graphical user interfaces (GUIs) become popular. CLIs usually do not support the use of a mouse. Instead, you enter instructions through a keyboard, and the computer receives and runs the instructions. By using the CLI, you can accurately control the system and efficiently and reliably perform complex operations.

### Prerequisites

Before performing the steps in this tutorial, you must have done the following: [Import ECS instances](#)

### Context

Alibaba Cloud CLI is an open source tool built on the Go SDK provided by Alibaba Cloud. Alibaba Cloud CLI can directly call the EDAS API. Make sure that you have activated EDAS and know how to use SDKs to call operations in EDAS. For more information about how to call operations, see *Developer Guide*. You can use Alibaba Cloud CLI to deploy all applications developed based on the HSF, Dubbo, or Spring Cloud framework in ECS or Swarm clusters in EDAS.

### Procedure

#### 1. Install CLI

Alibaba Cloud CLI is available after you download and decompress it. It is supported on MacOS, Linux, and Windows (64-bit) clients. Download the appropriate installation package:

- [MacOS](#)
- [Linux](#)

- o Windows (64-bit)

After decompressing the installation package, move the *aliyun* file to the */usr/local/bin* directory or add it to the `$PATH` environment variable.

## 2. Configure CLI

Before using Alibaba Cloud CLI, run the `aliyun configure` command to configure the AccessKey, region, and language for calling your Alibaba Cloud account.

You can create and view your AccessKey on the [Security Management](#) page, or obtain the AccessKey from your system administrator.

```
$ aliyun configureConfiguring profile 'default' ...Aliyun Access Key ID [None]: <Your AccessKey ID>Aliyun Access Key Secret [None]: <Your AccessKey Secret>Default Region Id [None]: cn-hangzhouDefault output format [json]: jsonDefault Language [zh]: zh
```

## 3. Use CLI to create applications

Run the following script to create an application:

```
#!/bin/bash # Region for deployment REGION="cn-beijing" # ID of the ECS instance ECS_ID="i-2z*****b6" # ID of the VPC where the ECS instance is located VPC_ID="vpc-t*****c" # Name of a namespace (which is automatically created if it does not exist) NAMESPACE="myNamespace" # Name of a cluster (which is automatically created) CLUSTER_NAME="myCluster" # Name of an application APP_NAME="myApp" # Step 1: Create a namespace. aliyun edas InsertOrUpdateRegion --RegionTag $REGION:$NAMESPACE --RegionName $NAMESPACE --region $REGION --endpoint "edas.cn-beijing.aliyuncs.com" >> /dev/null # Step 2: Create a cluster. CLUSTER_ID=`aliyun edas InsertCluster --ClusterName $CLUSTER_NAME --ClusterType 2 --NetworkMode 2 --VpcId $VPC_ID --logicalRegionId $REGION:$NAMESPACE --region $REGION --endpoint "edas.cn-beijing.aliyuncs.com" | sed -E 's/. *"ClusterId":("[a-z0-9-"]*)"/\1/g` # Step 3: Convert the ECS instance (which takes some time). aliyun edas TransformClusterMember --InstanceId $ECS_ID --TargetClusterId $CLUSTER_ID --Password Hello1234 >> /dev/null for i in `seq 300` do OUT=`aliyun edas ListClusterMembers --ClusterId $CLUSTER_ID | grep Eculd` && break sleep 1 done ECU_ID=`echo $OUT | sed -E 's/. *"Eculd":("[a-z0-9-"]*)"/\1/g` # Step 4: Create an application. APP_ID=`aliyun edas InsertApplication --ApplicationName $APP_NAME --BuildPackId 51 --EcuInfo $ECU_ID --ClusterId $CLUSTER_ID --logicalRegionId $REGION:$NAMESPACE | sed -E 's/. *"AppId":("[a-z0-9-"]*)"/\1/g` printf "An application is created by CLI, App ID:"$APP_ID"\n"
```

## 4. Use CLI to deploy applications

Run the following code to use Alibaba Cloud CLI to deploy an application:

```
#!/bin/bash # ID of the application to be deployed (which must be created in advance) APP_ID="87a6*****4d1" # ID of the group to which the application belongs GROUP_ID="54b*****f27" # Name of the OSS bucket for uploading (the bucket must support public read) OSS_BUCKET="eda*****mo" # Installation package file (created by your CI system) PACKAGE="hello-edas.war" # Step 1: Upload the deployment package to OSS. aliyun oss cp -f $PACKAGE oss://$OSS_BUCKET/$PACKAGE >> /dev/null PKG_URL=`aliyun oss sign oss://$OSS_BUCKET/$PACKAGE | head -1` # Step 2: Initiate a deployment request. CO_ID=`aliyun edas DeployApplication --AppId $APP_ID --PackageVersion $VERSION --DeployType url --WarUrl "${PKG_URL}" --GroupId $GROUP_ID | sed -E 's/. *"ChangeOrderId":("[a-z0-9-"]*)"/\1/g` # Step 3: Wait until the application is deployed. for i in `seq 300` do STATUS=`aliyun edas GetChangeOrderInfo --ChangeOrderId $CO_ID | sed -E 's/. *"Status":(.)"/\1/g` [ 2 = ${STATUS} ] && break sleep 1 done
```

In the preceding configuration items, APP\_ID and GROUP\_ID are two configuration parameters of the application. All parameters in the preceding code are for reference only. Replace them with the actual values.

To obtain the values of these configuration items, perform the following steps:

- Log on to the EDAS console.

- ii. In the left-side navigation pane, choose **Application Management**. On the Applications page, locate the row that contains the target application and click the name of the application. On the Application Management page, click **Deploy Application**.
- iii. On the **Deploy Application** page, click **Generate Maven Plug-in Configuration** to retrieve the parameter values.

### 1.5.1.2.3. Use Alibaba Cloud Toolkit for Eclipse to deploy applications


Alibaba Cloud Toolkit (hereinafter referred to as "Cloud Toolkit") is a free IDE plug-in that helps users use Alibaba Cloud more efficiently. You only need to register or use an existing Alibaba Cloud account to download Cloud Toolkit for free. After the plug-in is downloaded, you can install it to Eclipse. You can use Cloud Toolkit to automatically deploy applications that are developed based on the HSF, Dubbo, or Spring Cloud framework in ECS or Swarm clusters. This topic describes how to install Cloud Toolkit to Eclipse and use Cloud Toolkit to deploy an application in EDAS.

#### Prerequisites

- You have downloaded and installed **JDK 1.8 or later**.
- You have downloaded and installed **Eclipse IDE 4.5.0 (code: Mars) or later**. The program must be suitable for Java EE developers.

#### Install Cloud Toolkit

1. Start Eclipse.
2. In the top navigation bar, choose **Help > Install New Software**.
3. In the **Available Software** dialog box, set **Work with** to the URL <http://toolkit.aliyun.com/eclipse/> of Cloud Toolkit for Eclipse.
4. In the Name section, select **Alibaba Cloud Toolkit Core** and **Alibaba Cloud Toolkit Deployment Tools**. In the Details section, clear **Connect all update sites during install to find required software**. Then, click **Next**.
5. Perform the subsequent steps as instructed on the Install page of Eclipse.

 **Note** During the installation process, a dialog box indicating no digital signature may appear. In this case, click **Install anyway**.

6. After Cloud Toolkit is installed, restart Eclipse. Then, the Alibaba Cloud Toolkit icon appears in the toolbar.

#### Configure Cloud Toolkit

1. Start Eclipse.
2. Set the AccessKey ID and AccessKey Secret.
  - i. In the toolbar, click the drop-down arrow of the Alibaba Cloud Toolkit icon. In the drop-down list, select **Alibaba Cloud Preference...**
  - ii. In the **Preference (Filtered)** dialog box, choose **Accounts** from the left-side navigation pane.

- iii. On the **Accounts** page, set **Access Key ID** and **Access Key Secret**, and click **OK**.

 **Note**

If you use the **AccessKey ID** and **AccessKey Secret** of a RAM user, make sure that the RAM user has the permission to **deploy applications**.

- If you already have an Alibaba Cloud account, on the **Accounts** page, click **Manage existing Account** to go to the logon page of Alibaba Cloud. After you log on to the system with an existing account, you are redirected to the Security Management page. On this page, obtain the **AccessKeyId** and **AccessKeySecret** of the account.
- If you do not have an Alibaba Cloud account, on the **Accounts** page, click **Sign up**. You are redirected to the Register account page of Alibaba Cloud. On this page, register an Alibaba Cloud account. Then, obtain the **AccessKeyId** and **AccessKeySecret** of the account.

3. Set an endpoint.

- i. In the **Preference (Filtered)** dialog box, choose **Appearance & Behavior > Endpoint** from the left-side navigation pane.
- ii. On the **Endpoint** page, set an endpoint and click **Apply and Close**.



**Note** To obtain an endpoint, contact EDAS Customer Services.

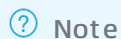
## Deploy applications to EDAS

Currently, you can use Cloud Toolkit to deploy applications to EDAS by using WAR or JAR packages.

1. In the **Package Explorer** left-side navigation pane of Eclipse, right-click your application project and choose **Alibaba Cloud > Deploy to EDAS** from the shortcut menu.
2. In the **Deploy to EDAS** dialog box, select **Region**, **Namespace**, **Application**, **Group**, and **Deploy File** as needed. Then, click **Deploy**.

Parameters for deploying an application to EDAS:

- **Region**: The region where the application is located.
- **Namespace**: The namespace where the application is located.
- **Application**: The name of the application.
- **Group**: The group of the application.



**Note** If you have not created an application in EDAS, click **Create application on EDAS console** in the upper-right corner of the dialog box to go to the EDAS console and create an application.

3. When the deployment process starts, the deployment logs are printed on the **Console** tab of Eclipse. You can view the deployment result based on the logs.

## Stop Cloud Toolkit


If you want to stop Cloud Toolkit, end the **EDAS-deploy** process on the **Progress** tab.

### 1.5.1.2.4. Use Alibaba Cloud Toolkit for IntelliJ IDEA to deploy applications

Alibaba Cloud Toolkit for IntelliJ IDEA (hereinafter referred to as "Cloud Toolkit") is a free IDE plug-in that helps users use Alibaba Cloud more efficiently. You only need to register or use an existing Alibaba Cloud account to download Cloud Toolkit for free. After the plug-in is downloaded, you can install it to IntelliJ IDEA. You can use Cloud Toolkit to automatically deploy applications that are developed based on the HSF, Dubbo, or Spring Cloud framework in ECS or Swarm clusters. This topic describes how to install Cloud Toolkit in IntelliJ IDEA and how to use Cloud Toolkit to deploy an application in EDAS.

## Prerequisites

- You have downloaded and installed **JDK 1.8 or later**.
- You have downloaded and installed **IntelliJ IDEA (2018.3 or later)**.

 **Note** The official server of the JetBrains plug-in is deployed outside China. If you cannot download IntelliJ IDEA due to a slow network response, join the discussion group provided at the end of this topic to obtain the offline installation package for IntelliJ IDEA from Cloud Toolkit Customer Services.


## Install Cloud Toolkit

1. Start IntelliJ IDEA.
2. Install Cloud Toolkit to IntelliJ IDEA.
  - **MacOS system:** On the **Preferences** page, choose **Plugins** from the left-side navigation pane. Search for **Alibaba Cloud Toolkit** and then click **Install**.
  - **Windows system:** Go to the **Plugins** page. Search for **Alibaba Cloud Toolkit** and then click **Install**.
3. After Cloud Toolkit is installed to IntelliJ IDEA, restart IntelliJ IDEA. The Alibaba Cloud Toolkit icon appears in the toolbar.

## Configure Cloud Toolkit


After Alibaba Cloud Toolkit is installed, use the AccessKey ID and AccessKey Secret to configure the Cloud Toolkit account.

1. Start IntelliJ IDEA.
2. Set the AccessKey ID and AccessKey Secret.
  - i. Click the Alibaba Cloud Toolkit icon and select **Preferences** from the drop-down list. On the Settings page, choose **Alibaba Cloud Toolkit > Accounts** from the left-side navigation pane.
  - ii. On the **Accounts** page, set **Access Key ID** and **Access Key Secret**, and click **OK**.

 **Note** If you use the AccessKey ID and AccessKey Secret of a RAM user, make sure that the RAM user has the permission to **deploy applications**.

- If you already have an Alibaba Cloud account, on the **Accounts** page, click **Get existing AK/SK** to go to the logon page of Alibaba Cloud. After you log on to the system with an existing account, you are redirected to the Security Management page. On this page, obtain the **AccessKeyId** and **AccessKeySecret** of the account.
- If you do not have an Alibaba Cloud account, on the **Accounts** page, click **Sign up**. You are redirected to the Register account page of Alibaba Cloud. On this page, register an Alibaba Cloud account. Then, obtain the **AccessKeyId** and **AccessKeySecret** of the account.


3. Set an endpoint.
  - i. On IntelliJ IDEA, click the Cloud Toolkit icon and select **Preferences** from the drop-down list.
  - ii. In the **Preferences** dialog box, choose **Appearance & Behavior > Endpoint** from the left-side navigation pane.
  - iii. On the **Endpoint** page, set the endpoint of EDAS and click **Apply**.

 **Note** To obtain an endpoint, contact EDAS Customer Services.

## Deploy applications to EDAS

Currently, you can use Cloud Toolkit to deploy applications to EDAS by using WAR or JAR packages.

1. On IntelliJ IDEA, click the Alibaba Cloud Toolkit icon and select **EDAS on Alibaba Cloud** from the drop-down list.
2. In the **Deploy to EDAS** dialog box, configure the application deployment parameters. Then, click **Apply** to save the configurations.
  - i. In the **Deploy to EDAS** dialog box, select **Region**, **Namespace**, **Application**, and **Group** in the **Application** section as needed.
    - **Region**: The region where the application is located.
    - **Namespace**: The namespace where the application is located.
    - **Application**: The name of the application.
    - **Group**: The group of the application.
  - ii. Set the build mode.
    - **Maven Build**: If this option is selected for building the application, the system adds a Maven task by default to build the deployment package.
    - **Upload File**: If this option is selected for building the application, upload the WAR package or JAR package, and then deploy the application.

 **Note** If you have not created an application in EDAS, click **Create application on EDAS console** in the upper-right corner of the dialog box to go to the EDAS console and create an application.

3. Click **Run** to run the configurations you made in the preceding step. The deployment logs are printed on the **Console** tab of IntelliJ IDEA. You can view the deployment result based on the logs.

## Manage Maven tasks

In Cloud Toolkit installed in IntelliJ IDEA, you can deploy Maven tasks. In the **Deploy to EDAS** dialog box, you can also add, delete, modify, or move Maven tasks in the **Before launch** section.

In the **Select Maven Goal** dialog box, click the folder icon on the right of the **Working directory** field and select all available modules for the current project. Enter the building command in the **Command line** field.

## Deploy multi-module projects

Most Maven projects involve multiple modules. These modules can be separately developed and some of them may use the functions of other modules. This type of project is a multi-module project.

If your project is a Maven multi-module project and you want to deploy a submodule in the project, make sure that the last Maven task in the **Before launch** section in the **Deploy to EDAS** dialog box is built for the submodule. For more information about how to manage Maven tasks, see [Manage Maven tasks](#).

For example, the CarShop project has the following submodules:

- carshop
  - itemcenter-api
  - itemcenter
  - detail

Itemcenter and detail are submodules and depend on the itemcenter-api module. In this case, how is the itemcenter submodule deployed? In the **Before launch** section of the Deploy to EDAS dialog box, add the following two Maven tasks:

1. Add a Maven task to run the `mvn clean install` command in the carshop parent project.
2. Add a Maven task to run the `mvn clean package` command in the itemcenter submodule.

### 1.5.1.3. Canary release for ECS clusters

To upgrade Spring Cloud or Dubbo microservice-oriented applications deployed in an Elastic Compute Service (ECS) cluster, you can use canary release to perform small-scale verification first, and then a full upgrade if the verification succeeds.


#### Limits

- High-Speed Service Framework (HSF) applications: Canary release is not supported.
- Dubbo applications: You can implement canary releases of Dubbo applications without limits.
- Spring Cloud applications: If you use `Deployment.Metadata.Name` or `Deployment.Metadata.Uid` to configure some features of an application, do not implement a canary release for the application. Otherwise, the native features of the application may be abnormal after the canary release.

#### Procedure

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and a **Namespaces**, and then click the application name.
4. On the Application Details page, click **Deploy Application** in the upper-right corner.
5. On the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner of the **Canary Release (Grayscale)** section.
6. On the **Canary Release** page, upload the deployment package of the new application version, set the canary release policy and rules, and then click **Confirm**.
  - i. Upload the deployment package for the new application version.

- ii. In the **Release Strategy** section, configure release policy parameters.  
The **Release strategy configuration information** section on the right shows the canary release process based on the configuration.  
Parameter description:
  - **Grayscale Grouping**: the group of the instance for canary release.
  - **Each Batch after Grayscale**: indicates that the application instances in other groups are released based on preset batches after canary release is completed.
    - If all groups are selected, instances in each group are released based on the selected batch number. If the number of instances in a group is less than the selected batch number, the instances in the group are released based on the actual number of batches.
    - If a specific group is selected, instances in the specified group of the application are released based on the selected batch number.
  - **Post-gray Scale Batching**: When **Each Batch after Grayscale** is set to 2 or more, you need to configure this parameter. Valid values are *Automatic* and *Manual*.
    - *Automatic*: automatically releases instances in batches based on the release interval. In Automatic mode, you need to configure **Batch Wait Time**. Valid values of **Batch Wait Time** are *Do Not Wait* and 1-5 minutes.
    - *Manual*: manually triggers the release of the next batch.
  - **Java Environment**: the runtime environment of this application. Valid values are OpenJDK 8 and OpenJDK 7.
- iii. Set the canary release rule.  
Currently, you can select either of the two canary release rules for inbound traffic: **Grayscale by Content** and **Proportional Grayscale**.
  - **Grayscale by Content**: Click **Create New of Entrance Flow Rules**, and set an inbound traffic rule.

 **Note** You can create multiple inbound traffic rules.

Parameter description:

- **Protocol Type**: Valid values are *Spring Cloud* and *Dubbo*. You can select a protocol type as needed.
  - *Spring Cloud*: You need to configure **path**.
  - *Dubbo*: You need to configure **Service** and **Method**.
- **Scene Rules**: Valid values are **At the Same Time Meet the Following Conditions** and **Meet Any of the Following Conditions**.
- **Conditions List**: The conditions for *Spring Cloud* and *Dubbo* are different. Valid values are **Cookie**, **Header**, and **Parameter**. Select a condition as needed.
  - *Spring Cloud*: Valid values are *Cookie*, *Header*, and *Parameter*. Select a condition as needed.
  - *Dubbo*: Configure **Parameters** and **Parameters Value Gets the Expression** to actual ones.
- **Proportional Grayscale**: Set **Traffic Ratio**. Traffic will be forwarded to the current canary instance group according to the configured proportion.

After canary release is started, the application of the new version is deployed first in the specified canary instance group. On the **Basic Information** tab, the message **A change process is being executed for this application. Status: Executing** appears. Click **View Details**. On the **Change Details** page that appears, view the deployment progress and status.

**Abort Change:** The application is in a canary release and the change has been terminated. Please roll back the app before doing anything else. appears.

7. Monitor whether the canary traffic meets expectations.
8. After the canary traffic verification is completed, click **Start the Next Batch** on the **Change Details** page. Complete the subsequent batch release. If any problems are found during the verification process, you can click in the upper-right corner of the **Change Details** page. After the change is terminated, on the **Basic Information** tab, the message **The application is in a canary release and the change has been terminated. Please roll back the app before doing anything else.** appears.

## Verify the result

After a canary release, check whether the version in **Deployment Package** is the new version on the **Basic Information** tab. On the **Instance Information** page, check whether the value of **Running Status** is **Running**.

### 1.5.1.4. Deploy applications in hybrid clouds

EDAS provides complete solutions for scaling, networking, and central management in hybrid clouds, allowing you to deploy applications in hybrid cloud environments. You can connect instances from Alibaba Cloud, on-premises IDCs, and other cloud service providers (CSPs) through leased lines, and add the instances to hybrid cloud (non-Alibaba Cloud) ECS clusters in EDAS. Then, you can deploy and manage HSF, Dubbo, and Spring Cloud applications in the EDAS console in a unified manner. EDAS supports the auto scaling of ECS instances in Alibaba Cloud.

## Prerequisites

- You have created a VPC.
- You have activated Express Connect.
- You have applied for a physical connection to connect your on-premises IDC to Alibaba Cloud VPC.
- The instances in your on-premises IDC meet the following requirements:
  - Operating system: CentOS 7
  - Docker not supported
  - Hardware: no special requirements for CPU and memory

## Context

Your application system may have the following requirements or problems:

- The Alibaba Cloud traffic has a certain degree of volatility and you may face traffic spikes in special scenarios, such as flash sales. You can predict the traffic volumes in such scenarios, but deviations may exist. Since you need to buy ECS instances in advance, it is hard to control the number of needed ECS instances. Knowing when to add ECS instances is also a challenge.
- Some core business systems have high security requirements and you may want to deploy such applications in your own IDC. However, you cannot deploy and manage applications in different environments in a unified manner because instances from Alibaba Cloud, on-premises IDCs, and other CSPs cannot communicate with each other.
- Considering your business needs and availability requirements, you may want to deploy your


applications on instances from multiple CSPs, that is, in multi-cloud mode. In this mode, manual processing is required because you cannot centrally manage these applications. This often leads to misoperations.

- Connect Alibaba Cloud to on-premises IDCs or to the clouds of other CSPs through Express Connect.
- Create a hybrid cloud cluster. Then, add ECS instances from Alibaba Cloud and instances from on-premises IDCs and other CSPs to the cluster.
- Deploy your applications to instances in this cluster.

In hybrid clouds, EDAS is used in the following scenarios:

- Manage applications deployed on instances in on-premises IDCs through Alibaba Cloud. After connecting your IDC to the Alibaba Cloud VPC through a leased line, you can manage your applications in the IDC by using Alibaba Cloud EDAS.
- Scale applications deployed on instances from Alibaba Cloud in or out. EDAS supports auto scaling and helps you automatically purchase and release instances in Alibaba Cloud. You only need to associate EDAS with your billing account and do not need to buy instances in advance.
- Deploy and manage instances from other CSPs. EDAS allows you to deploy applications to instances from CSPs other than Alibaba Cloud and manage these instances in a unified manner.

This topic describes how to use Alibaba Cloud to manage applications deployed on instances in on-premises IDCs. To deploy and manage instances from other CSPs, you only need to connect the target instances to the Alibaba Cloud VPC of EDAS through a leased line. Then, you can operate and manage these instances in the same way as instances in on-premises IDCs. For more information about how to scale out applications deployed on ECS instances from Alibaba Cloud, see [Scaling \(applicable to ECS clusters\)](#).

 **Note** Currently, only EDAS Professional Edition and EDAS Enterprise Platinum Edition allow you to deploy applications in hybrid cloud environments.

## Procedure

1. Create a cluster.
  - i. Log on to the EDAS console. For more information, see [Log on to the EDAS console](#).
  - ii. In the left-side navigation pane, choose **Resource Management** > **Clusters**.
  - iii. On the **Clusters** page, select the **region** and **namespace**, and click **Create Cluster**.
  - iv. In the **Create Cluster** dialog box, enter the cluster information and click **Create**.

Parameters for creating a cluster:

    - **Cluster Name**: Enter a name for the cluster. The name can only contain letters, numbers, underscores (\_), and periods (.), with a length up to 64 characters.
    - **Cluster**: Select **Non-Alibaba Cloud**.
    - **Cluster Type**: The default value is ECS, which cannot be changed.
    - **Network Type**: The default value is VPC, which cannot be changed.
    - **VPC**: From the drop-down list, select the VPC where you want to create the cluster.
    - **Namespace**: The namespace you selected for the hybrid cluster on the Clusters page, which cannot be edited.

After the cluster is created, **Created successfully** appears in the upper-right corner of the page, and the cluster appears in the cluster list.

2. Add instances to the cluster.

To add ECS instances from Alibaba Cloud and instances from on-premises IDCs and other CSPs, perform the following steps:

- i. On the Clusters page, click the name of the cluster you just created.
- ii. On the Cluster Details page, click **Add ECS Instance**.
- iii. In the **Add ECS Instance** dialog box, copy the command for installing EDAS Agent.
- iv. Use the **root** account to log on to your **Alibaba Cloud ECS instance** or the **instance in the on-premises IDC**.
- v. Paste the EDAS Agent installation command and run it.

3. Open the required ports.

To ensure that your applications in the hybrid cloud cluster can use EDAS normally, you must open the following ports after adding the instances:

- 8182: This port is used to capture infrastructure monitoring and trace monitoring logs.
- 12200 to 12300: These ports are used for Remote Procedure Calls (RPCs).
- 65000 to 65535: These are web ports.

You must open the ports based on the instance type.

- ECS instances from Alibaba Cloud: Open the ports by referring to relevant documents.
- Instances from on-premises IDCs and other CSPs: Open the ports by referring to relevant solutions.

4. Check the cluster and instance statuses.

- i. Return to the **Clusters** page. In the cluster list, locate the cluster you just created and check the values of **Status** and **Instances**.  
If the cluster status is **Normal**, the cluster is created. If the value of **Instances** is same as the number of instances you added, the instances are added successfully.
- ii. Click the cluster name. On the Cluster Details page, check the values of **Instance Name** and **Status** in the cluster information section.  
If the cluster status is **Running**, the instance is running properly.

5. Deploy an application.

Currently, the hybrid cloud cluster type can only be ECS cluster. Therefore, you can deploy applications only in hybrid cloud ECS clusters.

The method for deploying applications in hybrid cloud clusters is the same as that for deploying applications in ECS clusters. See relevant topics to deploy applications.

## Result

Wait several minutes until the application is created. After the application is created, you can view the application information on the Application Details page. On the Application Details page, click the **Instance Information** tab. On the Instance Information tab, view the instance running status. If Running Status/Time is **Running**, the application is published.

## 1.5.2. Lifecycle management for applications in ECS clusters

Applications are the basic units for EDAS management. A single application contains a group of instances on which the same application is deployed. EDAS provides a comprehensive application lifecycle management mechanism, covering the entire process from application publishing to operation, including application creation, deployment, start up, rollback, scaling, stop, and deletion. Application lifecycle management includes application publishing, management, and configuration.

- Application publishing includes application creation, deployment, start, and stop.
- Application management includes application rollback, scale-out, scale-in, and deletion and instance reset and deletion.
- Application configuration includes container, JVM parameter, SLB, and health check configuration.

 **Note**

- You can deploy, scale out, roll back, reset, and configure an application no matter if the application is running or stopped.
- After the parameters of the Tomcat container and JVM are set and saved, the related configuration files are modified. The changes take effect only after you restart the application.

## 1.5.2.1. Deploy an application (applicable to ECS clusters)

You can deploy an empty application after it is created. After the application is deployed, you can upgrade the application by redeploying the application.




### Prerequisites

- You have created an empty application. For more information, see [Deploy web applications in ECS clusters](#).
- You have created a Server Load Balancer (SLB) instance. For more information, see *SLB User Guide*.

### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, click the name of the created empty application.
4. On the Application Details page, click **Deploy Application** in the upper-right corner.
5. On the **Deploy Application** page, set deployment parameters and click **Deploy**.  
Deployment parameters:

Parameter	Description
-----------	-------------

Parameter	Description
Deployment Method	<p>Select <b>WAR</b> or <b>JAR</b>.</p> <p> <b>Note</b> Your deployment method has been determined and cannot be changed.</p> <p>The configuration processes for WAR package deployment and JAR package deployment are similar. Here, <b>WAR package deployment</b> is used as an example.</p>
File Uploading Method	<p>Select <b>Upload WAR Package</b> or <b>WAR Package Location</b>.</p> <ul style="list-style-type: none"> <li>◦ <b>Upload WAR Package</b>: Click <b>Select File</b> to the right of <b>Upload WAR Package</b>, and select a local WAR package for uploading.</li> <li>◦ <b>WAR Package Location</b>: Enter the path of the WAR package.</li> </ul> <p> <b>Note</b> The name of the application deployment package can only contain letters, digits, hyphens (-), and underscores (_).</p>
Version	<p>Enter a version number, for example, 1.1.0.</p> <p> <b>Note</b> We do not recommend that you use a timestamp as the version number.</p>
Group	Select the instance group where the application is deployed.
Batch	Specify a number of deployment batches. Select an option from the drop-down list. The options are automatically generated based on the number of instances for the application. If you select two or more batches, you must set <b>Batch Wait Time</b> .
Batch Mode	Select <b>Automatic</b> .
Java Environment (optional)	Select the runtime environment of the application from the drop-down list.

Go to the **Change Details** page to view the task progress and logs of the application deployment.

## Result

1. On the Application Details page, check whether the deployment package is of the new version.
2. Click the **Instance Information** tab to check whether **Running Status** of the ECS instance is **Normal** and whether **Change Status** is **Successful**.

## 1.5.2.2. Scale out and scale in applications (ECS clusters)

When the traffic is heavy and the application load is high, you can add Elastic Compute Service (ECS) instances to scale out an application and share the load. When the traffic is low and the load of an instance is low, you can remove the instance to scale in the application and reduce the cost.

### Scale out an application

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and **namespace**, and then click the name of the target application.
4. On the Basic Information page, click **Scale Out** in the upper-right corner.
5. On the **Scale-Out Method** tab of the **Add an Instance** dialog box, select **Target Group** and the target ECS instance, and then click **Scale Out**.

#### Note

The runtime status of the added ECS instance depends on the runtime status of the application on the instance.

- If the application is running, the added instance automatically deploys, starts, and runs the application.
- If the application is stopped, the added instance deploys the application but does not start or run the application.

### Scale in an application

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and **namespace**, and then click the name of the target application.
4. On the Basic Information page, click the **Instance Information** tab.
5. On the **Instance Information** tab, delete the target instance.
  - If the ECS instance is in the **Normal** state, click **Stop** and then **Delete**.
  - If the ECS instance is in the **Stop** state, click **Delete**.

## 1.5.2.3. Create branch versions of an application

When you create an application, Enterprise Distributed Application Service (EDAS) automatically creates an application group named "Default Group" for the application and adds the Elastic Compute Service (ECS) instances of the application to this group. You can create new groups under the default group and add some instances to the new groups. If you deploy different versions of the application on the instances in the new groups, these versions of the application are the branch versions of the application.

### Context

You can create branch versions if you have the following requirements for your application:

- Online test before you publish a new version
- A/B testing
- Canary release

## Procedure

### 1. Create a group.

- i. [Log on to the EDAS console.](#)
- ii. In the left-side navigation pane, choose **Application Management > Applications**.
- iii. On the **Applications** page, click the name of the created empty application.
- iv. On the Basic Information page, click the **Instance Information** tab. On the tab that appears, click **Create Group** in the upper-right corner.
- v. In the **Create Group** dialog box, enter a **group name** and click **Create**.

After the group is created, the message **Group created successfully** appears in the upper-right corner of the page.

### 2. Add instances to the new group.

After the group is created, you can add instances to the new group in two ways: **scale out** and **change group**. For more information about application scale-out methods, see [Scaling \(applicable to ECS clusters\)](#). This topic describes how to add instances from the default group to the new group by changing the group.

- i. On the **Instance Information** tab of the Basic Information page, find the target instance, and click **Change Group** in the Actions column.
- ii. In the **Change Group** dialog box, set **Target Group**.
- iii. Click **Confirm**.

#### Note

- If no application is deployed in the new group while an application deployment package has been deployed on the added instance, this deployment package is deployed in the group.
- If an instance is added to an existing group rather than a new group, the versions of the deployment package in the group and on the instance are different. When the system displays the following messages, select the appropriate option as needed:
  - Select **Re-deploy the current instance using the target group's version** to redeploy the deployment package on the instance using that in the group.
  - Select **Change Group Only Without Re-deployment** to add the instance without changing its deployment package.

### 3. Deploy the application in the new group.

- i. On the Basic Information page, click **Deploy Application** in the upper-right corner.
- ii. Set **Group** to the target new group, set the deployment parameters, and click **Deploy**.

## Result


On the **Instance Information** tab of the Basic Information page, you can view the deployment package version and running status of the new group to check whether the new application version is published.

## 1.5.2.4. Roll back an application

To roll back a published application to an earlier version, you can use the **application rollback** feature and select the target version.

### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, click the target application name.
4. On the **Basic Information** page, click **Roll Back** in the upper-right corner.
5. On the **Roll Back** page, select **Deployment Package Version** and **Group** for the rollback, set **Batch** and **Batch Mode**, and then click **Roll Back**.

 **Note** You can view a maximum of five deployment package versions for rollback.

## 1.5.2.5. Delete an application

After an application is deleted, all information related to the application is deleted, all instances under the application are released, and all deployment packages and container files on the instances are deleted.

### Prerequisites

Before you delete an application, be sure to save the logs, WAR packages, and configurations of all instances in the application.

### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, click the target application name.
4. On the **Basic Information** page of the application, click the **Instance Information** tab.
5. On the **Instance Information** tab, find the instance and click **Stop** in the **Actions** column.
6. After the instance status changes to **Stop**, click **Delete**.  
After the application is deleted, the message **Deleted successfully** appears in the upper-right corner of the page.

## 1.5.3. Application settings

On the Application Settings page, you can set the JVM parameters, Tomcat, SLB, and health check of applications.

### 1.5.3.1. Set JVM parameters

You can set JVM parameters to enable the container parameter setting when an application is started in Enterprise Distributed Application Service (EDAS). Correctly, setting JVM parameters helps reduce the overhead of garbage collection (GC) and shorten the server response time and improve throughput.

### Context

- If no container parameters are set, JVM parameters are allocated by default.
- The configured JVM parameters take effect only after you manually restart the application.

## Procedure

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and **namespace**, and then click the name of the target application.
4. In the **Application Settings** section of the **Basic Information** tab, click **Settings**, and select **JVM** from the drop-down list.
5. In the **Application Settings** dialog box, click to expand **Memory Configuration**, **Application**, **GC Policy**, **Tool**, and **Custom** respectively to set related parameters and then click **Save**.

### Note

- You can move the pointer over *i* to the right of a specific parameter to learn the meaning of this parameter.
- The JVM parameter settings are written in the *bin/setenv.sh* file in the container directory. To apply the settings, restart the application.

## Result

After the settings are completed, the message **JVM parameters successfully configured** appears in the upper-right corner of the page.

### 1.5.3.2. Configure Tomcat

EDAS supports Tomcat container parameter settings. You can configure settings such as the port number, application access path, and the number of connections in the connection pool of the Tomcat container in the EDAS console.

## Prerequisites


### Note

- After setting Tomcat container parameters, restart the container to apply the parameter settings.
- Tomcat container configuration is supported by EDAS Agent 2.8.0 and later.

## Procedure

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management**. On the **Applications** page, click the name of the target application.
3. On the Application Details page, click **Settings** on the right of the **Application Settings** section.
4. In the **Application Setting** dialog box, click the **Tomcat** tab and set Tomcat parameters. Then, click **Save**. Tomcat configuration description

Name	Description
Application Port	The port range is (1024, 65535). The admin authority is needed for container configuration and the root authority is required to operate on ports with numbers less than 1024. Therefore, enter a port number greater than 1024. If this parameter is not set, the default value 8080 is used.
Tomcat Context	The access path of an application. <ul style="list-style-type: none"><li>◦ If you select <b>Package Name</b>, you do not need to set a custom path. The default value is the WAR package name.</li><li>◦ If you select <b>Root</b>, you do not need to set a custom path. The default value is a slash (/).</li><li>◦ If you select <b>Custom</b>, enter a custom path below, namely, the access path. If this parameter is not set, the default value is the WAR package name.</li></ul>
Maximum Threads	The maximum number of connections in a connection pool. The parameter is maxThreads. The default value is 400. We recommend that this parameter be set under professional guidance.
Tomcat encoding	Select an encoding format for Tomcat: UTF-8, ISO-8859-1, GBK, or GB2312. The default format is ISO-8859-1. Select useBodyEncodingForURI as needed.

 **Note** Click **Advanced Settings** to configure the full text of server.xml. The application groups use the application configuration after the advanced settings are enabled.

### 1.5.3.3. Enable the Intra-zone Provider First feature

When an application calls a service, the service provider that resides in the same zone as the application is called first. This is how the Intra-zone Provider First feature works. EDAS allows you to enable the Intra-zone Provider First feature for HSF applications that are deployed in ECS clusters.

#### Scenarios

When applications are deployed in multiple zones, applications may call each other across zones. The following figure shows the call link. In this link, Application A in Zone 1 calls Application B in Zone 2. The cross-zone call increases network latency. As a result, the HTTP response time is increased.

When the Intra-zone Provider First feature is enabled for an application service, the consumer application preferentially calls the provider that is deployed the same zone. The following figure shows the call link. This link prevents network latency that is caused by the cross-zone call. This reduces the HTTP response time.

#### Limits

If you need to enable the Intra-zone Provider First feature, follow the following limits:


- Only HSF services that are deployed in ECS clusters support the Intra-zone Provider First feature. In addition, the application runtime environment must be upgraded to the latest version: EDAS Container 3.6.0.
- The Intra-zone Provider First rule takes effect only when the number of provider nodes that are deployed in the same zone is greater than 20% of the total number of nodes.

- If no providers are deployed in the current zone, a provider that is deployed in another zone is called.

## How do I enable the Intra-zone Provider First feature?

The Intra-zone Provider First feature is in public preview. To use this feature, submit a [ticket](#) to apply for the feature. Then, you can refer to the following steps to enable **Intra-zone Provider First** on the Application Details page of the service provider.

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**. In the top navigation bar, select a region and a namespace. On the **Applications** page, click the name of the HSF application that is deployed in the ECS cluster.

 **Note** On the Applications page, select the application for which **Cluster Type** is **ECS Cluster** and **Application Runtime Environment** is **EDAS-Container 3.6.0**. If your application runtime environment uses an earlier version, you can upgrade the runtime environment to use the feature. For more information, see [Container version management \(only applicable to HSF applications in ECS clusters\)](#).

3. On the **Basic Information** tab, enable **Intra-zone Provider First** in the **Application Settings** section.

When other applications call this HSF service after this feature is enabled, HSF service nodes that are deployed in the same zone are preferentially called.

### 1.5.3.4. Bind an SLB instance to EDAS

In an Elastic Compute Service (ECS) cluster, you can bind an application to a Server Load Balancer (SLB) instance to implement load balancing.

#### Scenarios

##### Dedicated SLB instance for an application

You have an application that provides order query and contains multiple ECS instances. You want the application to provide a public IP address for external access. In this case, you can bind an SLB instance to the application to achieve this purpose.

The following diagram shows the simple mapping between the SLB instance and the application in the preceding scenario.

##### Dedicated listening port for an application to distribute traffic

You have application A that provides order query and application B that provides user login. Both applications are accessed using the same public IP address and are bound to the same domain name. You can distribute traffic by binding different listening ports of the same SLB instance to the two applications.

The following diagram shows the simple mapping between the SLB instance and the application in the preceding scenario.

##### An SLB instance shared by different domain names

In the Internet scenario, port 80 is selected by default to provide the HTTP service externally. If you want to use an SLB instance to distribute traffic, a common solution is to use domain names to route data to different applications. Assume that `u.domain.com` is the domain name that is bound to the user application, and `o.domain.com` is the domain name that is bound to the order application.

The following diagram shows the simple mapping between the SLB instance and the application in the preceding scenario.

## Set SLB rules in an application group

For example, in the flash sales scenario, the number of visits to the URL (o.domain.com/orders/queryitem) that queries product information in the order system is significantly higher than that to URLs that provide other services and functions. We hope to transfer the high service traffic of the same type to a separate group of instances, which provide the order query service while instances in other groups provide other services.


The following diagram shows the topology.

## Prerequisites

- You have [Deploy web applications in ECS clusters](#) in the EDAS console.
- If you do not have an SLB instance, go to the SLB console to create an SLB instance.
- To configure forwarding rules for a deployed group, you have set an application group for the application.

## Bind an SLB instance to an application in the EDAS console

1. In the **Application Settings** section of the **Basic Information** page, click **Add** on the right of **SLB (Internet)**.

 **Note** If you have configured an SLB instance, the IP address and port number of the SLB instance are displayed. You can click **Modify** to go to the configuration page and modify the information of the SLB instance. You can also click **Unbind** to unbind the SLB instance.

2. In the **Bind SLB to Application** dialog box, select the SLB instance, and then click **Next**.
3. On the **Select configuration listener** tab, select the configuration listener protocol and listener port, and then click **Next**.
  - You can select an existing listener from **Select an existing listening port**.
  - You can also create a listener in **Add a new listening port**. For example, set the listener protocol to HTTP and the frontend port number to 82.
4. On the **Configuring virtual grouping and forwarding policies** tab, set the bound server group. Click **Next**.
  - You can choose **Default server group** to bind all the servers under this application to the default server group of the SLB instance.
  - You can also select a virtual server group from the **Existing virtual server group** list.
  - You can also enter a virtual server group name in **New virtual server group** to create a virtual server group as the bound server group.
5. On the **Confirm change SLB** tab, view the change information of the SLB instance, and click **Confirm change** to complete the configuration.

## Results

Copy the configured IP address and port number of the SLB instance such as 118.31.XXX.XXX:81, paste it in your browser address bar, and press Enter to go to the homepage of the application.

If the IP address and port number do not appear on the right side of the SLB instance, the binding failed. Go to **Change Logs** to view the change details, and troubleshoot and fix the failure based on the change logs.

## FAQ

**After I bound an SLB instance with forwarding rules to an application Group and then unbound the SLB instance from the application Group, the application cannot be accessed through SLB traffic. What can I do if the HTTP Code 503 error is reported?**

Cause: You entered the forwarding rules and bound the SLB instance to the application group, and then unbound the SLB instance in the EDAS console. In this case, EDAS did not delete the forwarding rules of the SLB instance. Since you unbound the SLB instance from the application group in the EDAS console, the servers in the virtual SLB group were also unbound. As a result, the SLB traffic forwarding failed and the HTTP 503 error was reported.

Solution: Manually delete the forwarding rules in the SLB console.

**Why does an application bound to an SLB instance fail to be accessed through the IP address of the SLB instance after traffic management is enabled?**

Cause: In this scenario, traffic management was enabled and the application was bound to an SLB instance (over HTTP). At this time, there is a limitation. When the SLB instance used HTTP to detect whether the backend nodes were alive, the message HEAD/HTTP/1.0 was sent, and Tengine responded with HTTP 400, which caused the failure of SLB listener health check. As a result, the error code 502 (bad gateway) was reported when you accessed the application.

Solution: Disable traffic management on the Application Information page of the EDAS console for applications that do not require traffic management. This will uninstall Tengine, modify the application configurations, and restart the application. To retain this function, you can select Code 4xx returned by health check in Health Check of the SLB instance.

### 1.5.3.5. Set JVM -D startup parameters

This topic describes how to set JVM -D startup parameters when developing High-Speed Service Framework (HSF) applications.

#### **-D hsf.server.port**

Specifies the port for starting HSF services. The default value is 12200. Use another port than the default port if you start multiple HSF providers locally.

#### **-D hsf.server.max.poolsize**

Specifies the maximum size of the thread pool of the HSF provider. The default value is 720 .

#### **-D hsf.server.min.poolsize**

Specifies the minimum size of the thread pool of the HSF provider. The default value is 60 .

#### **-D hsf.client.localcall**

Enables or disables the precedence of calling local HSF clients. The default value is true .

#### **-D pandora.qos.port**

Specifies the Pandora monitoring port. The default value is 12201 . Use another port than the default port if you start multiple HSF providers locally.

#### **-D hsf.http.enable**

Specifies whether to enable the HTTP port. The default value is true .

#### **-D hsf.http.port**

Specifies the HTTP port used by the HSF application to provide services externally. The default value is 12220 . Use another port than the default port if you start multiple HSF providers locally.

**-D hsf.run.mode**

Specifies whether the HSF consumer performs a targeted call, that is, bypassing Config Server. The value `1` indicates that a targeted call is disallowed, and the value `0` indicates a targeted call is allowed. The default value is `1`. Do not set this parameter to `0` unless necessary.

**-D hsf.shuthook.wait**

The wait time for gracefully disconnecting an HSF application, in ms. The default value is `10000`.

**-D hsf.publish.delayed**

Specifies whether to delay publishing all services. The default value is false, indicating not to delay service publishing.

**-D hsf.server.ip**

Specifies the IP address to be bound. By default, the IP address of the first network interface controller (NIC) is bound when multiple NICs exist.

**-D HsfBindHost**

Specifies the host to be bound. By default, the HSF server binds the IP address of the first NIC and reports it to the address registry when multiple NICs exist. If you set this parameter to `-DHsfBindHost=0.0.0.0`, the HSF server port is bound to all NICs of the local device.

**-D hsf.publish.interval=400**

Specifies the time interval between the publishing of two services. HSF services are instantly exposed when being published. You can set this parameter to mitigate the burden on starting applications during service exposure. The default value is 400, in ms.

**-D hsf.client.low.water.mark=32 -D hsf.client.high.water.mark=64**

**-D hsf.server.low.water.mark=32 -D hsf.server.high.water.mark=64**

Specifies the write buffer limit for each channel of the consumer or provider.

- The unit is KB. When the consumer exceeds the upper limit, the channel forbids writing new requests and returns an error. Writing is resumed when the write buffer drops below the lower limit.
- When the provider exceeds the upper limit, the channel forbids writing new responses, and the consumer times out because no response is received. Writing is resumed when the write buffer drops below the lower limit.
- The upper and lower limits must be set as a pair, and the upper limit must be greater than the lower limit.

**-D hsf.generic.remove.class=true**

Retrieves the result of a generic call, without output of the `class` field.

**-D defaultHsfClientTimeout**

Specifies the global time-out period of the consumer.

**-D hsf.invocation.timeout.sensitive**

Determines whether the HSF call duration includes the time consumption logic such as connection creation and address selection. The default value of `hsf.invocation.timeout.sensitive` is false.

## 1.5.4. View application changes

After you perform lifecycle operations on applications in the Enterprise Distributed Application Service (EDAS) console, you can go to the Application Details page to check the change status or go to the Change Logs page to check the change logs. Application lifecycle operations include the deployment, startup, scale-out, and scale-in of applications.

## View change details

This topic uses an example of application deployment to describe how to view application changes.

1. After you perform a change operation on an application, return to the application details page. At the top of the application details page, the following message appears: **A change process is ongoing for this application. The application is in Executing state.**
2. Click **View Details** to go to the **Change Details** page. On this page, you can view the change information and the real-time status of the application.
  - Change summary information: includes the change process ID, the execution status, and the change type.
  - Information about change process execution: includes each stage of the entire process and the specific tasks at each stage.
    - If a task is executed on a single instance, icons are used to mark the tasks and the execution results in each stage.
    - If a task is executed on multiple instances, the executed tasks and the execution results are displayed by instance in each stage.
3. In the left part of the change process execution section, click a stage below **Batch x Change**, and then click an instance IP address on the right side. Then, you can view the task execution status of the instance in this stage.
4. Click a task to view the task execution logs.

The system automatically unfolds the logs for failed tasks. For information about how to handle exceptions, see [How do I resolve problems in a change process?](#).

## View application change records

1. In the left-side navigation pane of the application details page, click **Change Records** to view all change records of this application.
2. In the Actions column, click **View** to view the change details and the details of each operation.

## 1.5.5. Log management

The EDAS console provides the runtime log function, allowing you to view the runtime logs of applications without having to log on to the ECS instance. When an exception occurs in an application, you can check logs to troubleshoot the problem.

### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management**. On the **Applications** page, click the name of the target application.
3. On the Application Details page, choose **Log Management > Log Directories** from the left-side navigation pane.

By default, the **Log Directories** page contains two log paths: the log path of the Tomcat container (such as `/home/admin/taobao-tomcat-production-2.0.59.4/logs`) and the log path of EDAS Agent (such as `/home/admin/edas-agent/logs`). Tomcat The path to container logs varies depending on the actual version.

4. Click the log folder or path to show all log files in the folder.

 **Note** Only readable files but not folders are displayed.

5. Double-click a log file to view log details.
  - Select an instance from the **ECS Instance ID/Name/IP** drop-down list to view its real-time logs.
  - Click **Enable Real-time Additions** in the lower-right corner of the page to ensure that the latest additions to the file have been added (similar to the `tail -f` ).
6. (Optional)Bookmark a log path.
  - i. On the **Log Directories** page, select a path or folder and click **Bookmark Log Directory** in the upper-right corner of the page.
  - ii. In the Add Application Log Path dialog box, enter an **application log path** and click **Add**.

 **Notice**

- The path must be in the `/home/admin` directory and contain "log" or "logs".
- The file name must end with a slash (/) to indicate that it is a folder.

To cancel the bookmark status, click the name of a **folder** in the selected directory and click **Remove Directory from Bookmark** in the upper-right corner of the page. When a path is removed from favorites, it is no longer displayed on the logs page. This operation does not delete or change any files on the server.

## 1.5.6. Perform auto scaling on applications in ECS clusters

Auto scaling is an important O&M capability for distributed application management. The auto scaling feature provided by Enterprise Distributed Application Service perceives the status of each instance for an application and implements dynamic scaling accordingly. This ensures the quality of service (QoS) and improves application availability.

### Overview

Auto scaling applies to applications created in **Elastic Compute Service (ECS)** clusters. You need to ensure that the real-time monitoring data of the applications for which auto scaling will be enabled is displayed on the Infrastructure Monitoring and Service Monitoring pages in Application Monitoring.

If the monitoring data is not displayed, perform the following steps to install Log Collector:

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Resources > VPC**.
3. On the **VPC** page, select a Virtual Private Cloud (VPC) where the ECS instance deployed with the application is located and click **Install Log Collector** in the Actions column. To install Log Collector, you need to select an ECS instance in a VPC. We recommend that you install Log Collector on an ECS instance randomly selected out of every 30 to 40 ECS instances. You do not need to

install Log Collector on all the ECS instances in this VPC.

Two to three minutes after Log Collector is installed, you can view data on the Infrastructure Monitoring page. Then, you need to check for service requests on the Service Monitoring page.

If the ECS instances in your cluster are insufficient for scale-out, you can use Alibaba Cloud [Auto Scaling](#) to create an instance. You must use your Enterprise Distributed Application Service (EDAS) account to activate Auto Scaling in advance. **The instance created by using elastic resources is a pay-as-you-go instance.**

The auto scaling feature can determine the capacity based on the CPU, RT, and load of an application instance and implement auto scaling.

- **CPU**: the CPU utilization that is expressed as a percentage.
- **RT**: the time for the system to respond to a request, in ms.
- **Load**: the load on an application instance, which is a positive integer.

Metrics sources:

CPU sources:

ECS cluster: /home/admin/edas-agent/stat/sys\_stat.log[.YYYY-MM-DD]

Swarm cluster: /root/edas-agent/stat/sys\_stat.log[.YYYY-MM-DD]


RT and Load sources:

ECS cluster: /home/admin/logs/eagleeye/stat-eagleeye-hsf.log

ECS cluster: /home/admin/logs/eagleeye/stat-eagleeye-trace.log

Swarm cluster: /root/logs/eagleeye/stat-eagleeye-hsf.log

Swarm cluster: /root/logs/eagleeye/stat-eagleeye-trace.log

 **Note** The data of the preceding three metrics is collected once a minute. The stat-eagleeye-hsf.log file records the High-speed Service Framework (HSF) service calls, while the stat-eagleeye-trace.log file records the HTTP requests.

All these metrics are entered in positive integers without floating-point numbers. If an application runs on multiple ECS instances, the preceding metrics specify the average values of all these instances.

## Auto scale-out

Auto scaling includes auto scale-out and scale-in, for which rules can be configured separately.

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the page, select a namespace. On the **Applications** page, click the name of the desired application.
3. On the Application Details page, click **Auto Scaling** in the left-side navigation pane.
4. Click the switch next to **Scale-Out Rule** to enable scale-out.
5. Set the scale-out rule parameters and click **Save**.
  - **Application Source** (for non-Docker applications):

- **Existing Resources:** During auto scaling, a specified number of idle ECS instances are selected from the cluster where the application is located, and are added for the application. When the instances are added, the application is automatically deployed and started on the instances.
- **Elastic Resources:** During auto scale-out, the auto scaling feature provided by ECS is used. The existing ECS instances are used or new ECS instances are purchased by using the ECS startup template. The existing or newly purchased ECS instances are automatically added to the EDAS cluster (ECS or Swarm cluster) that hosts the application. Also, the application is automatically deployed and started on the added ECS instances.
- **Existing Resources First:** During auto scale-out, idle ECS instances in the cluster are preferentially used. If the number of idle ECS instances in the cluster is insufficient, the auto scaling feature provided by ECS is used to create ECS instances.

If you select **Elastic Resources** or **Existing Resources First**, Auto Scaling may be used to create instances, which are pay-as-you-go instances. In addition, you must set the following parameters for these instances:

- **Building Method:**
  - **Purchase Based on Existing Example Specifications:** An ECS instance where the application has been deployed is selected as the template. By using this template, you can configure specifications such as the CPU, memory, network, disk, and security group to create an ECS instance.
  - **Purchase Based on Instance Startup Template:** You need to first create a startup template in the ECS console and customize data for instance creation. You need to add and set the hostname in userdata. Otherwise, the created ECS instance has an incorrect hostname-to-IP address mapping, which affects the application startup.
- **Login Key:** The key pair is used for Secure Shell (SSH) logon authentication for new ECS instances.
- **Superior Options - Network Type and Multi-Zone Scaling Policy:** **Network Type** indicates the network of the application that you want to scale out. The network cannot be changed. If the current network is a VPC, you need to specify the VSwitch that is connected to the new instance. If you specify multiple VSwitches, EDAS will automatically allocate VSwitches based on the **multi-zone scaling policy**.
- **Trigger Indicators:** set the thresholds of the CPU, response time (RT), and load. When one or more metrics exceed corresponding thresholds, scale-out is triggered.
- **Trigger Conditions:**
  - **Any One of the Indicators:** Auto scale-out is triggered when the threshold of any metric is exceeded.
  - **All Indicators:** Auto scale-out is triggered only when the thresholds of all metrics are exceeded.
- **Last for More Than:** indicates the duration when the metric continuously reaches the threshold, in minutes. Within the duration, if the average value of a metric every minute continuously reaches the set threshold, auto scale-out is triggered. You can configure the duration based on the sensitivity of the cluster service capabilities.
- **Number of Instances for Each Scale-Out:** indicates the number of instances that are automatically added upon each scale-out. You can configure this parameter based on the service capabilities of a single instance of the application.

- **Maximum Number of Instances:** indicates the maximum number of instances in a cluster. When the maximum number is reached, scale-out stops. You can set this parameter based on the resource quota.

## Auto scale-in

The configuration method of **auto scale-in** is similar to the configuration method of **auto scale-out**. For more information about the metrics and setting method, see **Auto scale-out**.

### Note

- When configuring the scale-in and scale-out rules, make sure that the metric values of the scale-in rules are not greater than the metric values of the scale-out rules. Otherwise, an error message appears when you click **Save**.
- The instances created by using elastic resources are preferentially released during scale-in.

## Auto scaling results

After you set auto scaling rules, if auto scale-out or scale-in is enabled, you can click the **Instance Information** tab on the **Basic Information** page to check whether instances are added or deleted. You can also click **Change Logs** in the left-side navigation pane to check the change records where Change Type is **Scale Out** or **Scale In**, Source is **auto\_scale**, and Changed By is **admin**.

## 1.5.7. Throttling and degradation (only applicable to HSF applications in ECS clusters)

Throttling and degradation are mainly used to solve slow system response or breakdown due to excessive burden on backend core services. These features are generally used in high-traffic scenarios, such as flash sales, shopping sprees, major promotions, and empty box scam protection.

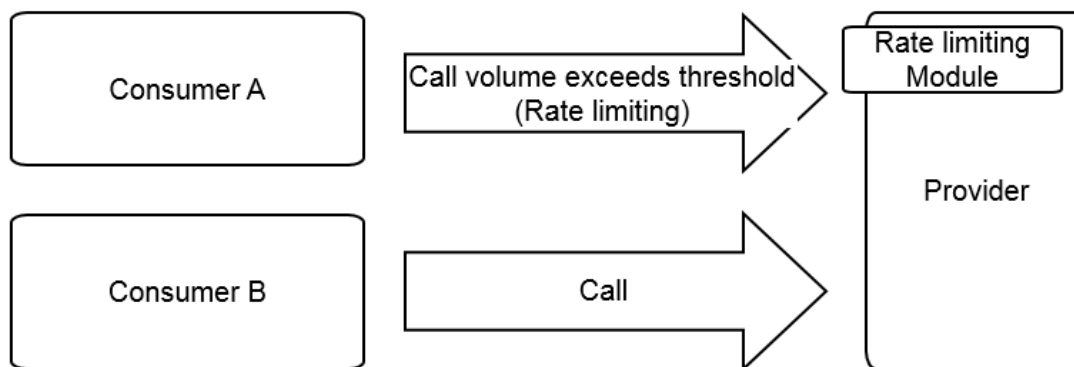
### Throttling

This function controls the traffic threshold or adjusts the traffic ratio. It controls traffic when front-end websites are dealing with heavy access traffic to prevent service unavailability that results from damage to backend core systems. By adjusting the traffic threshold, the throttling function controls the maximum traffic volume of the system to make sure secure and stable system operation.

#### Principles

After the throttling code is configured for a provider and a throttling policy is configured in EDAS, the provider has the throttling function. When a consumer calls the provider, all access requests are calculated by the throttling module. If the call volume of the consumer exceeds the preset threshold in a specific period, the throttling policy is triggered.

Throttling



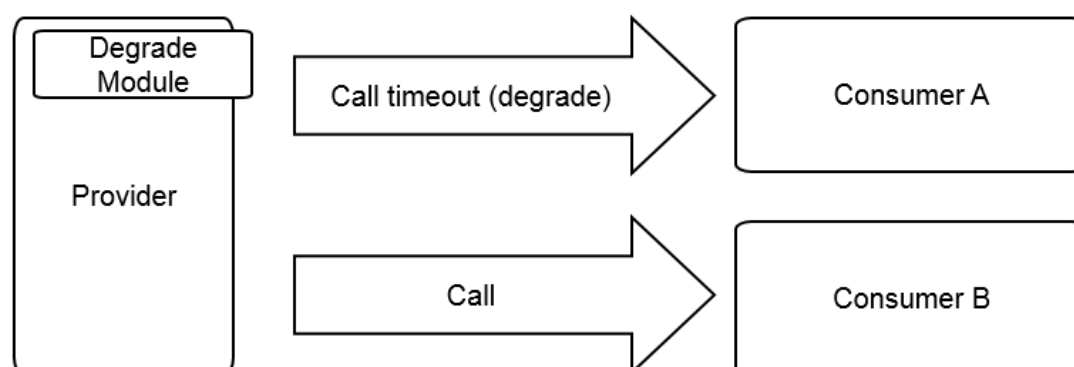
## Degradation

In EDAS, degradation refers to the reduction of the call priority of downstream non-core providers that have timed out to make sure the availability of core consumers.

### Principles

After degradation code is configured for a consumer and a degradation policy is configured in EDAS, the consumer has the degradation function. When the consumer calls a provider, if the response time of the provider exceeds the preset threshold, the degradation policy is triggered, as shown in the following figure.

Degradation




### 1.5.7.1. Throttling management

An application enables multiple services. Enterprise Distributed Application Service (EDAS) allows you to configure throttling rules for the services. This way, traffic that exceeds the service capabilities can be denied. This ensures service stability. EDAS allows you to configure throttling rules based on the queries per second (QPS) and threads. This ensures the optimal operation stability of application systems during traffic peaks.

### Context

- Throttling for High-Speed Service Framework (HSF) services: When the traffic during a traffic peak exceeds the upper threshold defined by the throttling rules, the BlockException error is reported by some consumers. Based on the set threshold, the same number of services as the set threshold are called within 1s.
- Throttling for HTTP requests: During traffic peaks, some consumers are redirected to an error page, the homepage of Taobao.com. Based on the set threshold, some requests can be sent to the services.

 **Notice** Throttling rules apply only to providers. You cannot set throttling rules for consumers. Before you set throttling rules, verify that your application serves as a provider.

## Procedure

1. Write code to set throttling rules.
  - i. [Log on to the EDAS console](#).
  - ii. In the left-side navigation pane, choose **Application Management > Applications**.
  - iii. On the **Applications** page, click an application that is deployed as a provider.
  - iv. On the application details page, choose **Throttling and Degradation > Throttling Rules** in the left-side navigation pane.
  - v. On the **Throttling Rule** page, click **Application Configuration Guide** in the upper-right corner.
  - vi. Write code to set throttling rules based on the provided example.
2. Add the code that you use to set throttling rules to the application code and deploy the application again. For more information, see [Deploy an application \(applicable to ECS clusters\)](#).
3. On the application details page of the EDAS console, choose **Throttling and Degradation > Throttling Rules** in the left-side navigation pane.
4. On the **Throttling Rule** page, click **Add Throttling Rule** in the upper-right corner.
5. In the **Add Throttling Rule** dialog box, set the throttling rule parameters and click **OK**. Throttling rule parameters

Parameter	Description
<b>Throttling Type</b>	Select HSF Throttling or HTTP or HTTPS Throttling. Select the throttling type based on the access type of the application.
<b>API to be Throttled</b>	Select the API to which the throttling rule applies from the listed APIs as needed.
<b>Methods to be Throttled</b>	The system automatically loads all the methods based on the selected API. Select a method or all the methods to which the throttling rule applies.
<b>Throttled Application</b>	The application list includes all applications except the current application because all the applications may access the current application. Select the application to which the throttling rule applies from the application list as needed.
<b>Throttling Granularity</b>	Select QPS Throttling or Thread Traffic Throttling. <ul style="list-style-type: none"> <li>◦ QPS Throttling limits the number of QPS.</li> <li>◦ Thread Traffic Throttling limits the number of threads.</li> </ul> In normal cases, a large number of threads leads to a large QPS. However, the QPS of a thread is usually greater than 1 because a thread continuously sends requests and the request response time is usually dozens of milliseconds.
<b>Throttling Threshold</b>	Throttling is triggered when the set threshold is exceeded.

## What's next


On the **Throttling Rule** page, find the rule that you want to manage, and click **Edit**, **Deactivate**, **Enable**, or **Delete** in the Actions column.

## 1.5.7.2. Degradation management

An application calls multiple external services. Service degradation can be configured to pinpoint and block poor services. This feature ensures the stable operation of your application and prevents the performance of your application from being compromised by dependency on poor services.

### Context

Enterprise Distributed Application Service (EDAS) allows you to configure degradation rules based on the response time. This helps reduce the dependency of your application on poor services during traffic peaks. A consumer that triggers a degradation rule does not initiate new remote calls within the specified time window. Instead, it throws the `DegradeException` error. After the time window ends, the original remote service calls are recovered.

 **Note** The degradation rules apply only to consumers and cannot be configured for providers. Before you configure a degradation rule, ensure that the application serves as a consumer.

### Procedure

1. Write the degradation rule code.
  - i. Log on the EDAS console. For more information, see [Log on to the EDAS console](#).
  - ii. In the left-side navigation pane, choose **Application Management > Applications**.
  - iii. On the **Applications** page, click an application that is deployed as a service provider.
  - iv. In the left-side navigation pane of the application details page, choose **Throttling and Degradation > Degradation Rules**.
  - v. On the **Degradation Rules** page, click **Application Configuration Guide** in the upper-right corner.
  - vi. Write the degradation rule code. Sample code is provided on the page.
2. Add the degradation rule code to the application and redeploy the application. For more information, see [Deploy an application \(applicable to ECS clusters\)](#).
3. In the left-side navigation pane of the application details page in the EDAS console, choose **Throttling and Degradation > Degradation Rules**.
4. In the upper-right corner of the **Degradation Rules** page, click **Add Downgrade Rule**.
5. On the **Add Downgrade Rule** page, set parameters for the degradation rule and click **OK**.  
Degradation rule parameters

Parameter	Description
Degradation Type	Select HSF Downgrade or HTTP Downgrade based on your actual business needs.
API to be Downgraded	All APIs that the consumer is consuming are listed. Select the API to be degraded as needed.

Parameter	Description
Method to be Downgraded	All methods related to the selected APIs are automatically loaded. Select whether to degrade all methods or a specific method as needed.
RT Threshold	The maximum acceptable service response time (RT) that is measured in milliseconds. If this threshold is exceeded, the selected APIs or methods are degraded.
Time Window	The rule execution duration after degradation is triggered. Unit: seconds.

## What's next

On the **Degradation Rules** page, find the rule that you want to manage, and click **Edit**, **Deactivate**, **Enable**, or **Delete** on the right side.

## 1.5.8. Container version management (only applicable to HSF applications in ECS clusters)

EDAS allows you to view container versions and historical publishing details and perform upgrade and downgrade.

### Context

An EDAS container consists of Ali-Tomcat, Pandora, and custom Pandora plug-ins. In addition to the support for existing **Apache Tomcat** core functions, EDAS provides a class isolation mechanism, QoS, and Tomcat-Monitor. Highly custom plug-ins are added to EDAS containers to implement complex and advanced functions, such as container monitoring, service monitoring, and tracing. Applications deployed by using EDAS must run in EDAS containers.

You must select a container version when creating an application in EDAS. EDAS containers are maintained and published by the EDAS development team. Choose **Application Management > Container Version** to view the container publishing history and the description of each publishing operation. Generally, a container of a later version is superior to a container of an earlier version in terms of stability and function variety.

EDAS container publishing does not affect deployed applications. Once a new container is available, you can immediately upgrade your container to the latest version.

### Procedure

1. In the left-side navigation pane, choose **Application Management** to go to the Applications page.
2. Click the name of the target application to go to the Application Details page.
3. In the left-side navigation pane, choose **Container Version** to go to the Container Version page.
4. Locate the row that contains the target container version and click **Upgrade to This Version** or **Downgrade to This Version** on the right to upgrade or downgrade the container in one click.

## 1.6. Application management for Container Service for Kubernetes clusters

## 1.6.1. Application deployment

### 1.6.1.1. Configure a Kubernetes base image

For the security of image repositories, you cannot use the features of a Kubernetes application in EDAS immediately after you deploy it. To use the features of the application, configure an image repository first. This topic describes how to configure a Kubernetes base image in EDAS. The image repository of Container Registry is used in this example.

#### Prerequisites

An organization and a resource set are created. For more information, see the *Manage organizations* and *Manage resource sets* sections of Apsara Uni-manage User Guide.

#### Step 1: Create an image repository in Container Registry

Create an image repository in the Container Registry console. Push the EDAS images that are open to the Internet to the image repository that is exclusive to Apsara Stack.

1. Log on to the **Container Registry console**. If you use Container Registry for the first time, you must set a password that is used to log on to the Container Registry console.  
(Optional)
  - i. In the Container Registry console, click **Reset Docker Login Password**.
  - ii. In the **Reset Docker Login Password** dialog box, set the **Password** and **Confirm Password** parameters and click **OK**.
2. In the left-side navigation pane, click **Namespace**. On the **Namespace** page, click **Create Namespace** in the upper-right corner.
3. In the **Create a namespace** dialog box, enter a namespace name and click **Confirm**. The area information that consists of the organization, the resource set, and the region is automatically specified in the system. You do not need to enter the information.

Create Namespace

\* Namespace Name

\* Namespace ID  It can only contain letters or digits.

Region

Description  0/64

Create Cancel

4. In the left-side navigation pane, click **Repositories**.
5. On the **Repositories** page, click **Create Repository**.
6. In the **Create a repository** dialog box, create a public image repository.  
In EDAS, you must create the following image repositories: edas-agent, edas-controller, edas-logtail, hydra, and rollout. This topic describes how to create the edas-agent repository.

- i. On the **Repository Info** tab, set the parameters for the image repository and click **Create Repository**.
  - **Area**: consists of the **Tissue**, **Resource set**, and **Geographical** parameters. The system automatically set these parameters. You do not need to enter the values of them.
  - **Namespace**: the namespace of the image repository. Select the namespace that you created.
  - **Warehouse name**: the name of the image repository.
  - **Summary**: the summary of the image repository.
  - **Specific information**: the description of the image repository.
  - **Warehouse type**: Set this parameter to **Open warehouse**.
- ii. Repeat the preceding operations to create the other image repositories.

## Step 2: Pull images to a created image repository

1. On the **Repositories** page, find the created repository to which you want to synchronize images, and click **Admin** in the **Actions** column.
2. On the **Details** page, view the domain name and the logon username of the image repository.
3. To pull EDAS images from the Internet, run the following commands on an on-premises server where Docker is installed and started or on an ECS instance in Apsara Stack. Make sure that the ECS instance can access the image repositories on the Internet and in Apsara Stack.

```
docker pull registry.cn-beijing.aliyuncs.com/edas/edas-agent:3.0.0
docker pull registry.cn-beijing.aliyuncs.com/edas/edas-logtail:1.0.0
docker pull registry.cn-beijing.aliyuncs.com/edas/edas-controller:1.0.0
docker pull registry.cn-beijing.aliyuncs.com/edas/edas-controller:2.0.5
docker pull registry.cn-beijing.aliyuncs.com/edas-controller/hydra:1.0.0
docker pull registry.cn-beijing.aliyuncs.com/edas-controller/rollout:1.0.0
```

4. Push the pulled EDAS images to the created image repository.
  - i. Run the **docker login --username={username} {registry.domain}** command to log on to the image repository.  
{username} and {registry.domain} are the logon username and the domain name that you can view on the **Details** page of the image repository.

- ii. Run the **docker tag** command in the image repository to mark the public EDAS images as the images in the created image repository.

```
docker tag registry.cn-beijing.aliyuncs.com/edas/edas-agent:3.0.0 {registry.domain}/{namespace}/edas-agent:3.0.0
docker tag registry.cn-beijing.aliyuncs.com/edas/edas-logtail:1.0.0 {registry.domain}/{namespace}/edas-logtail:1.0.0
docker tag registry.cn-beijing.aliyuncs.com/edas/edas-controller:1.0.0 {registry.domain}/{namespace}/edas-controller:1.0.0
docker tag registry.cn-beijing.aliyuncs.com/edas/edas-controller:2.0.5 {registry.domain}/{namespace}/edas-controller:2.0.5
docker tag registry.cn-beijing.aliyuncs.com/edas-controller/hydra:1.0.0 {registry.domain}/{namespace}/hydra:1.0.0
docker tag registry.cn-beijing.aliyuncs.com/edas-controller/rollout:1.0.0 {registry.domain}/{namespace}/rollout:1.0.0
```

{registry.domain} and {namespace} are the domain name of the image repository and the namespace to which the image repository belongs.

- iii. Run the **docker push** command in the image repository to push the EDAS images to the created image repository.

```
docker push {registry.domain}/{namespace}/edas-agent:3.0.0
docker push {registry.domain}/{namespace}/edas-logtail:1.0.0
docker push {registry.domain}/{namespace}/edas-controller:1.0.0
docker push {registry.domain}/{namespace}/edas-controller:2.0.5
docker push {registry.domain}/{namespace}/hydra:1.0.0
docker push {registry.domain}/{namespace}/rollout:1.0.0
```

{registry.domain} and {namespace} are the domain name of the image repository and the namespace to which the image repository belongs.

## Step 3: Configure the address of the image repository in EDAS

1. To log on to the CaiFs terminal, perform the following steps:
  - i. Log on to Apsara Infrastructure Management Framework. For more information about how to log on to Apsara Infrastructure Management Framework, see the **Log on to Apsara Infrastructure Management Framework** section of the *Routine maintenance* topic in *Operations and Maintenance Guide*.
  - ii. In the left-side navigation pane, choose **Operations > Service Operations**. On the **Services** page, enter *edasService* in the **Service** search box. Select **edas-edasService** from the search results. Then, click **Operations** in the **Actions** column.
  - iii. On the *edas-edasService* Service Details page, click the **Clusters** tab. In the **Cluster** list, click an instance name.
  - iv. In the **Service Role** section of the **Cluster Details** page, select *edas-edasService.CaiFs*. Then, click **Terminal** in the **Actions** column.
2. On the **TerminalService** page, view and access the */home/admin/bin* directory of the CaiFs container.
  - i. In the left-side navigation pane of the **TerminalService** page, click the name of a machine that belongs to the instance.
  - ii. In the */home/admin* directory, run the **docker ps** command to list the containers on the machine.

- iii. Find and record the container ID of the CaiFs container.
- iv. In the CaiFs container, run the **docker exec -ti <CaiFs container ID> bash** command.
- v. Run the **cd /home/admin/bin** command to access the */home/admin/bin* directory.
3. In the */home/admin/bin* directory, run the **bash update\_image.sh {registry.domain} {namespace}** command to configure the image repository address.  
 {registry.domain} and {namespace} are the domain name of the image repository and the namespace to which the image repository belongs.  
 After you run the command, the command output appears. If the message **update succeed** appears, the configuration is successful.

## Step 4: Restart the EdasEam service role in Apsara Infrastructure Management Framework

1. Log on to Apsara Infrastructure Management Framework. For more information, see *Apsara Infrastructure Management Framework Operations Guide*.
2. In the left-side navigation pane, choose **Operations > Service Operations**. On the **Services** page, enter *edasService* in the **Service** search box. Select **edas-edasService** from the search results. Then, click **Operations** in the **Actions** column.
3. On the *edas-edasService* Service Details page, click the **Clusters** tab. In the **Cluster** list, click an instance name.
4. In the **Service Role** section, select **edas-edasService.EdasEam**. In the **Machines** list, find the machine and click **Restart** in the **Actions** column.

### 1.6.1.2. Create an application image

You can run commands in local development tools to create a WAR or a JAR package for an application. Create an image based on the WAR or JAR package. Then, upload the image to the Alibaba Cloud image repository to deploy the application. This topic describes how to create Dockfiles for the images of applications that use different frameworks. This topic also describes how to upload the images to the Alibaba Cloud image repository.

#### Prerequisites

Before you create an application image, make sure that you have read [Appendix: Environment variables during runtime](#). Create an image for an EDAS application based on the following instructions.

#### Usage specifications and limits

When you use a Dockerfile to create a custom image, make sure that the following specifications are met and the following limits are not violated.

- **Tenant and encryption information**  
 The tenant and encryption information is used to authenticate EDAS application users and encrypt credentials.  
**Resource**

Resource type	Resource name	Description
Secret	edas-certs	The encryption dictionary that stores the EDAS tenant information.

Environment variables

Environment variable key	Type	Description
tenantId	String	The ID of an EDAS tenant.
accessKey	String	The AccessKey ID for authentication.
secretKey	String	The AccessKey secret for authentication.

#### Local files

Path	Type	Description
/home/admin/.spas_key/default	File	The authentication information of an EDAS tenant. This includes the preceding environment variable information.

- Service information

The service information contains the information such as the EDAS domain name and the port to be connected during runtime.

#### Resource

Resource type	Resource name	Description
ConfigMap	edas-envs	The EDAS service information.

#### Environment variables

Environment variable key	Type	Description
EDAS_ADDRESS_SERVER_DOMAIN	String	The service domain name or IP address of the configuration center.
EDAS_ADDRESS_SERVER_PORT	String	The service port of the configuration center.
EDAS_CONFIGSERVER_CLIENT_PORT	String	The service port of ConfigServer.

- Environment variables during application runtime

The following environment variables are provided when you deploy an application in EDAS. This ensures that the application runs properly. Do not overwrite the current configuration.

#### Environment variables

Environment variable key	Type	Description
POD_IP	String	The IP address of a pod.
EDAS_APP_ID	String	The ID of the EDAS application.
EDAS_ECC_ID	String	EDAS ECC ID

Environment variable key	Type	Description
EDAS_PROJECT_NAME	String	The same as EDAS_APP_ID. This parameter is used for trace parsing.
EDAS_JM_CONTAINER_ID	String	The same as EDAS_ECC_ID. This parameter is used for trace parsing.
EDAS_CATALINA_OPTS	String	The CATALINA_OPTS parameter that is required during the middleware runtime.
CATALINA_OPTS	String	The same as EDAS_CATALINA_OPTS. This parameter is used as a default startup parameter of Tomcat.
CATALINA_HOME	String	The Tomcat path.
PANDORA_LOCATION	String	The Pandora path, which can be viewed in the HSF applications.

## Create a standard Dockerfile

A **Dockerfile** is a configuration file in a text format. You can use the Dockerfile to quickly create an image.

You can use Dockerfiles to create images for HSF, Spring Cloud, or Dubbo applications based on your application framework. The following examples describe how to create Dockerfiles for the applications that use different frameworks.

An EDAS standard **Dockerfile** describes all the instructions for creating application runtime environments in EDAS. The instructions help you learn how to download, install, and start OpenJDK, Tomcat, WAR and JAR packages, and other components. You can modify the Dockerfile for purposes such as replacing the OpenJDK version, modifying the Tomcat configuration, and changing the runtime environment. For more information, see [Custom image: Use the latest version of OpenJDK](#).

- [Example of the Dockerfile for an HSF application](#)
- [Example of the Dockerfile for a Spring Cloud or Dubbo application](#)

## Example of the Dockerfile for an HSF application

```
# Using the centos7 + openjdk8 + ali tomcat7 + pandora 3.5.9 for the base
FROM apaas/edas-centos-openjdk8-alitomcat7:latest
MAINTAINER Alibaba Cloud EDAS Team<edas-dev@list.alibaba-inc.com>
ENV TZ="Asia/Shanghai"
# Default put your application package into /home/admin/app/
ENV APP_HOME /home/admin/app/
#####----> First case: deploy a fat-jar file
ARG APP_LOCATION=https://edas-hz.oss-cn-hangzhou.aliyuncs.com/prod/demo/HSF_PROVIDER.jar
#####----> Second case: deploy a war file, simply change the file name your wanted.
#ARG APP_LOCATION=https://edas-hz.oss-cn-hangzhou.aliyuncs.com/prod/demo/HSF_PROVIDER.war
#####----> Third case: deploy a local storage file.
#ARG APP_LOCATION=/Users/yanliang.lyl/workspace/java/demo/HSF_PROVIDER.war
# Then download to package into /home/admin/app/
ADD ${APP_LOCATION} ${APP_HOME}/
# Default working dir is set to /home/admin
# WORKDIR /home/admin
# Entry point set to /home/admin/bin/start.sh,
# which including inside basic image (apaas/edas-centos-openjdk8-alitomcat8)
# CMD ["/home/admin/bin/start.sh"]
```

## Example of the Dockerfile for a Spring Cloud or Dubbo application

```
# Using centos7 + openjdk8 + tomcat8 for the base
FROM apaas/edas:latest
# Default put your application package into /home/admin/app/
ENV APP_HOME /home/admin/app/
##### SIMPLE REPLACE YOUR PACKAGE FILE
#####----> First case: deploy a fat-jar file
ARG APP_LOCATION=https://edas-hz.oss-cn-hangzhou.aliyuncs.com/prod/demo/DUBBO_PROVIDER.jar
#####----> Second case: deploy a war file, simply change the file name your wanted.
#ARG APP_LOCATION=https://edas-hz.oss-cn-hangzhou.aliyuncs.com/prod/demo/DUBBO_PROVIDER.war
#####----> Third case: deploy a local storage file.
#ARG APP_LOCATION=/Users/yanliang.lyl/workspace/java/demo/SPRINT_CLOUD_PROVIDER.war
# Then download to package into /home/admin/app/
ADD ${APP_LOCATION} ${APP_HOME}/
```

## Description of EDAS base images

You can choose one of the following base images based on the runtime environment of your application:

Image name	OS version	Java version	Tomcat version	Pandora version	Remarks
apaas/edas	CentOS 7	OpenJDK 1.8	8.5.42	N/A	Same as apaas/edas-centos-openjdk8-tomcat8. Latest version number: latest.

Image name	OS version	Java version	Tomcat version	Pandora version	Remarks
apaas/edas-centos-openjdk8-tomcat8	CentOS 7	OpenJDK 1.8	8.5.42	N/A	We recommend that you use this version for common Spring Cloud or Dubbo applications. Latest version number: latest.
apaas/edas-centos-openjdk8-tomcat7	CentOS 7	OpenJDK 1.8	7.0.93	N/A	We recommend that you use this version for common Spring Cloud or Dubbo applications based on Tomcat 7.x. Latest version number: latest.
apaas/edas-centos-openjdk8-alitomcat8	CentOS 7	OpenJDK 1.8	8.5.37	3.5.9	We recommend that you use this version for HSF applications based on Tomcat 8.x. Latest version number: latest.
apaas/edas-centos-openjdk8-alitomcat7	CentOS 7	OpenJDK 1.8	7.0.92	3.5.9	We recommend that you use this version for HSF applications based on Tomcat 7.x. Latest version number: latest.

## Custom image: Use the latest version of OpenJDK

You can customize settings in the created standard Dockerfile based on your requirements.

Upgrade OpenJDK: You can download and install the latest version of OpenJDK in the Dockerfile. The following example shows how to download and install OpenJDK 9:

```
# Uninstall the original OpenJDK, and then download and install OpenJDK 9.  
RUN yum erase -y java; yum -y install java-1.9.0-openjdk-devel
```

## Custom image: Upgrade the EDAS Container version for HSF applications

For the HSF applications that are deployed in Container Service for Kubernetes clusters, you can upgrade the container version to use the new features of the middleware or to fix the existing issues that may occur in the original version. To upgrade the container version, perform the following steps:

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, click **Create Application** in the upper-right corner. In the **Application Runtime Environment** section of the **Application Information** page, view and obtain the **Pandora version** and the **Ali-Tomcat version**.
4. Replace the version number in the Dockerfile, such as 3.5.4.

```
# Set the version of EDAS Container or Pandora.  
ENV UPGRADED_CONTAINER_VERSION 3.5.4  
RUN /home/admin/bin/upgradecontainer.sh
```

5. Recreate and publish an application image.

## Custom Image: Modify JVM startup parameters in the image

The JAVA\_OPTS environment variable is used to pass the Java Virtual Machine (JVM) parameters that are based on the base image. The following example shows how to modify the JVM startup parameters.

```
FROM apaas/edas:latest  
# Set the JVM parameter ENV JAVA_OPTS="\.  
-Xmx3550m \  
-Xms3550m \  
-Xmn2g \  
-Xss128k"
```

## Custom image: Modify the configurations of an application of the Spring Boot type

For the FatJar applications based on the EDAS base images, you can modify the Tomcat startup configurations, such as context path, startup port, and parameters. These parameters are passed based on the APP\_ARGS environment variable. The following example shows how to modify the startup configurations.

```
FROM apaas/edas:latest
# The default configurations in the base image:
# - The root context path (/) is used.
# - The encoded URI in ISO-8859-1 is used.
# - The application starts on port 8080.
# - The maximum number of the thread pools used by Tomcat is 400.
# You can also rewrite the environment variable APP_ARGS to modify the default configurations. For example:
ENV APP_ARGS="--server.context-path=/ \
--server.tomcat.uri-encoding=ISO-8859-1 \
--server.port=8080 \
--server.tomcat.max-threads=400"
```

## Build an on-premises image

Access the directory where the Dockerfile is located from the local command line. Run the docker build command to build an image.

```
docker build -t [Label name, preferably application name]:[Version].
or docker build -t [Label name, preferably application name]:[Version] -f /path/to/custom_dockerfile_name.
# If the created Dockerfile is located in another directory or is not named Dockerfile, use the latter command
.
```

Example:

```
docker build -t hsf-provider:1.0.0 .
```

Then, run the `docker images | grep <Image label name>` command to view the packaged on-premises image.

## Upload images to the image repository

The application images that are created and built locally can be uploaded to the container image repository provided by Apsara Stack. Log on to the Container Registry console. On the Container Registry console, switch to the region of the application that has already been created or is to be created in EDAS. View the image repository list and select an existing image repository or create an image repository to store your packaged application image. We recommend that you use the application name as the repository name for easy identification.

To upload an on-premises image to the image repository, run the commands that are used to push images to Container Registry. These commands are provided on the Basic Information tab of the specified image repository.

```
docker login --username=[Current login username] [region_id].aliyuncs.com # Enter the fixed or temporary password that is used as the access credential for the default instance on Container Registry. The password is not the one for your Apsara Stack tenant account.
docker tag [ID of the on-premises application image] registry.[region_id].aliyuncs.com/[Namespace name]/[Repository name]:[Image version]
docker push registry.[region_id].aliyuncs.com/[Namespace name]/[Image Repository name]:[Image version]
```

Example:

```
docker login --username=tdy218@gmail.com registry.cn-hangzhou.aliyuncs.com
docker tag 2b64f63601a7 registry.cn-hangzhou.aliyuncs.com/webapps/hsf-provider:1.0.0
docker push registry.cn-hangzhou.aliyuncs.com/webapps/hsf-provider:1.0.0
```

## Appendix: Environment variables during runtime

If you create a custom image by using a Dockerfile, some environment variables are automatically filled by EDAS during runtime. The following table lists these environment variables.

Environment variable key	Description
POD_IP	POD IP
EDAS_APP_ID	The ID of the EDAS application.
EDAS_ECC_ID	EDAS ECC ID
EDAS_PROJECT_NAME	The same as EDAS_APP_ID. This parameter is used for trace parsing.
EDAS_JM_CONTAINER_ID	The same as EDAS_ECC_ID. This parameter is used for trace parsing.
EDAS_CATALINA_OPTS	The CATALINA_OPTS parameter that is required during the middleware runtime.
CATALINA_OPTS	The same as EDAS_CATALINA_OPTS, which is a default startup parameter of Tomcat.
CATALINA_HOME	The Tomcat path.
PANDORA_LOCATION	The Pandora path, which can be viewed in HSF applications.

Do not use the file in the following directory: /home/admin/.spas\_key/default. This file is overwritten when the pod runs.

### 1.6.1.3. Deploy an application to a Container Service for Kubernetes cluster by using WAR or JAR packages

After you import a Container Service for Kubernetes cluster into EDAS, you can deploy applications in the Container Service for Kubernetes cluster.

#### Prerequisites

A self-managed Kubernetes cluster is imported. For more information, see [Import Container Service Kubernetes clusters](#).

#### Context

You can use an image, JAR package, or WAR package to deploy an application. This topic describes how to deploy an application by using a JAR package. The deployment process for a WAR package is similar to that for a JAR package. If you need to deploy an application by using an image, see [Deploy microservice applications in a Kubernetes environment](#).

#### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.

3. In the top navigation bar of the **Applications** page, select a **Region**. At the top of the page, select a **namespace**. In the upper-left corner of the page, click **Create Application**.
4. On the **Application Information** tab, specify Cluster Type and Application Runtime Environment and click **Next**.

Parameter	Description
Cluster Type	Select <b>Kubernetes Cluster</b> .
Application Runtime Environment	<p>In this topic, select <b>Java</b> as an example.</p> <ul style="list-style-type: none"> <li>◦ <b>Custom</b>: Select this option if you need to deploy an application in a Kubernetes cluster by using a custom image.</li> <li>◦ <b>Java</b>: Select this option if you need to deploy a Dubbo or Spring Cloud application in a Kubernetes cluster by using a universal JAR package. You can change the Java environment after you select this mode.</li> <li>◦ <b>Tomcat</b>: Select this option if you need to deploy a Dubbo or Spring Cloud application in a Kubernetes cluster by using a universal WAR package. You can change the Java environment and the container version after you select this mode.</li> <li>◦ <b>EDAS-Container (HSF)</b>: Select this option if you need to deploy an HSF application by using a WAR or FatJar package. You can change the Java environment, Pandora version, and Ali-Tomcat version after you select this mode.</li> </ul>

5. On the **Application Configuration** tab, configure the environment information, basic information, deployment method, and resource parameters for the application, and click **Next**.

Parameter	Description
Namespace	<p>Namespaces are used to isolate resources from services. Select a namespace from the drop-down list.</p> <ul style="list-style-type: none"> <li>◦ If no namespace is created, click <b>Create Namespace</b>. For more information, see <a href="#">Create a namespace</a>.</li> <li>◦ If you do not have the isolation requirement, select <i>Default</i> from the drop-down list. This value indicates that the default namespace is selected.</li> </ul>
Cluster	Select the imported Kubernetes cluster from the drop-down list on the right.

Parameter	Description
K8s Namespace	<p>Internal system objects are allocated to different namespaces to form logically isolated projects, groups, or user groups. This way, different groups can share the resources of the entire cluster when they are separately managed.</p> <ul style="list-style-type: none"><li>◦ <b>default</b>: The default namespace. When no namespace is specified for an object, default is used for the object.</li><li>◦ <b>kube-system</b>: The namespace of objects that are created by the system.</li><li>◦ <b>kube-public</b>: The namespace that is automatically created by the system. This namespace can be read by all the users, including users who are not authenticated.</li></ul>
Application Name	<p>Enter the application name. The name must start with a letter and can contain digits, letters, and hyphens (-). You can enter up to 36 characters.</p>
Application Description	<p>Enter the description of the application. The description can be up to 128 characters in length.</p>
Image Repository	<p>Select the image repository that you created in the list.</p>
Source of Deployment Package	<ul style="list-style-type: none"><li>◦ <b>Customer Program</b> Specify <b>File Uploading Method</b>.<ul style="list-style-type: none"><li>▪ <b>Upload JAR Package</b>: Select and upload a local JAR package.</li><li>▪ <b>JAR Package Address</b>: Enter the address of your deployment package.</li></ul></li><li>◦ <b>Official Demo</b> EDAS provides the following demo types: <b>Spring Cloud Server Application</b>, <b>Spring Cloud Client Application</b>, <b>Dubbo Server Application</b>, and <b>Dubbo Client Application</b>. Select a demo type based on your requirements.</li></ul>
Version	<p>Enter a version number. You can specify a version number or use a timestamp as the version number.</p>
Time Zone	<p>Specify the time zone of the application.</p>

Parameter	Description
<b>Total Pods</b>	Specify the number of pods to be deployed for this application. Pods are the smallest units for application deployment. An application can contain multiple pods. When an SLB instance is used, a request is randomly allocated to a pod for processing. When a pod fails to run or encounters a fault, this pod can automatically restart or services on this pod can be rapidly migrated to other pods. This ensures high availability of applications. If stateful applications that use persistent storage are redeployed, instance data is retained. If stateless applications are redeployed, instance data is not retained. You can specify up to 50 pods.
<b>Single Pod Resource Quota</b>	Enter the CPU cores and the memory for an individual pod. To specify the quota, enter a number. The default value 0 indicates no quota.

6. (Optional)Configure the advanced application settings and click **Create Application**. Advanced settings consist of **Startup Command**, **Environment Variables**, and **Application Life Cycle Management**.

i. Specify a startup command.

Parameter	Description
<b>Startup Command</b>	Enter a startup command, such as <code>nginx</code> .
<b>Startup Parameters</b>	Each parameter is in a row. For example, enter <code>-g</code> in the field, click <b>Add</b> , and enter the <code>daemon off</code> parameter in a new row.

#### Notice

- If you are not familiar with the **CMD** or the **ENTRYPOINT** content of the original Dockerfile image, do not customize startup commands and startup parameters. Otherwise, you cannot create applications due to invalid custom commands.
- The Docker runtime supports only one **ENTRYPOINT** command. Therefore, the startup command specified in the EDAS console overwrites the **ENTRYPOINT** and **CMD** commands that are specified in the process of creating the Docker container image.

ii. Specify environment variables. When you create the application, inject the environment variables that you have entered to the container that is to be generated. This saves you from repeatedly adding common environment variables.

- If you need to specify parameters such as the JVM heap memory, JVM property parameters, and javaagent, you can add the relevant parameters when you specify environment variables:
  - Variable Name: CATALINA\_OPTS.
  - Variable Value: [Parameters to be added] \$(EDAS\_CATALINA\_OPTS).

Type	Variable Name	Variable Value/Variable Reference
Custom	CATALINA_OPTS	-Xmx1024m -Dhttp.publish.default.id=true \$EDAS_CATALINA_OPTS

+ Add

[Configure environment variables](#)

- When you use a MySQL image, you can add the following environment variables:
  - MYSQL\_ROOT\_PASSWORD (required): allows you to specify a MySQL root password.
  - MYSQL\_USER and MYSQL\_PASSWORD (optional): allows you to add an account in addition to the root account and specify a password.
  - MYSQL\_DATABASE (optional): allows you to specify the database that you want to create when the container is generated.
- If you use another type of image, specify the environment variables based on your requirements.

iii. Specify the command script for application lifecycle management.

In general, some operations are performed before and after an application is started or stopped. For example, you can deploy resources before you start an application and gracefully disconnect the application. You can notify other services or applications before you stop the application. EDAS integrates with the lifecycle hook feature based on Kubernetes. This allows you to configure PostStart and PreStop hooks for container lifecycles.

If you deploy an application in a Kubernetes cluster, you must check whether the pod is alive and capable of providing services. EDAS integrates with the pod probe configuration feature based on Kubernetes. This feature allows you to configure the Liveness probe to determine the time when containers are restarted. It also allows you to configure the Readiness probe to determine whether containers are capable of receiving traffic.

Parameter	Description
PostStart Configuration	A container hook. It is immediately triggered after a container is created to notify the container of its creation. The hook does not pass parameters to the relevant hook handler. If the relevant hook handler fails to run, the container is stopped and the restart policy of the container is used to determine whether to restart the container. For more information, see <a href="#">Container Lifecycle Hooks</a> .
PreStop Configuration	A container hook. It is triggered before a container is deleted. The corresponding hook handler must be run before the request to delete the container is sent to the Docker daemon. The Docker daemon sends an SGTERN semaphore to itself to delete the container, regardless of the running result of the corresponding hook handler. For more information, see <a href="#">Container Lifecycle Hooks</a> .
Liveness Configuration	A container probe. It monitors the health status of applications. If an application is unhealthy, its relevant container is deleted and recreated. For more information, see <a href="#">Pod Lifecycle</a> .
Configure Readiness	A container probe. It monitors whether applications have been started and are running properly. If an application is not running, the container status is updated. For more information, see <a href="#">Pod Lifecycle</a> .

7. In the **Creation Completed** page, confirm **Basic Information**, **Configurations**, and **Advanced Settings**, and then click **Create Application**.

After the system starts to deploy the application, the message **A change process is ongoing for this application. The application is in Executing state.** is prompted on the top of the **Basic Information** page. It requires about 2 minutes to deploy the application.

You can also click **View Details** next to the prompted message to navigate to the **Change Details** page of the application. On this page, you can check the deployment progress and log data.

## 1.6.1.4. Canary release for Kubernetes clusters

For Spring Cloud or Dubbo microservice applications deployed in a Kubernetes cluster, you can use canary release to perform a small-scale verification. After the verification passes, you can perform a full upgrade. This makes the upgrade secure.

### Limits

- HSF applications do not support canary release.
- Dubbo applications have no limits when canary release is used.
- Spring Cloud applications have the following limits when canary release is used:
  - Spring Cloud applications that are built by using Netflix Zuul and Spring Cloud Gateway are not supported.
  - If an application depends on the features and the configurations of `Deployment.Metadata.Name` or `Deployment.Metadata.Uid`, do not use canary release. Otherwise, exceptions occur for the native features after the release.



### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**. In the top navigation bar, select a region.
3. On the **Applications** page, select **Namespaces**. Select **Container Service or Serverless Kubernetes Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to manage.
4. In the upper-right corner of the **Basic Information** tab, click **Deploy Application**.
5. On the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner of the **Canary Release (Phased)** section.
6. On the **Canary Release (Phased)** page, specify the deployment parameters, release policy, and canary release rules for the application and click **OK**.

i. Specify the deployment parameters. Description of deployment parameters

Parameter	Description
<b>Configure Image</b> (applicable to only applications deployed by using images)	The images cannot be changed. You can update only the image versions.
<b>Application Runtime Environment</b> (applicable to applications deployed by using JAR and WAR packages)	<p>The value of this parameter must be consistent with the runtime environment that is used for the previous deployment.</p> <ul style="list-style-type: none"> <li>■ JAR package: The application runtime environment is <b>Standard Java Application Runtime Environment</b> and cannot be changed.</li> <li>■ WAR package: The application runtime environment is <b>Apache Tomcat</b> and cannot be changed. You can change the version based on your requirements.</li> </ul>
<b>Java Environment</b> (applicable to applications deployed by using JAR and WAR packages)	Select the value from the drop-down list based on your requirements.
<b>File Uploading Method</b> (applicable to applications deployed by using JAR and WAR packages)	<p>The deployment package type (WAR or JAR package) must be the same as that of the previous deployment and cannot be changed. Specify the parameter based on your requirements. You can select <b>Upload JAR Package</b> or <b>Upload WAR package</b> to upload the relevant deployment package. You can also select <b>JAR Package Address</b> or <b>WAR Package Address</b> to enter the deployment package address.</p>
<b>Version</b> (applicable to applications deployed by using JAR and WAR packages)	The deployment package version. You can use a timestamp as the version number.
<b>Time Zone</b> (applicable to applications deployed by using JAR and WAR packages)	Select a time zone from the drop-down list based on your requirements.

- ii. In the **Release Policy** section, configure release policy parameters. Description of release policy parameters

Parameter	Description
<b>Number of First-batch Instances for Phased Release</b>	<p>The number of application instances released in the first batch. The current number of instances for the application appears on the right side. The number of instances for canary release cannot exceed 50% of the total number of instances. This makes the application stable.</p> <div> <b>Note</b> After the canary instance group is released, you must manually release the remaining batches.</div>
<b>Remaining Batches</b>	<p>After the first batch is released, the remaining application instances are released based on the specified batches.</p>
<b>Batch Mode</b>	<p>The following modes are supported:</p> <ul style="list-style-type: none"><li>▪ <b>Automatic:</b> automatically releases application instances in batches based on the values of the <b>Interval</b> parameter. <b>Interval:</b> the release interval for the remaining batches. Unit: minute.</li><li>▪ <b>Manual:</b> manually triggers the release of the next batch.</li></ul> <div> <b>Note</b> <b>Batch Mode</b> appears only when the value of the Remaining Batches parameter is greater than 1.</div>
<b>Intra-Batch Deployment Interval</b>	<p>If the number of application instances is greater than 1 in a batch, the application instances are deployed at this interval. Unit: second.</p>


The **Publish Policy Configuration** section on the right shows the canary release process based on the configuration.

iii. Specify canary release rules.

EDAS supports the following canary release rules: **Canary Release by Content** and **Canary Release by Ratio**.

Parameters of canary release rules


Tab	Parameter	Description
Canary Release by Content	<b>Protocol Type</b>	<ul style="list-style-type: none"> <li>▪ <b>Spring Cloud</b>: Path is required.</li> <li>▪ <b>Dubbo</b>: Select <b>Service</b> and <b>Method</b> are required.</li> </ul>
	<b>Conditional Mode</b>	Select <b>Meet all the following conditions</b> or <b>Meet any of the following conditions</b> .
	<b>Conditions</b>	<ul style="list-style-type: none"> <li>▪ <b>Spring Cloud</b>: Specify specific parameters based on the <b>Cookie</b>, <b>Header</b>, or <b>Parameter</b> type.</li> <li>▪ <b>Dubbo</b>: Specify specific parameters based on <b>Parameter</b> and <b>Expression for Getting Parameter Values</b> of your application.</li> </ul>
Canary Release by Ratio	<b>Traffic Ratio</b>	Traffic is forwarded to the current canary instance group based on the configured ratio.

 **Note** Click **Add Inbound Traffic Rule** to create multiple inbound traffic rules. Multiple rules can take effect at the same time.

iv. (Optional) Configure advanced application settings.

You can configure settings such as start up commands, environment variables, and application lifecycle management. For more information, see the steps for configuring advanced application settings in [Deploy applications in Container Service Kubernetes clusters by using WAR or JAR packages](#).

After a canary release is started, EDAS deploys the new application version to the specified canary instance group. The **Change Records** page displays the deployment progress and status.

 **Note** If you need to monitor whether the canary traffic meets your expectation, see [Monitor canary traffic](#).

7. After the traffic verification is complete, click **Start Next Batch** on the **Change Records** page to complete the subsequent batch release. If you find issues during the verification process, click **Rollback** in the upper-right corner of the **Change Records** page. In the **Confirm** dialog box, click **OK**.

## Verify the result

After the canary release is complete, check whether the version of the **deployment package** is the newly deployed application version at the top of the **Application Overview** page.

## 1.6.1.5. Batch release (applicable to Kubernetes clusters)

In application release and product iteration, batch release is often used to control release risks.

### Introduction

Batch release is the process in which only some instances of an application are upgraded based on batches. If an error occurs during batch release, you can terminate the upgrade process and roll back the instances. After the error is fixed, you can deploy the new version of the application to the instances again.

When an application that is deployed in a Kubernetes cluster is released in batches, application instances are evenly distributed to each batch for deployment. If the instances cannot be evenly distributed, a small number of instances are allocated for the earlier batches and a large number of instances are allocated for the later batches.

### Scenario

Assume that an application contains 10 instances and each application instance is deployed in the V1 version. Now, each application instance must be upgraded to the V2 version.

Assume that all the instances in the application are deployed in three batches. The following figure shows the release process based on the batch release policy.

### Usage notes

When you use batch release for an application in a Kubernetes cluster, you need to create a deployment job.

### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select a **region** and a **Namespaces** and click the name of a specific application.
4. On the **Basic Information** tab, click **Deploy Application** in the upper-right corner.
5. On the **Select Deployment Mode** page, click **Start Deployment** in the **Batch Release** section.
6. On the **Batch Release** page, upload the application deployment package of the new version.

Parameter	Description
Application Runtime Environment	By default, the value is <b>Standard Java Application Runtime Environment</b> .
Java Environment	Select <b>Open JDK 8</b> , <b>Open JDK 7</b> , <b>JDK 8</b> , <b>JDK 7</b> , or <b>Dragonwell 8</b> .

Parameter	Description
File Uploading Method	<p>Select the following two methods for uploading: <b>Upload JAR Package</b> or <b>JAR Package Address</b>.</p> <div> <p><b>Note</b> The file uploading method must be consistent with the method of the previous deployment. The file uploading method includes JAR packages, WAR packages, or images. This topic uses JAR packages as the file uploading method.</p> </div>
Upload JAR Package	When you select <b>Upload JAR Package</b> for <b>File Uploading Method</b> , click <b>Select File</b> to upload the JAR package.
JAR Package Address	<p>When you select <b>JAR Package Address</b> for <b>File Uploading Method</b>, enter the JAR package address.</p> <div> <p><b>Note</b> When an Object Storage Service (OSS) file that has an authentication signature accesses the URL, EDAS caches the file during the deployment for subsequent operations, such as rollback and scale-out.</p> </div>
Version	Enter the JAR package version. You can also click <b>Use Timestamp as Version Number</b> on the right side to automatically generate the corresponding timestamp.
Time Zone	See the UTC time zone that corresponds to the specified region.

## 7. Specify Release Policy.

Release Policy

**1** If your application has used the following Kubernetes native features or configurations (including HPA, Rancher, Istio, and features and configurations that depend on Deployment.Metadata.Name or Deployment.Metadata.Uid), do not use canary release or phased release. Otherwise, these Kubernetes native features or configuration will be abnormal after the application is deployed. For more information, see [Instructions](#).

Release Batch **1**

Batch Mode Automatic

Interval  min

Deployment Interval Between Batches  Seconds

Publish Policy Configuration

1 Start Deployment

2 [Auto Start] Batch 1: 2  
Batch Interval 5min

3 [Auto Start] Batch 2: 2

4 End

The following table describes the **Release Policy** parameters.

Parameter	Description
<b>Release Batch</b>	The number of batches for releasing application instances to finish canary release.
<b>Batch Mode</b>	<p>The following two methods are available: <b>Automatic</b> and <b>Manual</b>.</p> <ul style="list-style-type: none"><li>◦ <b>Automatic</b>: automatically releases application instances in batches based on the value of the <b>Interval</b> parameter. <b>Interval</b>: the release interval for the remaining batches. Unit: minute.</li><li>◦ <b>Manual</b>: manually triggers the release of the next batch.</li></ul>
<b>Intra-Batch Deployment Interval</b>	The deployment time interval between application instances when the number of application instances is greater than one. Unit: second.

8. (Optional) Configure advanced settings, such as startup commands, environment variables, and application lifecycle management. For more information, see the configuration steps of advanced application settings in [Deploy applications in Container Service Kubernetes clusters by using WAR or JAR packages](#).
9. After you specify the parameters, click **OK**.

## Verify the result

On the **Change Records** page, view the batch release status of applications. The batch release succeeds until all the batches are run.

On the Application Details page, view the instance deployment information. When the instance version is changed to V2 and all the instances are in the **Running** state, the release is successful.

## Roll back an application

During a batch release, if at least one application instance is not upgraded to the new version, the release is considered in the **Executing** state. When you monitor an upgrade, if you notice that an application instance in the first batch stops responding, you can go to the Change Details page of the instance and click **Roll Back** to roll back the released instance to the previous service package and configuration.

If an exception occurs during batch release.

- Exceptions such as unavailable deployment packages or health check failures may lead to failures of application upgrades. The current application changes are automatically terminated and the application is rolled back.
- During the upgrade, the maximum timeout period for a single batch is 30 minutes. If the change process is suspended due to time-out, you must go to the **Change Details** page to manually terminate the release process and roll back the application.

## 1.6.2. Application lifecycle management of Kubernetes clusters

This topic describes how to perform management operations such as create, update, scale, and delete applications in ACK clusters.

### Create an application


You can develop an application based on the Apache Dubbo, Spring Cloud, or HSF framework. You can also build this application as a WAR package, JAR package, or image and deploy the application to a Kubernetes cluster in EDAS.

- [Deploy microservice applications in a Kubernetes environment](#)
- [Deploy applications in Container Service Kubernetes clusters by using WAR or JAR packages](#)

### Deploy an application

Assume that an application is created in EDAS and is not deployed. On the **Basic Information** tab, click **Deploy Application** in the upper-right corner to deploy and release the application.

On the **Basic Information** tab, click **Deploy Application** in the upper-right corner to upgrade a deployed application.

 **Notice** When you upgrade the application, you must select a deployment package type that is the same as that of the initial deployment.

### Scale out and scale in an application

Application scale-out indicates that the number of application instances is increased to increase the computing capacity of the application. Application scale-in indicates that the number of application instances is reduced to reduce the computing capacity of the application. Assume that application instances are overloaded. On the **Basic Information** tab, click **Application Scaling** in the upper-right corner to increase the number of application instances. Assume that application instances are idle. On the **Basic Information** tab, click **Application Scaling** in the upper-right corner to reduce the number of application instances.

### Delete an application

On the **Basic Information** tab, click **Delete Application** in the upper-right corner to delete the application. After an application is deleted, all the information about this application is deleted and all the instances (pods) of this application are released.

## 1.6.3. View application changes

After you perform lifecycle operations on applications in the Enterprise Distributed Application Service (EDAS) console, you can go to the Application Details page to check the change status or go to the Change Logs page to check the change logs. Application lifecycle operations include the deployment, startup, scale-out, and scale-in of applications.

### View change details

This topic uses an example of application deployment to describe how to view application changes.

1. After you perform a change operation on an application, return to the application details page. At the top of the application details page, the following message appears: **A change process is**

ongoing for this application. The application is in Executing state.

2. Click **View Details** to go to the **Change Details** page. On this page, you can view the change information and the real-time status of the application.
  - Change summary information: includes the change process ID, the execution status, and the change type.
  - Information about change process execution: includes each stage of the entire process and the specific tasks at each stage.
    - If a task is executed on a single instance, icons are used to mark the tasks and the execution results in each stage.
    - If a task is executed on multiple instances, the executed tasks and the execution results are displayed by instance in each stage.
3. In the left part of the change process execution section, click a stage below **Batch x Change**, and then click an instance IP address on the right side. Then, you can view the task execution status of the instance in this stage.
4. Click a task to view the task execution logs.

The system automatically unfolds the logs for failed tasks. For information about how to handle exceptions, see [How do I resolve problems in a change process?](#).

## View application change records

1. In the left-side navigation pane of the application details page, click **Change Records** to view all change records of this application.
2. In the Actions column, click **View** to view the change details and the details of each operation.

## 1.6.4. View application events


In the EDAS console, you can view the event information about applications that are deployed in Kubernetes clusters. This allows you to obtain the information about the application running status, and focus on problems in a timely manner.

### Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select **Region** and **Namespace**. Select **Container Service for Kubernetes Cluster** from the **Cluster Type** drop-down list and click the name of the application that you want to view.
4. In the left-side navigation pane, click **Application Event**.
5. Specify the following filter conditions for application events based on your requirements and click **Search For**. Filter conditions for application events

Parameter	Description
Source Type	The values are <b>Deployment</b> , <b>Pod</b> , <b>Service</b> , and <b>HorizontalPodAutoscaler</b> .
Source Name	Enter the event source name, such as an application name or an application instance name.

Parameter	Description
Cause of Event	Enter the event cause, such as FailedScheduling of a pod.
Event Level	The values are <b>Warning</b> and <b>Normal</b> .

 **Notice** Take note of the warning-level events and check your application.

## 1.6.5. Logs

### 1.6.5.1. View real-time logs

Real-time logs are available in applications that are deployed in ACK clusters.

#### View real-time logs

You can view real-time logs to troubleshoot pod-related problems when an application is abnormal.

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select **Region** and **Namespace**. Select **Self-managed Kubernetes Cluster** from the **Cluster Type** drop-down list and click the name of the application that you want to view.
4. On the Application Details page, click **Real-Time Logs** in the left-side navigation pane.
5. To view real-time logs, select the required pod from the **Pod Name** drop-down list and the required container from the **Container** drop-down list.


### 1.6.5.2. Log directories

You can check instance and application logs to troubleshoot application exceptions. Enterprise Distributed Application Service (EDAS) allows you to bookmark log directories, view logs, and search for logs.

#### Bookmark log directories

The Log Directories page displays the default log directories of EDAS. You can bookmark log directories of your applications. You can view the instance logs in a log directory after you bookmark the log directory.

You can also **add the log directory to Log Service** after you bookmark the log directory so that the application logs in the directory can be viewed and searched on the **Log Search** page.

 **Note** Only log directories can be bookmarked and removed from bookmarks.

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the page, select a namespace. On the **Applications** page, click the name of the desired application.
3. On the Application Details page, choose **Logs > Log Catalog**. On the **Log Directories** page, click

### Add Log Directory into Bookmark.

4. In the **Add Application Log Directory into Bookmark** dialog box, enter the directory in **Application Log Directory**, choose whether to turn on **Add Path to Log Service**, and then click **Add**.


Follow the following requirements when you enter the application log directory:

- The directory must be in /home/admin.
- The complete directory must contain log or logs.
- The directory name must end with a slash (/) to indicate that it is a folder.

## View instance logs

After you bookmark application logs, you can view logs of instances and applications.

1. On the Application Details page, choose **Logs > Log Catalog** from the left-side navigation pane.
2. On the **Log Directories** page, click the > button on the left. In the log list that expands, click **View** in the Actions column.


 **Note** In **File for Log Frame Configuration**, you can view details of a log and modify the level of the log. The log levels are TRACE, DEBUG, INFO, WARN, and ERROR in ascending order. After the log level is changed, the system displays the logs of the set level and higher levels at the bottom of the page.

3. On the **Real-Time Logs** page, select an instance from the **ECS Instance ID/Name/IP** drop-down list on the right side to view the log details of this instance.  
Click **Enable Real-time Additions** in the lower-right corner of the page to ensure that the latest appended content to the file has been added (similar to the `tailf` command).

## Grant Log Service permissions to a RAM user

If you are a Resource Access Management (RAM) user, you must use Log Service as the RAM user, including adding log directories or files to Log Service, viewing application logs, or performing distributed search. In addition, you must use an Alibaba Cloud account to authorize the RAM user in the RAM console. The procedure is as follows:


1. Log on to the RAM console with an Alibaba Cloud account.
2. In the left-side navigation pane, click **Users**. On the **Users** page, find the target RAM user such as *d-octest* in the user list, and click **Add Permissions** in the Actions column.
3. In the **Add Permissions** pane, select **System Policy** under **Select Policy** and enter *log* in the search box next to it. Click **AliyunLogReadOnlyAccess** to add this permission to the Selected list on the right side, and then click **OK**.

 **Note** **Principal** is loaded by default and does not need to be set. To grant permissions to multiple RAM users, enter keywords in the search box and search for and add the RAM users in the **Principal** field.

4. On the **Authorization Result** page, view the RAM user and the granted permission, and then click **Complete**.

## Remove log directories from bookmark

When you remove a log directory from bookmark, the specified log directory is removed. After the log directory is removed, it is no longer displayed on the **Log Directories** page. You cannot view instance logs in this directory, but the actual log directory and files are not deleted.

 **Note** A default directory can also be removed from bookmarks.

When you remove a log directory from bookmarks, you can also choose **Remove Log Directory from Bookmark**.

- If the log directory is not removed from Log Service, you can view the original application logs in the directory.
- If the log directory is removed from Log Service, you cannot view the application logs in this directory.
- On the **Log Directories** page, select a log directory, and click **Remove Log Directory from Bookmark**.
- In the **Remove Selected Log Directory from Bookmark** dialog box, verify the target log directory, choose whether to enable **Delete Project and Logstore Related with the Log**, and then click **Confirm**.

## 1.6.6. Monitoring

### 1.6.6.1. Dashboard

Based on different groups, the dashboard displays the overall metrics related to service provisioning, service consumption, and infrastructure monitoring by using charts.

#### Context

- **Service provisioning**: displays the metrics for the RPC and HTTP services.
- **Service consumption**: displays the metrics for database access.
- **Infrastructure monitoring**: displays the metrics for CPU, load, memory, disk, and network.

#### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management**. On the **Applications** page, click the name of the target application.
3. In the left-side navigation pane, choose **Application Management**. On the **Applications** page, click the **name of the application** to be monitored.
4. On the **Application Details** page, choose **Application Monitoring > Dashboard** from the left-side navigation pane.  
On the **Dashboard** page, you can view the monitoring charts for service provisioning, service consumption, and infrastructure monitoring.
  - Place the pointer over a point on an abscissa of a monitoring chart to view the data and status at that time point.
  - Click a project name, such as **RPC Service**, at the top of a monitoring chart to switch to the **Service Monitoring** tab and view details.

### 1.6.6.2. Infrastructure monitoring

EDAS collects data from the ECS instances that run applications and provides the CPU, memory, load, network, and disk metrics by instance or cluster based on the analysis results.

## Prerequisites

Infrastructure monitoring involves ECS instances. Ensure that your RAM user is authorized. For the authorization procedure, see *the Apsara Stack Console User Guide and read the RAM management* topic.

## Context

Due to the latency between data collection and data analysis, EDAS cannot provide real-time dashboards. The current latency is 2 minutes. All monitoring data is collected and processed by application.

## Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management**. On the **Applications** page, click the name of the target application.
3. On the Application Details page, choose **Application Monitoring > Infrastructure Monitoring** from the left-side navigation pane.  
On the Infrastructure Monitoring page, cluster data from the past half an hour appears by default.
4. Select a type of monitoring data.  
Monitoring data includes group data and single-instance data.  
These two data types correspond to the same set of monitoring metrics related to clusters (groups) and instances (single instance), including:
  - i. **CPU Data** indicates the CPU usage, which is the sum of the user utilization and system utilization. The **group data** graph shows the average value of this data for all instances in the cluster.
  - ii. **Memory Data** indicates the total size and actual utilization of the physical memory. The **group data** graph shows the sum of this data for all instances in the cluster.
  - iii. **Load Data** indicates the "1 min load" field in the system load. The **group data** graph shows the average value of this data for all instances in the cluster.
  - iv. **Network Speed Data** indicates the read and write speeds of the NIC. If an ECS instance contains multiple NICs, this parameter indicates the total read and write speeds of all NICs whose names start with "eth". The **group data** graph shows the average value of this data for all instances in the cluster.
  - v. **Disk Data** indicates the total size and actual utilization of all disks attached to the instance. The **group data** graph shows the value of this data for all instances in the cluster.
  - vi. **Disk Reading and Writing Speed** indicates the sum of the read and write speeds of all disks attached to the instance. The **group data** graph displays the average value of the data for all instances in the cluster.
  - vii. **Disk Reading and Writing Numbers** indicates the sum of the input/output per second (IOPS) of all disks attached to the instance. The **group data** graph displays the average value of the data for all instances in the cluster.
5. Set Time Interval. You can set Time Interval to Half an Hour, 6 Hours, One Day, or 1 Week.
  - **Half an Hour:** collects monitoring data from the past half an hour. Time Interval is set to Half an Hour by default for infrastructure monitoring. In this statistical cycle, data is collected every minute, which is the finest query granularity provided by EDAS.

- **6 Hours:** collects monitoring data from the past 6 hours. In this statistical cycle, data is collected every 5 minutes.
- **One Day:** collects monitoring data from the past 24 hours. In this statistical cycle, data is collected every 15 minutes.
- **1 Week:** collects monitoring data from the past seven days. In this statistical cycle, data is collected every hour, which is the longest statistical cycle provided by EDAS.

 **Note**

Start Time and End Time on the page indicate the time span of the current view. When you set one of the parameters, the corresponding parameter is automatically updated. For example, if you select Half an Hour and set End Time to 2016-05-20 12:00:00, then Start Time automatically changes to 2016-05-20 11:30:00.  
After setting, monitoring data is automatically updated based on the selected interval.

6. (Optional) View the enlarged graph of a detailed metric.  
When viewing a dashboard, you can click **Zoom In** under a metric to view the enlarged graph of the metric, and adjust the interval in the enlarged graph.

### 1.6.6.3. Service monitoring

By collecting and analyzing tracked logs in different network call middleware products, you can obtain the traces of systems for a single request. This helps sort out application request portals and service call sources and dependencies, analyze system call bottlenecks, estimate the link capacity, and quickly locate exceptions.

#### Procedure

##### Monitoring service

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management**.
3. On the Applications page, click the **name of the application** to be monitored.
4. On the Application Details page, choose **Application Monitoring > Service Monitoring** from the left-side navigation pane.

Monitoring metrics:

- **RPC Services Provided:** displays the call records on the RPC service provided by the current application.
- **RPC Call Source:** displays the applications that call the RPC service provided by the current application.
- **RPC Call Dependency:** displays the applications whose services are called by the current application.

The **Summary** tab page of the default group appears by default.

5. (Optional) Set the monitoring conditions and click **Update** to update the monitoring data.

Name	Description
Latest	The data at the current time is displayed by default. Select a period from the drop-down list.

Name	Description
Sort By	Data is sorted by QPS by default. Select an option from the drop-down list to sort data by the elapsed time or errors/s (average QPS errors per minute).
Results	10 is selected by default. Select the number of results to be displayed from the drop-down list. The options are 1, 5, 30, 50, 100, and Unlimited.
Display	Results are displayed in blocks by default. You can also set the display mode to multi-graph or table.

6. View monitoring data.

For information about monitoring metrics, see [Application monitoring](#).

7. Click a metric of a column in the monitoring graph. The custom query page appears. You can view the monitoring data of the metric.

8. In the Metrics section, select metrics to view data of different groups.

**View traces**

9. In the monitoring graph, click **View Trace** next to a call service or called service and choose **Trace Analysis > Trace Query**.

10. On the **Trace Query** page, you can view the traces between the application and the call service or called service.

**Monitor drilled-down applications**

11. On the **RPC Services Provided**, **RPC Call Source**, or **RPC Call Dependency** tab page, click **Source Application**, **Called Service**, or **Call Service** next to **Drill Down** at the top of the monitoring graph. The monitoring page of the drilled-down application appears.

12. Monitor data of the drilled-down application. The method for monitoring the data of a drilled-down application is the same as that for monitoring the application.

## 1.6.6.4. Advanced monitoring

Application Real-Time Monitoring Service (ARMS) is an application performance management (APM) product of Alibaba Cloud. EDAS can seamlessly connect to ARMS. You need only to enable advanced monitoring to use the APM feature of ARMS so that you can monitor applications in EDAS. This way, you can further manage the performance of your applications.

### Enable ARMS advanced monitoring for an application in the EDAS console

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**, and click the name of the application for which you want to enable the ARMS application monitoring.
3. In the left-side navigation pane, choose **Application Monitoring > Advanced Monitoring** and click **Enable Advanced Application Monitoring**.
4. In the **Tip** dialog box, click **OK**.

### View the monitoring data of the EDAS application in ARMS

After the ARMS application monitoring is enabled, you can log on to the ARMS console to view the detailed monitoring data of the application.

1. Log on to the ARMS console.
2. In the left-side navigation pane, choose **Application Monitoring > Applications**.
3. On the **Applications** page, click the name of the application.  
On the page that appears, you can view metrics, including **Health Rate**, **Requests**, **Errors**, **Response Time**, **Exceptions**, and **Status**.

After applications in EDAS are connected to ARMS advanced monitoring, you can use ARMS to monitor the applications. For more information about how to use the application monitoring feature, see the *Overview* topic in **User Guide > Application monitoring**.

## 1.6.6.5. Alerts

### 1.6.6.5.1. Create contacts


When an alert rule is triggered, notifications are sent to the contact group that you specified. Before you create a contact group, you must create contacts. When creating a contact, you can specify the mobile phone number and email address of the contact to receive notifications. You can also provide a DingTalk chatbot webhook URL used to automatically send alert notifications.

#### Prerequisites


To add a DingTalk chatbot as a contact, you must obtain its webhook URL first. For more information, see **Enable DingTalk chatbot alert** [Configure a DingTalk chatbot to send alert notifications](#).

#### Procedure

1. Log on to the [ARMS console](#).
2. In the left-side navigation pane of the console, choose **Alerts > Contact Management**.
3. Log on to the console. Click the target application in **Applications**. In the left-side navigation pane, choose **Alerts > Alert Policies**.
4. On the **Alert Policies** page, click **Create Alert** in the upper-right corner.
5. On the **Contacts** tab, click **Create Contact** in the upper-right corner.
6. In **Create Contact** dialog box, edit contact information.
  - To add a contact, enter the **Name**, **Phone Number** and **Email**.

 **Note** The phone number and email address cannot be blank at the same time. Each phone number or email address must be used for only one contact. You can create a maximum of 100 contacts.

- To add a DingTalk chatbot, enter the name and the webhook URL of the chatbot.

 **Note** For more information about how to obtain the webhook URL of the DingTalk chatbot, see **Enable DingTalk chatbot alert** [Configure a DingTalk chatbot to send alert notifications](#).

#### What to do next

- To search for contacts, on the **Contacts** tab, select **Name**, **Phone Number**, or **Email** in the drop-down list, then enter the entire or a part of the selected name, phone number or email in the search box, and click **Search**.

- To edit a contact, click **Edit** in the **Actions** column of the contact, edit the information in the **Update Contact** dialog box, then click **OK**.
- To delete a single contact, click **Delete** in the **Actions** column of the contact, then click **Delete** in the **Delete** dialog box.
- To delete multiple contacts, select the target contacts, click **Batch Delete Contacts**, then click **OK** in the **Note** dialog box.

## Related information

- [Create a contact group](#)
- [Configure a DingTalk chatbot to send alert notifications](#)
- [Create ARMS alerts](#)
- [Manage alerts](#)

### 1.6.6.5.2. Create and manage a contact group


When you create an alert rule, you can specify a contact group as the recipient of alert notifications. If an alert is triggered, Enterprise Distributed Application Service (EDAS) sends alert notifications to the contacts in this contact group. This topic describes how to create a contact group.

## Prerequisites

A contact is created. For more information about how to create a contact, see [Create contacts](#).

## Create a contact group

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select **User-built K8s Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to manage.
4. In the left-side navigation pane, choose **Alerts > Contacts**.
5. On the **Contacts** page, click **Contact Groups**. On the **Contact Groups** tab, click **New Contact Group** in the upper-right corner.
6. In the **Create Contact Group** dialog box, specify **Group Name** and **Contact Members**, and click **OK**.

 **Note** If no contact appears in the **Contact Members** section, create a contact first. For more information, see [Create contacts](#).

## Manage a contact group

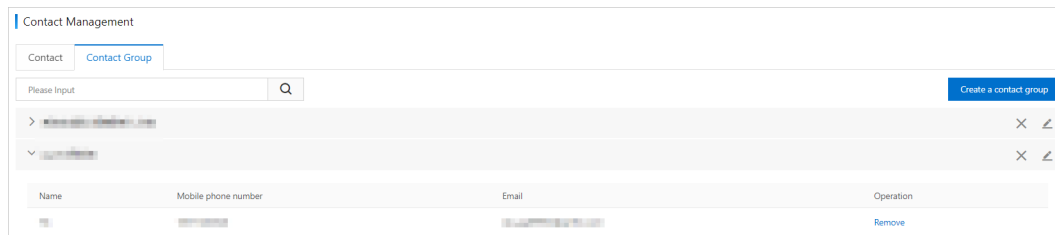
- To search for a contact group, enter all or some characters of the contact group name in the search box of the **Contact Groups** tab. Then, click **Search**.

 **Notice** Keywords for group searches are case-sensitive.

- To edit a contact group, click the pencil icon on the right side of the contact group. In the **Edit Contact Group** dialog box, edit the information.
- To view the contacts in a contact group, click the downward arrow on the right side of the contact group to expand the group information.

View contacts in a contact group

### view contacts in a contact group



**Note** You can remove one or more contacts from a contact group that is expanded. To remove a contact, find the contact that you want to remove, and click **Delete** in the **Operating** column.

- To delete a contact group, click the X icon on the right side of the contact group.

**Notice** Before you delete a contact group, make sure that no monitoring job is running. Otherwise, alerting and other features may become ineffective.

## 1.6.6.5.3. Create an alert

You can specify alert rules for specific monitored objects by creating alerts. When a rule is triggered, the system sends an alert notification to the specified contact group in specified notification mode. This reminds you to take necessary measures to solve the problem.

### Prerequisites

- A monitoring job is created. For more information, see [Create an application monitoring job](#) and [Create a custom monitoring job](#).
- Contacts are created. Only contact groups can be specified for the notification receiver of an alert.

### Context

Default alert conditions:

- To prevent you from receiving a large number of alert notifications in a short period of time, the system sends only one message for repeated alerts within 24 hours.
- If no repeated alerts are generated within 5 minutes, the system sends a recovery email to notify you that the alert has been cleared.
- After a recovery email is sent, the alert status is reset. If this alert is generated again, it is considered as a new alert.

The alert widget is essentially a data display method for datasets. When you create an alert widget, a dataset is created to store the underlying data of the alert widget.


**Note** New alerts take effect within 10 minutes. It takes 1 to 3 minutes to check an alert.

## Create an application monitoring alertCreate an alert

To create an alert for an application monitoring job on the number of Java Virtual Machine-Garbage Collection (JVM-GC) times in corresponding-period comparison, perform the following operations:

- Log on to the [ARMS console](#).
- In the left-side navigation pane, choose **Alerts > Alert Policies**.

3. On the **Alert Policies** page, choose **Create Alarm > Application Monitoring Alarm** in the upper-right corner.
4. Log on to the console. On the **Applications** page, click the application that you want to manage. In the left-side navigation pane, choose **Alerts > Alert Policy Management**.
5. On the **Alert Policies** page, click **Create Alarm** in the upper-right corner.
6. In the **Create Alarm** dialog box, enter all the required information and click **Save**.
  - i. Specify **Alarm Name**. For example, you can set this parameter to alert on JVM-GC times in corresponding-period comparison.
  - ii. Select an application for **Application Site**, and an application group for **Application Group**.
  - iii. Select the type of the monitoring metrics from the **Type** drop-down list. For example, you can select **JVM\_Monitoring**.
  - iv. Set Dimension to **Traverse**.
  - v. Specify Alarm Rules.
    - a. Select **Meet All of the Following Criteria**.
    - b. Edit the alert rule. For example, an alert is triggered when the value of N is 5, and the average value of JVM\_FullGC increases by 100% compared with that in the previous hour.

 **Note** To add another alert rule, click the + icon on the right side of **Alarm Rules**.

- vi. Specify Notification Mode. For example, you can select Email.
- vii. Specify Notification Receiver. In the **Contact Groups** box, click the name of a contact group. If the contact group appears in the **Selected Groups** box, the setting is successful.

**Create Alarm**

\*Alarm Name:

\*Application Site:

Application Group:

\*Type:  Dimension:  Non-IP:

\*Alarm Rules: ☒ Meet All of the Following Criteria ☐ Meet Any of the Following Criteria

\*Last N Minutes: N=

\*Notification Mode: ☐ SMS ☐ Email ☐ Ding Ding Robot ☐ Webhook

\*Notification Receiver:

Contact Groups	Selected Groups
<input type="checkbox"/> Group 1	
<input type="checkbox"/> Group 2	
<input type="checkbox"/> Group 3	
<input type="checkbox"/> Group 4	
<input type="checkbox"/> Group 5	

Alert advanced options doc: [Alert advanced options doc](#)

[Advanced Configuration](#)

## Create a browser monitoring alert

To create a Page\_Metric alert to monitor JS\_Error\_Rate and JS\_Error\_Count for a browser monitoring job, perform the following steps:

1. In the left-side navigation pane, choose **Alerts > Alert Policies**.
2. On the **Alert Policies** page, choose **Create Alarm > Browser Monitoring Alarm** in the upper-right corner.
3. In the **Create Alarm** dialog box, enter all the required information and click **Save**.
  - i. Specify Alarm Name. For example, you can set this parameter to Page\_Metric alert.
  - ii. From the **Application Site** drop-down list, select the monitoring job that you created.
  - iii. Select the type of the monitoring metric from the **Type** drop-down list. For example, you can select **Page\_Metric**.
  - iv. Set Dimension to **Traverse**.
  - v. Specify Alarm Rules.
    - a. Select **Meet All of the Following Criteria**.
    - b. Edit the alert rule. For example, an alert is triggered when the value of N is 10 and the average value of JS\_Error\_Rate is at least 20.
    - c. To add another alert rule, click the **+** icon on the right side of Alarm Rules. For example, an alert is triggered when the value of N is 10 and the value of JS\_Error\_Count is at least 20.

- vi. Specify Notification Mode. For example, you can select SMS and Email.
- vii. Specify Notification Receiver. In the **Contact Groups** box, click the name of a contact group. If the contact group appears in the **Selected Groups** box, the setting is successful.

Create Alarm ?

\*Alarm Name:

\*Application Site:

\*Type:  [Dimension](#)

\*Alarm Rules: ☒ Meet All of the Following Criteria ☐ Meet Any of the Following Criteria

\*Last N Minutes: N=

\*Notification Mode: ☐ SMS ☐ Email ☐ Ding Ding Robot ☐ Webhook

\*Notification Receiver:

Contact Groups

Selected Groups

Alert advanced options doc: ?

[Advanced Configuration](#)

Save Cancel

## Create a custom monitoring alert

To create a user access alert for a custom monitoring job, perform the following operations:

1. In the left-side navigation pane, choose **Alerts > Alert Policies**.
2. On the **Alert Policies** page, choose **Create Alarm > Custom Monitoring Alarm** in the upper-right corner.
3. In the **Create Alarm** dialog box, enter all the required information and click **Save**.
  - i. Specify Alarm Name field. For example, you can set this parameter to user access notification.
  - ii. Set Type to **Create Alert Based On Existing Drilled-down Dataset**.
  - iii. Specify Alarm Variable Definition. Select a dataset for variable *a* and set Drill-down Dimension to **Traverse**.

**Note** To define another alert variable, click **+** on the right side of **Alarm Variable Definition**. In the dialog box that appears, define variable *b*.

iv. Specify Alarm Rules.

- a. Select **Meet All of the Following Criteria**.
- b. Edit the alert rule. For example, an alert is triggered when the value of N is 3 and the average number of agents that you created is at least 0.

**Note** You can also include a simple compound indicator in the alert rule. For example, an alert is triggered when the value of N is 3 and the average value of dataset A divided by dataset B is at least 5.

- v. Set Notification Mode. For example, you can select Email.
- vi. Specify Notification Receiver. In the **Contact Groups** box, click the name of a contact group. If the contact group appears in the **Selected Groups** box, the setting is successful.


## Create a Prometheus monitoring alert

To create an alert for a Prometheus monitoring job, such as an alert on network receiving load, perform the following operations:


1. You can select one of the two available methods to go to the Create Alarm page.
  - o On the **New Dashboard** page of the **Prometheus Grafana dashboard**, click the icon to go to the ARMS Prometheus **Create Alarm** dialog box.
  - o In the left-side navigation pane of the console, choose **Alerts > Alert Policies**. On the **Alert**

**Policies** page, choose **Create Alarm > Prometheus** in the upper-right corner.

2. In the **Create Alarm** dialog box, enter all the required information and click **Save**.
  - i. Specify Alarm Name. For example, you can set this parameter to alert of network receiving load.
  - ii. Select the **cluster** of the Prometheus monitoring job.
  - iii. Set **Type** to **grafana**.
  - iv. Select the specific **dashboard** and **chart**.
  - v. Specify Alarm Rules.
    - a. Select **Meet All of the Following Criteria**.
    - b. Edit the alert rule. For example, an alert is triggered when the value of N is 5 and the average value of Received\_Bytes (MB) is at least 3.

 **Note** A Grafana chart may contain data of Curve A, Curve B, and Curve C. You can select one of the curves to monitor.

- c. In the **PromQL** field, edit the existing PromQL statement or enter a new PromQL statement.

 **Notice** An error may be reported if a PromQL statement contains a dollar sign (\$). You must delete the equal sign (=) and the parameters on both sides of the equal sign (=) from the statement that contains the dollar sign (\$). For example, modify `sum(rate(container_network_receive_bytes_total{instance=~"^$HostIp.*"}[1m]))` to `sum(rate(container_network_receive_bytes_total[1m]))`

- vi. Specify Notification Mode. For example, you can select SMS.
- vii. Specify Notification Receiver. In the **Contact Groups** box, click the name of a contact group. If the contact group appears in the **Selected Groups** box, the setting is successful.

Create Alarm

\*Alarm Name:

\*Cluster:

arms-demo-fuling-zhuanyouban-en

\*Type:

grafana

\*Dashboard:

Etcd by Prometheus

\*Chart:

Etcd has a leader?

\*Alarm Rules:

☒ Meet All of the Following Criteria

☐ Meet Any of the Following Criteria

\*Last N Minutes:

N=

1-60

A

Average

Greater than or equ

Thresho

\*PromQL:

max(etcd\_server\_has\_leader)

\*Notification Mode:

☐ SMS

☐ Email

☐ Ding Ding Robot

☐ Webhook

\*Notification Receiver:

Contact Groups

Selected Groups

Alert advanced options doc:

Advanced Configuration

Save

Cancel

**Description of basic fields**  
The following table describes the basic fields in the **Create Alarm** dialog box.

Create Alarm ?

\*Alarm Name:

\*Application Site:

a3[cn-hangzhou]

\*Type:

Custom\_Qu

Dimension:

\*Alarm Rules:

☒ Meet All of the Following Criteria

☐ Meet Any of the Following Criteria

\*Last N Minutes:

N=

1-60

DNS Lookup

Average

Greater than or equ

Thresho

\*Notification Mode:

☐ SMS

☐ Email

☐ Ding Ding Robot

☐ Webhook

\*Notification Receiver:

Contact Groups

Selected Groups

Alert advanced options doc:

Advanced Configuration

Alarm Quiet Period:

Alarm Data Revision:

☐ Set 0

☐ Set 1

☒ Set Null (Won't Trigger)

Alarm Severity:

Warn

Effective Time::

00

:

00

To

23

:

59

Notification Time:

00

:

00

To

23

:

59

Notification Content:

[Alibaba Cloud]ARMS Notification -

Subtitle(Optional)

Alarm Name: \$AlarmName

Filter Condition: \$AlarmFilter

Alarm Time: \$AlarmTime

Alarm Content: \$AlarmContent

Attention! : This alarm is in progress until the alarm is received, and it will remind you again after 24 hours!

Save

Cancel

Field	Description	Remarks
Application Site	The monitoring job that has been created.	Select a value from the drop-down list.

Field	Description	Remarks
Type	The type of the alert metric.	<p>The metric types for the following three alerts are different:</p> <ul style="list-style-type: none"> <li>Application monitoring alert: displays application entry calls, the statistics for application call types, database metrics, JVM monitoring, host monitoring, and abnormal API calls.</li> <li>Browser monitoring alert: displays page metrics, API metrics, custom metrics, and page API metrics.</li> <li>Custom monitoring alert: allows you to create alerts based on existing drill-down datasets and existing general datasets.</li> </ul>
Dimension	The dimensions for alert metrics (datasets). You can select None, "=", or Traverse.	<ul style="list-style-type: none"> <li>When Dimension is set to None, the alert content shows the sum of all the values of this dimension.</li> <li>When Dimension is set to =, you must enter the specific content.</li> <li>When Dimension is set to Traverse, the alert content shows the dimension content that actually triggers the alert.</li> </ul>
Last N Minutes	The system checks whether the data results in the last N minutes meet the trigger condition.	Valid values of N: 1 to 60.
Notification Mode	Email, SMS, Ding Ding Robot, and Webhook are supported.	You can select multiple modes. To configure a DingTalk chatbot alert, see <a href="#">Configure a DingTalk chatbot to send alert notifications</a> . <a href="#">Configure a DingTalk chatbot to send alert notifications</a>
Alarm Quiet Period	You can enable or disable Alarm Quiet Period. By default, Alarm Quiet Period is enabled.	<ul style="list-style-type: none"> <li>Assume that Alarm Quiet Period is enabled. If data remains in the triggered state, the second alert notification is sent 24 hours after the first alert is triggered. When data is recovered, you receive a data recovery notification and the alert is cleared. If the data triggers the alert one more time, the alert notification is sent again.</li> <li>Assume that Alert Quiet Period is disabled. If the alert is continually triggered, the system sends the alert notification every minute.</li> </ul>
Alert Severity	Valid values include Warn, Error, and Fatal.	-

Field	Description	Remarks
Notification Time	The period during which the alert is sent. No alert notification is sent out of the period, but alert events are recorded.	For information about how to view the alert event records, see <a href="#">Manage alerts</a> .
Notification Content	The custom content of the alert notification.	You can edit the default template. In the template, the four variables, \$AlertName, \$AlertFilter, \$AlertTime, and \$AlertContent, are preset. Other preset variables are not supported. You can customize the rest of the content.

## Description of complex general fields: Alarm Data Revision policy

You can select Set 0, Set 1, or Set Null. Set Null is the default value. This feature is generally used to fix anomalies in data, including no data, abnormal compound indicators, and abnormal period-over-period comparisons.

- Set 0: fixes the checked value to 0.
- Set 1: fixes the checked value to 1.
- Set Null: does not trigger the alert.

Scenarios:

- Anomaly 1: no data  
User A wants to use the alert feature to monitor the number of page views. When User A creates the alert, User A selects Browser Monitoring Alert. User A specifies the alert rule: N is 5 and the sum of the page views is at most 10. If the page is not accessed, no data is reported and no alert is sent. To solve this problem, you can select Set 0 as the data revision policy of the alert. If you do not receive data, the system considers that zero data is received. This meets the alert rule and an alert is sent.
- Anomaly 2: abnormal compound indicators  
User B wants to use the alert feature to monitor the real-time unit price of a product. When User B creates the alert, User B selects Custom Monitoring Alert. User B sets the dataset of Variable a to the current total price, and the dataset of Variable b to the total number of current items. User B also specifies the alert rule: N is 3 and the minimum value of current total price divided by current total items is at most 10. If the current total of items is 0, the value of the compound indicator, current total price divided by current total items, does not exist. No alert is sent. To solve this problem, you can select Set 0 as the data revision policy of the alert. The value of the compound indicator, current total price divided by current total items, is now considered to be 0. This meets the alert rule and an alert is sent.
- Anomaly 3: abnormal period-on-period and period-for-period comparisons  
User C wants to use the alert function to monitor the CPU utilization of the node machine. When User C creates the alert, User C selects Application Monitoring Alert, and specifies the alert rule: N is 3, and the average user CPU utilization of the node machine decreases by 100% compared with the previous monitoring period. If the CPU of the user fails to work in the last N minutes,  $\alpha$  cannot be obtained. This indicates that the period-on-period result does not exist. No alert is sent. To solve this problem, you can select the alert data revision policy as Set 1, and consider the period-on-period comparison result as a decrease of 100%. This meets the alert rule and an alert is sent.

## What's next

You can query and delete alert records in the alert management system.

## 1.6.6.5.4. Manage alerts

On the Alert Policy Management page, you can manage all the alert rules within your Alibaba Cloud account, and query the history of alert events and alert notifications.


### Manage alert rules

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select **User-built K8s Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to manage.
4. In the left-side navigation pane, choose **Alerts > Alert Policy Management**.
5. On the **Alert Rules** tab, enter an alert name in the search box and click **Search**.
6. In the **Actions** column of the search results, perform the following operations as needed to manage an alert rule:
  - To edit an alert rule, click **Edit**. In the **Edit Alarm** dialog box, edit the alert rule and click **Save**.
  - To delete an alert rule, click **Delete** in the Actions column. In the **Delete** dialog box, click **Delete**.
  - To start an alert rule that is disabled, click **Start** in the Actions column. In the **Start** dialog box, click **Start**.
  - To stop an alert rule that is enabled, click **Stop**. In the **Stop** dialog box, click **OK**.
  - To view the history of alert events and alert notifications, click **Alert History**. On the **Alert History** tab, view the related records.

### Query the alert history

Alert events record whether the conditions for triggering an alert are met. The events are collected per minute. Alert events include alert-triggered events that meet the conditions and alert-not-triggered events that do not meet the conditions. On the **Alert History** tab, you can view the history of alert-not-triggered events, alert-triggered events, and alert notifications sent to specified contacts after alerts are triggered.

1. On the **Alarm Policies** page, click the **Alert History** tab.
2. On the **Alert History** tab, specify **Trigger State** and **Alarm Name**, and click **Search**.  
On the **Alert History** tab, the line charts and column charts display the current alert data and alert-triggered events. You can also view the alert details and the relationships between the alert data and alert-triggered events. The line charts represent the alert data and the column charts represent the alert events.
3. Scroll down to the lower part of the page and view the history of alert events on the **Alert Event History** tab.

 **Note** Alert notifications are sent only if the alert rule is in the **Triggered** state. In this scenario, a red dot appears in the **Trigger** column.

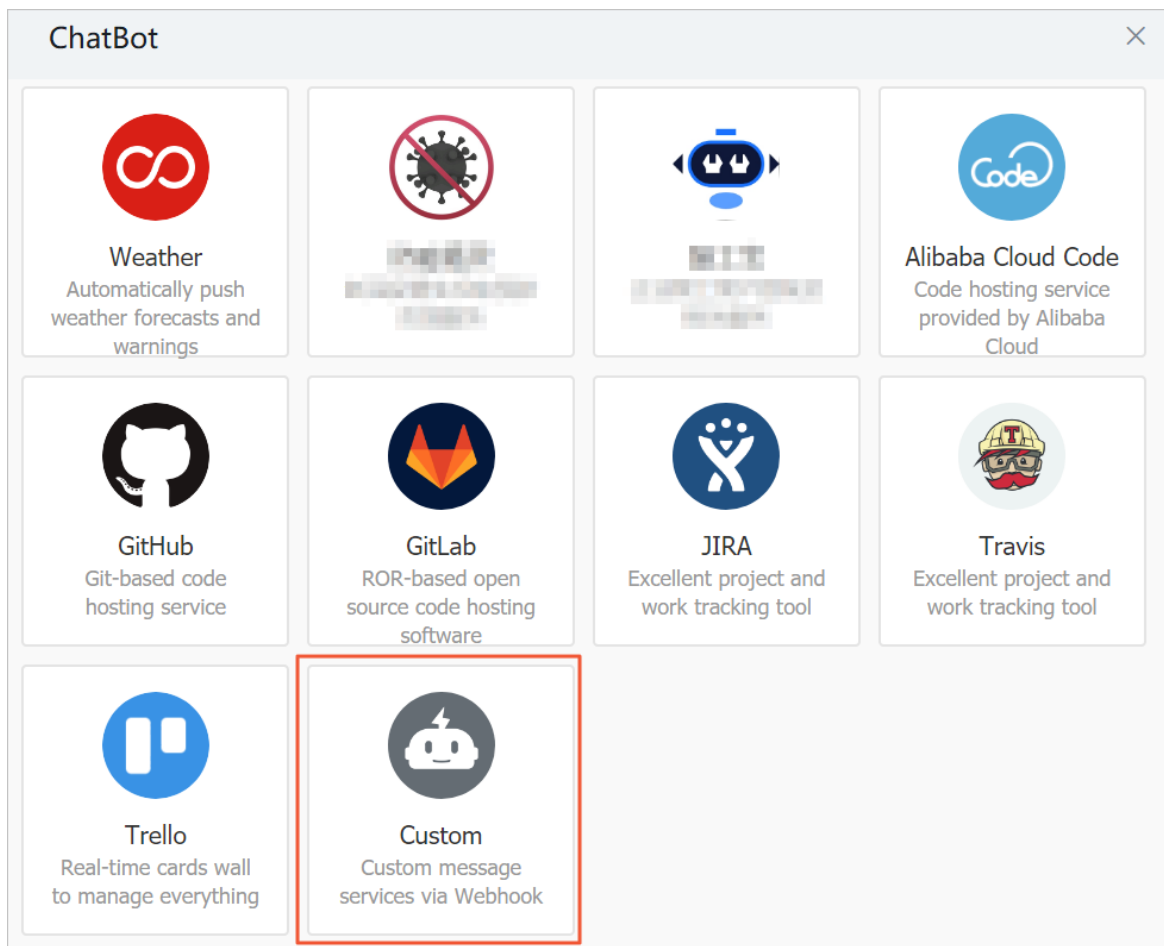
4. Click the **Alarm Post History** tab to view the history of alert notifications that are sent. The alert notifications include text messages and emails.

## 1.6.6.5.5. Configure a DingTalk chatbot to send alert notifications

You can add the webhook URL of a DingTalk chatbot so that alert notifications can be sent to DingTalk groups. This makes your O&M efficient and helps you monitor the alert events at the earliest opportunity.


### Add a custom DingTalk chatbot and obtain its webhook URL

1. Run the DingTalk client on a PC, go to the DingTalk group where you want to add a DingTalk chatbot, and click the Group Settings icon in the upper-right corner.
2. In the **Group Settings** panel, click **Group Assistant**.
3. In the **Group Assistant** panel, click **Add Robot**.
4. In the **ChatBot** dialog box, click the + icon in the **Add Robot** card. Then, click **Custom**.



5. In the **Robot details** dialog box, click **Add**.
6. In the **Add Robot** dialog box, edit the profile picture, enter a name, and then select at least one of the options in the **Security Settings** section. Read and select **I have read and accepted DingTalk Custom Robot Service Terms of Service**. Click **Finished**.

Add Robot



Chatbot name:

Custom

\* Add to Group:

\* Security Settings

☐ Custom Keywords

☐ Additional Signature

☐ IP Address


☐ I have read and accepted [《DingTalk Custom Robot Service Terms of Service》](#)

Cancel

Finished

7. In the **Add Robot** dialog box, copy the webhook URL of the chatbot. Then, click OK.

Add Robot



1. Add robot✓

2. Set up webhook, click setting instruction and check how to make robot effective

Webhook:

Copy

\* Keep Webhook address safe, do not upload to internet for public access.

Use Webhook address to send push message to DingTalk Groupchat

Finished

Setting ins...

## Create a contact


1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Applications**.
3. On the **Applications** page, select **User-built K8s Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to manage.
4. In the left-side navigation pane, choose **Alerts > Contacts**.
5. On the **Contacts** tab, click **Create Contact** in the upper-right corner.
6. In the **Create Contact** dialog box, specify **Name**. In the **Ding Ding Robot** field, enter the obtained webhook URL of the DingTalk chat bot, and click **OK**.

## Create a contact group

1. In the left-side navigation pane, choose **Alerts > Contacts**.
2. On the **Contact Groups** tab, click **New Contact Group** in the upper-right corner.
3. In the **Create Contact Group** dialog box, specify **Group Name**, select the created DingTalk chatbot in **Contact Members**, and then click **OK**.

## Create an alert

1. In the left-side navigation pane, choose **Alerts > Alert Policy Management**.
2. On the **Alarm Policies** page, click **Create Alarm** in the upper-right corner.

3. In the **Create Alarm** dialog box, specify all the required information and click **Save**.
    - i. Specify **Alarm Name**. For example, you can set this parameter to alert on JVM-GC times in period-over-period comparison.
    - ii. Select an application from the **Application Site** drop-down list. Select an application group from the **Application Group** drop-down list.
    - iii. Select a monitoring metric type from the **Type** drop-down list. For example, you can select **JVM\_Monitoring**.
    - iv. Set **Dimension** to **Traverse**.
    - v. Set an alert rule.
      - a. Select **Meet All of the Following Criteria**.
      - b. Edit an alert rule. For example, an alert is triggered when the average value of JVM\_FullGC within the last 5 minutes (N = 5) increases by 100% compared with that in the previous hour.
-  **Note** Click the + icon on the right side of **Last N Minutes** to create multiple alert rules.
- vi. Set **Notification Mode** to **Ding Ding Robot**.
  - vii. In the **Notification Receiver** section, select the contact group that you create in **Create a contact group**. In the **Contact Groups** section, click the name of the contact group. If the contact group appears in the **Selected Groups** section, the setting is successful.

Create Alarm

\*Alarm Name:

\*Application Site:

Application Group:

\*Type:  Dimension:  Non:

\*Alarm Rules: ☒ Meet All of the Following Criteria ☐ Meet Any of the Following Criteria

\*Last N Minutes: N=

\*Notification Mode: ☐ SMS ☐ Email ☐ Ding Ding Robot ☐ Webhook

\*Notification Receiver:

Contact Groups	Selected Groups
USK0S1Z3	jArmsTest
jArmsTest	
jArmsAlarmTest01	
jlttesthserver01	
jiaolongTest2	
jiaolongTest4	

Alert advanced options doc: [?](#)  
[Advanced Configuration](#)

4. Click **Save**.

## 1.6.7. Auto scaling

### 1.6.7.1. Auto scaling (applicable to Kubernetes clusters)

Auto scaling is an important O&M capability for distributed application management. The auto scaling feature provided by EDAS perceives the status of each instance for an application and accordingly implements dynamic scaling. This ensures the quality of service (QoS) and improves application availability.

#### Why is auto scaling used?

Applications such as Internet and game applications are prone to sudden traffic floods during promotional activities. The imbalance between SLA and resource costs causes issues such as system response latency and system breakdown. EDAS inherits the traffic flood management technology used by Alibaba to cope with Double 11. This provides second-level auto scaling to reduce instance retention costs and ensure SLA. Auto scaling is applicable to industries such as the Internet, games, and social networking platforms.

#### Metric-based scaling

EDAS monitors the CPU utilization and the memory usage of your applications, and automatically scales your application instances out or in by using auto scaling policies.

#### Notice

- You can configure only one metric-based scaling policy in a single application.
- When a scaling policy is enabled, do not manage the lifecycle of an EDAS application. Disable the policy and manage the lifecycle.
- You cannot add an auto scaling policy when you perform application changes, such as application deployment, application scaling, and specification changes.

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the page, select a namespace. On the **Applications** page, click the name of the desired application.
3. On the Application Details page, click the **Instance Deployment Information** tab and click the **Auto Scaling** collapse panel. After the collapse panel is expanded, click **Add Flexible Strategy**.
4. On the **Adding a flexible policy** panel on the right side, configure auto scaling rules and click **OK**.
  - **Strategy Name**: the name of the auto scaling policy. The name must start with a lowercase letter and can contain lowercase letters, digits, and hyphens (-). The name must be 1 to 32 characters in length.
  - **Strategy Type**: Only metric-based scaling policies are supported and scheduled scaling policies will be supported.
  - **Triggering Conditions**: supports **CPU Usage** and **Mem Usage**.
  - **Maximum number of application instances**: specifies the specified number of instances that can be scaled out when the auto scaling conditions are triggered.
  - **Minimum number of application instances**: specifies the specified number of instances that can be scaled in when the auto scaling conditions are triggered.

#### Note

- Assume that **CPU Usage** or **Mem Usage** is selected. If the **CPU utilization** or **memory usage** of the current application is at least the preset value, the application is scaled out and the specified number of application instances cannot exceed the value of **Maximum number of application instances**. Otherwise, the application is scaled in and the specified number of the application instances cannot be smaller than the value of **Minimum number of application instances**.
- Assume that both **CPU Usage** and **Mem Usage** are selected. If both values are at least the preset values, the application is scaled out and the specified number of the application instances cannot exceed the value of **Maximum number of application instances**. Otherwise, the application is scaled in and the specified number of the application instances cannot be smaller than the value of **Minimum number of application instances**.

The values of **Maximum number of application instances** and **Minimum number of application instances** are calculated by using the following formula: Specified number of instances = Current number of instances × (Current metric value/Expected metric value)

5. On the right side of **Monitoring indicator strategy list**, click **Enable** in the **Operation** column. If EDAS scales out or in application instances by using the specified auto scaling policy after the policy is triggered, the auto scaling is successful.

6. In the dialog box that appears, specify (Single Pod) CPU core number and (Single Pod) Memory (MB) and click OK.

## Verify the result

After the auto scaling policy is enabled, EDAS automatically scales out or in application instances by using the policy. You can perform the following steps to view the detailed scaling event records.

1. On the right side of **Monitoring indicator strategy list**, click **Event**.
2. On the **Application Event** page, view the detailed scaling events. On this page, you can specify the **Source Type**, **Source Name**, **Cause of Event**, and **Event Level** parameters to search for events and view detailed records.

## References

After an auto scaling policy is enabled, you can delete, disable, enable, and edit the policy and manage applications.

# 1.7. Configuration management

## 1.7.1. Configuration management overview

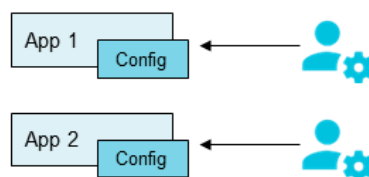
EDAS integrates with Application Configuration Management (ACM). You can perform centralized management on application configurations and push them by using ACM in EDAS.

### What is ACM?

ACM performs centralized management on application configurations and pushes them in a distributed architecture environment. ACM helps you perform centralized management on configurations in all the application environments and reduce the configuration management cost of distributed systems. In addition, it reduces the risk of decreased availability or even failures that are caused by invalid configuration changes.

### Configuration management in traditional architecture

In the traditional architecture, for any configuration changes, you often need to log on the specific server and manually modify the configurations for them to take effect, as shown in the following figure.



### Relationship between ACM and Nacos

Nacos is an open source product of ACM. It is dedicated to provide an easy-to-use dynamic service discovery, configuration and service management platform for building cloud native applications. Nacos provides two major features.

- Distributed configuration center: This feature corresponds to ACM. You can use the Nacos SDK to directly access ACM services.
- Service registration and discovery: This feature corresponds to the registry in EDAS.

## 1.7.2. Manage configurations

You can extract the application data, such as variables and parameters, from code and save them in a profile. This way, when you need to change the configurations for your applications, you need to change only the profile. This topic describes how to create, synchronize, query, edit, and delete configurations.

### Create a configuration

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Application Management > Configuration Management**.
3. In the top navigation bar of the **Configurations** page, select a **region** and a **namespace** and click **Create Configuration**.
4. On the **Create Configuration** page, specify configuration information and click **Create**.

Parameter	Description
Data ID	The ID of the configuration. We recommend that you use the naming convention <i>package.class</i> . In this convention, the class parameter specifies the configuration name that has the business meaning. Example: com.foo.bar.log.level. The data ID must be unique in a group.
Group	The configuration group. We recommend that you enter a product name or a module name. The group must be globally unique.
Configuration Format	The data format of the configuration content.
Configuration Body	Enter the configuration content. Example: <div>threadPoolSize=5 logLevel=WARN</div>
Configuration Description	The configuration description.
More Configurations	<ul style="list-style-type: none"><li>◦ <b>Application</b>: The name of the application to which the configuration belongs.</li><li>◦ <b>Tags</b>: Enter the tag information in the field and click the label selector.</li></ul>


### Synchronize a configuration

You may need to synchronize a configuration for an application in multiple environments, such as the development environment and the test environment. You can create multiple namespaces, create a configuration in one of the namespaces, and then synchronize the configuration to the other namespaces.

## 1.7.3. View historical versions and rollback configurations

### Procedure

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management > Configuration Management**.
3. Perform the following tasks as needed on the page.
  - To view the configuration content of a specific version, click **Details** in the **Actions** column.
  - To roll back to a specific version, click **Roll Back** in the **Actions** column, and on the **Configuration Rollback** page, click **Roll Back**.

 **Note** ACM currently saves change histories of up to 30 days.


## 1.7.4. Query listening

After a configuration is modified, you must check whether the modified configuration information has been pushed to the instances that listen to the configuration. This operation is valid only for clients that use the listening configuration interface to listen to configurations.

### Procedure

1. Log on to the EDAS console.
2. In the left-side navigation pane, choose **Application Management > Configuration Management**.
3. In the left-side navigation pane, choose **Configuration Listening Query**.
4. On the **Configuration Listening Query** page, select a query dimension from the **Query Dimension** drop-down list and enter a group in the **Group** field. Then, enter the data ID of the configuration to be queried in the **Data ID** field and click the search icon.

Listening Query		Microservice Namespace: Default Microservice Namespace	
Query dimension: Config...	Group: DEFAULT_GROUP	Data ID	Please enter Data ID
IP		Push status	
No data			

-  **Note**
- If you set Query Dimension to **Configuration**, the instances to which the configuration is pushed and the push status are queried.
  - If you set Query Dimension to **IP**, all the configurations to which the instance listens are queried.

## 1.8. Microservice governance

## 1.8.1. Overview of microservice governance

Microservice governance is an important function of Enterprise Distributed Application Service (EDAS). In EDAS, you can perform governance on Spring Cloud, Dubbo, and High-speed Service Framework (HSF) microservice-oriented applications, including service search, trace query, outlier instance removal, and service authentication.

### Microservice governance



#### Spring Cloud

Query Spring Cloud services

Ensure Spring Cloud application availability through outlier instance removal

Implement access control of Spring Cloud applications through service authentication

Gracefully disconnect Spring Cloud applications

Publish Spring Cloud applications through canary release

#### Dubbo

Query Dubbo services

Ensure Dubbo application availability through outlier instance removal

Implement access control of Dubbo applications through service authentication

Gracefully disconnect Dubbo applications

Publish Dubbo applications through canary release

Use Dubbo Admin to manage Dubbo services

#### HSF

Query HSF services

Ensure HSF application availability through outlier instance removal

Gracefully release HSF services

View HSF service reports

## 1.8.2. Spring Cloud service governance

## 1.8.2.1. Query Spring Cloud services

You can query the list and details of services of Spring Cloud applications that are deployed in Enterprise Distributed Application Service (EDAS).

### Context

You can switch between the old and new versions of the **Service Search** page.


- In the new version, the system uses EDAS Agent to query services in the EDAS registry, MSE-hosted registry, and on-premises registries, including ZooKeeper, Nacos, Eureka, and Consul.
- In the old version, you can only query services in the EDAS registry.

### Limits

- On the new Service Search page, you can query services of Spring Cloud Edgware and later versions and services in all registries.
- On the old Service Search page, you can query services of Spring Cloud Dalston and later versions that are registered in the EDAS registry.

### View the service list

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Service Query**.
3. On the **Service Query** page, select a **region** and a **Namespaces** to view the **Spring Cloud** services under the current account. The following information about a Spring Cloud service is provided: **Service Name**, **App Name**, and **Number of instances**.  
If many services exist, you can filter services by **Service Name**. Filter keywords are case-insensitive.

 **Note** If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps:

- i. The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.
- ii. Check whether the microservice framework version is supported. For more information about the supported versions, see [Limits](#).

### View service details

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Service Query**.
3. On the **Service Query** page, select a **region** and a **Namespaces**. Click a service name in the service list.
4. On the **Service Detail** page, view the details of the target service.  
The **Service Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.
  - **Basic Information**

<b>Basic information</b>			
Service name	edas.service.consumer	spring.application.name	edas.service.consumer
Service type	Spring Cloud	Application Name	test123

#### Service Call Relationship

<b>Service invocation relationship</b>	
<u>Service Provider (1)</u>	Service Consumer (0)
Please enter IP <input type="text"/> <input type="button" value="Q"/> query results: a total of 1 Results <input type="button" value="C"/>	
IP	Port
192.168.1.1	18080
Items per Page 10 Total 1 < Previous 1 Next >	

The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address** and **port number**.

#### Metadata

<b>Metadata</b>					
Interface metadata					
Category <input type="button" value="v"/>	Please input content <input type="text"/> <input type="button" value="Q"/>				<input type="button" value="C"/>
Category	Http Method	Request Path	Method Name / Description		Params List / Description
com.aliware.Controller	GET	/consumer-echo/feign/{str}	feign2	--	java.lang.String --
com.aliware.Controller	GET	/consumer-echo/{str}	feign1	--	java.lang.String --
com.aliware.Controller	GET	/consumer/alive	alive	--	
com.aliware.Controller	GET	/ping	ping	--	

Information under **Metadata** includes metadata of the service and EDAS-provided metadata for implementing microservice capabilities.

## 1.8.2.2. Query Spring Cloud service traces

You can log on to the EDAS console to query the traces of Spring Cloud services that are deployed in EDAS.

EDAS integrates with ARMS. You can use ARMS to query service traces and holographic events. For more information, see .

## 1.8.2.3. Ensure Spring Cloud application availability

### through outlier instance removal

In a microservice framework, an application is used to provide services and can be deployed on multiple instances. Some instances may become abnormal. If a consumer calls the application in this situation and does not perceive the abnormal instances, the call may fail. This affects service performance and availability. The outlier instance removal feature monitors the availability of instances and dynamically adjusts instances. This ensures successful service calls, and improves service stability and performance.

### Context

The following figure provides a sample scenario where a system includes four applications: Applications A, B, C, and D, and Application A separately calls Applications B, C, and D. Exceptions occur in some instances in Application B, C, or D. Application B has one abnormal instance and Applications C and D each have two abnormal instances, as shown in the red circles of the following figure. If Application A is unable to detect the exception, some calls fail. If a number of abnormal instances exist in Applications B, C, and D, the performance and even the service availability of Application A may be affected.

To ensure the service performance and availability of Application A, configure outlier application removal for Application A. After the configuration is complete, you can monitor the instance status of Applications B, C, and D and dynamically adjust them by adding or removing the instances. This ensures successful service calls.

The following list provides the process of outlier instance removal:

1. When Application B, C, or D has an abnormal instance, the system can detect the instance. In addition, the system can determine whether to remove the abnormal instance from the corresponding application based on the specified value of **Removal instance proportion upper limit**.
2. After the abnormal instance in Application B, C, or D is removed, the call requests of Application A are not allocated to the abnormal instance.
3. EDAS detects whether the abnormal instance is recovered based on the specified value of **Recovery detection unit time**.
4. The detection interval is proportional to the number of detection times and linearly increases based on the value of **Recovery detection unit time**. The default value is 0.5 minutes. When the specified value of **Maximum number of cumulative rollbacks** is reached, EDAS detects whether the abnormal instance is recovered at the maximum interval.
5. When the recovered instance is detected, the instance is added to the instance list of the application to process call requests. In addition, the detection interval is reset to the value of **Recovery detection unit time**, such as 0.5 minutes.


#### Note

- When a large number of abnormal instances exist in provider applications, the instances are removed based on the configured ratio. This indicates that the instances are removed based on the configured ratio when the number of the abnormal instances exceeds the upper limit of the instance removal ratio.
- When only one available instance is left among the provider applications, this instance is not removed even if the error rate exceeds the configured threshold.

## Create an outlier instance removal policy

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Spring Cloud/Dubbo/HSF**, and then click **Outlier Ejection**.
3. On the **Outlier Instance Removal** page, click **Create Strategy**.
4. In the **Create Strategy** wizard, set the parameters on the **Basic Info** page and click **Next**.
  - **Namespace**: Select a region and a namespace from the drop-down lists.
  - **Strategy Name**: Enter a policy name. The name can be up to 64 characters in length.
  - **Framework Used by the Called Service**: Select **Dubbo** or **Spring Cloud** as needed.

5. On the **Select Effect App** page in the **Create Strategy** wizard, select the target application and click **>** to add the application to **Selected application**. Then, click **Next**.  
After the target application is selected, all abnormal application instances that are called by this application are removed. Call requests from the effective application are not sent to the removed instances.
6. In the **Create Strategy** wizard, set the parameters on the **Configuration Strategy** page and click **Next**.
  - **Exception Type**: Select **Network Anomaly** or **Network Anomaly + Business Anomaly (Dubbo Exception)** as needed.
  - **QPS Lower Limit**: Enter a queries per second (QPS) lower limit based on the statistical time window. The time window of applications in Dubbo 2.7 is 15s, and 10s for applications in Dubbo of other versions and Spring Cloud. When the QPS in a statistical time window, for example, 15s, reaches the specified lower limit, Enterprise Distributed Application Service (EDAS) starts collecting and analyzing error rate statistics.
  - **Lower Error Rate**: Set a call error rate threshold. If the error rate for an instance of the called application exceeds this value, the instance will be removed. Default value: 50%. For example, you set this parameter to 50%. An instance is removed if it is called 10 times in the statistical time window, but six calls fail (that is, the error rate is 60%).
  - **Maximum Number of Removed Instances**: Set the maximum number of abnormal instances to be removed. No more abnormal instances will be removed after the threshold is reached. For example, the total number of instances of an application is 6 and this parameter is set to 60%. The number of instances that can be removed is 3.6 ( $6 \times 60\%$ ), which is rounded down to the nearest integer 3. If the calculated result is less than 1, no instance will be removed.
  - **Recovery Detection Unit Time**: Set an interval for detecting whether abnormal instances are recovered, in milliseconds. After abnormal instances are removed, EDAS continuously accumulates the detection interval by the specified time unit. Default value: 30000 ms, that is, 0.5 minute.
  - **Maximum Number of Cumulative Rollbacks**: Set the maximum number of cumulative rollbacks exceeding which the detection interval is no longer increased. For example, you set **Recovery Detection Unit Time** to 30000 ms and **Maximum Number of Cumulative Rollbacks** to 20. If the abnormal instance remains unrecovered after being detected 20 times, the instance is subsequently detected at an interval of 10 minutes ( $20 \times 30000$  ms). If the instance has been recovered before the specified threshold, the detection interval is reset to **Recovery Detection Unit Time**.

 **Note** We recommend that you do not set **Maximum Number of Cumulative Rollbacks** to a large value. A large value will lead to a long detection interval. If the instance is recovered early in the detection interval, the recovery cannot be detected in a timely manner. This results in resource waste and postponed processing of service call requests.

7. In the **Create Strategy** wizard, confirm the settings on the **Complete Creation** page and click **Submit**.

## Verify the result

After you configure and submit an outlier instance removal policy, the outlier instance removal feature is enabled. After you configure an outlier instance removal policy for an application, you can go to the details page of the application to view the monitoring information. You can view the monitoring information in **topology** to check whether all requests are still forwarded to abnormal instances. You can also check whether **Error Rate per Minute** of the application is higher than the configured **Lower Error Rate**. Based on the information, you can determine whether the outlier instance removal policy takes effect.

## 1.8.2.4. Implement access control of Spring Cloud applications based on service authentication

If a microservice application requires high security and you want to restrict the access to it from other applications, you can authenticate applications that call the microservice application. This ensures that only the applications that match the authentication rules can call the microservice application.

### Context

The following example shows how to use service authentication in Spring Cloud.

- Service authentication unconfigured  
Consumers 1, 2, and 3 and a provider belong to the same namespace. By default, Consumers 1, 2, and 3 can call all the paths (Paths 1, 2, and 3) of the provider.
- Service authentication configured
  - Configure authentication for all the paths  
You can configure an authentication rule for all the paths of the provider. For example, configure a blacklist rule to prevent Consumer 1 from accessing all the paths of the provider. In this case, Consumer 2 and Consumer 3 are in the whitelist and can access all the paths.
  - Configure authentication for the specified paths  
After you configure a global authentication rule, you can also configure path-specific authentication rules to restrict access to the specified path from specified consumers. Assume that Path 2 contains core business or data and you do not want Consumer 2 to access this path. After you configure a global authentication rule that allows only Consumer 2 and Consumer 3 to access all the paths of the provider, you can also configure a blacklist rule to prevent Consumer 2 from accessing Path 2. In this case, Consumer 2 can access only Path 1 and Path 3 of the provider.

The following figure shows the application call process after you configure the authentication rules.

### Create a service authentication rule

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Spring Cloud**.
3. In the left-side navigation pane of the **Spring Cloud** page, click **Service Authentication**.
4. On the **Service Authentication** page, click **Create rules**.
5. In the **Create rules** panel, set the service authentication parameters, and click **OK**.

\* Rule name

Uppercase and lowercase letters, numbers, "\_" and "-" are supported, and the length cannot exceed 64 characters.0/64

\* The callee

DemoGTSPProvider

\* Callee framework

☒ Spring Cloud ☐ Dubbo ☐ Service Mesh

+ Add all interface rules ?

All interface rules

Callee interface

All Path

Authentication method \*

☒ Whitelist (allow calls) ☐ Blacklist (call denied)

Caller \*

Select

+Add caller

+ Add specified interface rule ?

Specify interface rule 1

Callee Path \*

Please enter PATH

Switch to custom input

Authentication method \*

☒ Whitelist (allow calls) ☐ Blacklist (call denied)

Caller \*


Select

+Add caller



The callee path data is incomplete

Default state

☒



Service authentication rule parameters:

Parameter	Description
<b>Namespaces</b>	The <b>region</b> and the <b>Namespaces</b> where the service is deployed.
<b>Rule name</b>	The name of the authentication rule. The name can be up to 64 characters in length and can contain uppercase and lowercase letters, digits, underscores (_), and hyphens (-).
<b>Callee framework</b>	The framework that is used by the called application. Select <b>Spring Cloud</b> .
<b>The callee</b>	The called application.
<b>Add all interface rules</b>	
 <b>Notice</b> You can add only one global rule for all interfaces.	
<b>Callee Path</b>	By default, <b>All Path</b> is used and cannot be changed.
<b>Authentication method</b>	The service authentication method. Valid values: <b>Whitelist (allow calls)</b> and <b>Blacklist (call denied)</b> . Select an option based on your authentication requirements.
<b>Caller</b>	The application that requires authentication to call the service. Click <b>Add caller</b> to specify multiple applications.
<b>Add specified interface rule</b>	
 <b>Notice</b> Interface-specific rules are not supplementary. They override the global rule. Exercise caution when you create interface-specific rules.	
<b>Callee Path</b>	The path of the called application.
<b>Authentication method</b>	The service authentication method. Valid values: <b>Whitelist (allow calls)</b> and <b>Blacklist (call denied)</b> . Select an option based on your authentication requirements.
<b>Caller</b>	The application that requires authentication to call the service. Click <b>Add caller</b> to specify multiple applications.

Parameter	Description
Default state	<p>Specifies whether to enable the rule.</p> <ul style="list-style-type: none"><li>On: The rule is enabled immediately after you create it. This is the default value.</li><li>Off: The rule is not enabled after you create it. To enable the rule, find it on the <b>Service Authentication</b> page and click <b>Open</b> in the <b>Operation</b> column.</li></ul>

## Verify the result

After a service authentication rule is configured and enabled, check whether it takes effect based on your business requirements.

## What's next

After you create a service authentication rule, you can **modify**, **disable** (based on the rule status), or **enable** the rule. If a service authentication rule is no longer needed, **delete** the rule.

## 1.8.3. Dubbo service governance

### 1.8.3.1. Query Dubbo services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of Dubbo applications that are deployed in EDAS.

## Context

You can switch between the old and new versions of the **Service Search** page.


- In the new version, the system uses EDAS Agent to query services in the EDAS registry, MSE-hosted registry, and on-premises registries, including ZooKeeper, Nacos, Eureka, and Consul.
- In the old version, you can only query services in the EDAS registry.

## Limits

- On the new Service Search page, you can query services of all Dubbo versions and services in all registries.
- On the old Service Search page, you can only query services of Dubbo 2.7.x that are registered to the EDAS registry through Nacos.

## View the service list

- Log on to the [EDAS console](#).
- In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
- On the **Service Query** page, select a **region** and a **Namespaces** to view the **Dubbo** services under the current account. The following information about Dubbo services is displayed: **Service Name**, **Version**, **Group**, **App Name**, and **Number of instances**.  
If many services exist, you can filter services by **Service Name**, **IP**, or **App Name**. Filter keywords are case-insensitive. When you search for services by **IP**, pay attention to the following:
  - Elastic Compute Service (ECS) cluster: Enter the IP address of the ECS instance.
  - Container Service Kubernetes cluster: Enter the IP address of a pod.

-  **Note** If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps:
- i. The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.
  - ii. Check whether the microservice framework version is supported. For more information about the supported versions, see [Limits](#).

## View service details

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Service Query**.
3. On the **Service Query** page, select a **region** and a **Namespaces**. Click a service name in the service list.
4. On the **Service Detail** page, view the details of the target service.  
The **Service Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.
  - **Basic Information**
  - **Service Call Relationship**  
The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address**, **port number**, **serialization method**, and **timeout period**.
  - **Metadata**  
The Metadata section provides **Metadata** and **Interface Metadata**.
    - Information under **Metadata** includes metadata of the service and EDAS-provided metadata for implementing microservice capabilities.
    - **Interface Metadata** includes **Method Name**, **Parameters**, and **Return Type**.

### 1.8.3.2. Query Dubbo service traces

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the traces of Dubbo service that are deployed in EDAS.

EDAS is integrated with Application Real-Time Monitoring Service (ARMS). You can use ARMS to query service traces and holographic troubleshooting events.

### 1.8.3.3. Ensure Dubbo application availability through out-of-service instance removal

In a microservice framework, service calls are affected if consumers cannot detect the exceptions on the application instances of a provider. This further affects the performance and even availability of the services provided by the consumers. The out-of-service instance removal feature monitors the availability of application instances and dynamically adjusts the instances. This ensures successful service calls and improves the service stability and quality of service (QoS).

## Context

The following figure provides a sample scenario where a system includes four applications: Applications A, B, C, and D. In this scenario, Application A respectively calls Applications B, C, and D. Exceptions occur to the instances in red dashed circles. Application B has one abnormal instance. Application C has two abnormal instances and Application D also has two abnormal instances. If Application A cannot detect the exceptions and continues to call an application that has exceptions, some calls may fail. If a large number of abnormal instances exist in Applications B, C, and D, Application A may encounter performance degradation or even service unavailability.

To ensure the service performance and availability of Application A, configure out-of-service instance removal for it. After the configuration is complete, you can monitor the instance status of Applications B, C, and D and dynamically add or remove the instances to ensure successful service calls.

The following content describes the procedure of out-of-service instance removal:

1. When an exception occurs to an instance of Applications B, C, or D, the system can detect the exception. Then, the system removes the abnormal instance from the corresponding application if the number of removed instances has not exceeded the value of the **Maximum number of removed instances** parameter.
2. After the abnormal instance in Applications B, C, or D is removed, the call requests of Application A are not allocated to this abnormal instance.
3. EDAS detects whether the abnormal instance is recovered based on the specified value of **Recovery detection unit time**.
4. The detection interval is proportional to the detection times and linearly increases based on the **recovery detection unit time**, whose default value is 0.5 minutes. When the **maximum cumulative number of times not restored** that you set is reached, EDAS detects whether the abnormal instance is recovered at the maximum interval.
5. When the instance is recovered, it is added to the instance list of the application to process call requests. The detection interval is reset to the **recovery detection unit time**, for example, 0.5 minutes.

#### Note

- If the number of abnormal instances in the provider applications exceeds the value of the **Maximum number of removed instances** parameter, the instances are removed based on the specified ratio.
- When only one instance is available in the provider applications, this instance is not removed even if the error rate exceeds the configured threshold.


## Create an outlier instance removal policy

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Spring Cloud/Dubbo/HSF**, and then click **Outlier Ejection**.
3. On the **Outlier Instance Removal** page, click **Create Strategy**.
4. In the **Create Strategy** wizard, set the parameters on the **Basic Info** page and click **Next**.
  - **Namespace**: Select a region and a namespace from the drop-down lists.
  - **Strategy Name**: Enter a policy name. The name can be up to 64 characters in length.
  - **Framework Used by the Called Service**: Select **Dubbo** or **Spring Cloud** as needed.
5. On the **Select Effect App** page in the **Create Strategy** wizard, select the target application

and click > to add the application to Selected application. Then, click **Next**.

After the target application is selected, all abnormal application instances that are called by this application are removed. Call requests from the effective application are not sent to the removed instances.

6. In the **Create Strategy** wizard, set the parameters on the **Configuration Strategy** page and click **Next**.
  - o **Exception Type**: Select **Network Anomaly** or **Network Anomaly + Business Anomaly (Dubbo Exception)** as needed.
  - o **QPS Lower Limit**: Enter a queries per second (QPS) lower limit based on the statistical time window. The time window of applications in Dubbo 2.7 is 15s, and 10s for applications in Dubbo of other versions and Spring Cloud. When the QPS in a statistical time window, for example, 15s, reaches the specified lower limit, Enterprise Distributed Application Service (EDAS) starts collecting and analyzing error rate statistics.
  - o **Lower Error Rate**: Set a call error rate threshold. If the error rate for an instance of the called application exceeds this value, the instance will be removed. Default value: 50%. For example, you set this parameter to 50%. An instance is removed if it is called 10 times in the statistical time window, but six calls fail (that is, the error rate is 60%).
  - o **Maximum Number of Removed Instances**: Set the maximum number of abnormal instances to be removed. No more abnormal instances will be removed after the threshold is reached. For example, the total number of instances of an application is 6 and this parameter is set to 60%. The number of instances that can be removed is 3.6 ( $6 \times 60\%$ ), which is rounded down to the nearest integer 3. If the calculated result is less than 1, no instance will be removed.
  - o **Recovery Detection Unit Time**: Set an interval for detecting whether abnormal instances are recovered, in milliseconds. After abnormal instances are removed, EDAS continuously accumulates the detection interval by the specified time unit. Default value: 30000 ms, that is, 0.5 minute.
  - o **Maximum Number of Cumulative Rollbacks**: Set the maximum number of cumulative rollbacks exceeding which the detection interval is no longer increased. For example, you set **Recovery Detection Unit Time** to 30000 ms and **Maximum Number of Cumulative Rollbacks** to 20. If the abnormal instance remains unrecovered after being detected 20 times, the instance is subsequently detected at an interval of 10 minutes ( $20 \times 30000$  ms). If the instance has been recovered before the specified threshold, the detection interval is reset to **Recovery Detection Unit Time**.

 **Note** We recommend that you do not set **Maximum Number of Cumulative Rollbacks** to a large value. A large value will lead to a long detection interval. If the instance is recovered early in the detection interval, the recovery cannot be detected in a timely manner. This results in resource waste and postponed processing of service call requests.

7. In the **Create Strategy** wizard, confirm the settings on the **Complete Creation** page and click **Submit**.

## Verify results

The out-of-service instance removal feature is enabled after you configure an out-of-service instance removal policy. You can go to the details page of the application for which you have configured out-of-service instance removal to view the application monitoring information. For example, you can check whether call requests are still forwarded to abnormal instances and whether the **error rate per minute** for application calls is higher than the configured **lower error rate limit** through a **topology**. This way, you can check whether the out-of-service instance removal policy takes effect.

### 1.8.3.4. Implement access control of Dubbo applications through service authentication

If you do not want specific applications to call your microservice application, you can configure rules to authenticate applications. Only the applications that match the authentication rules are allowed to call your application.

#### Context

This topic uses an example to introduce scenarios where Dubbo service authentication is performed.

Consumers 1, 2, and 3 and a service provider are deployed in the same namespace. By default, Consumers 1, 2, and 3 can call all the services and interfaces of the provider.

You can specify an authentication method for all the services and interfaces of the provider. For example, set the authentication method to Blacklist (call denied) for Consumer 1 and set the authentication method to Whitelist (allow calls) for Consumer 2 and Consumer 3.

Then, you can also set an authentication method for specified services and interfaces of the provider. For example, after you apply the preceding settings, Consumer 2 and Consumer 3 can access all services and interfaces of the provider. However, Service and Interface 2 of the provider involves core business and data. To disable Consumer 2 from accessing Service and Interface 2, set the authentication method of Service and Interface 2 to Blacklist (call denied) for Consumer 2. This way, Consumer 2 can access only Service and Interface 1 and Service and Interface 3 of the provider.

The following figure shows the application call process after you configure the authentication rules.

#### Create a service authentication rule

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Dubbo**.
3. In the left-side navigation pane of **Dubbo**, click **Service Authentication**.
4. On the **Service Authentication** page, click **Create rules**.
5. On the **Create rules** page, set service authentication parameters, and click **OK**.

← Create rules

\* Namespace

China East 1 (Hangzhou) test

\* Rule name

Uppercase and lowercase letters, numbers, "\_" and "-" are supported, and the length cannot exceed 64 characters. 0/64

\* The callee

Please select callee

☒ Spring Cloud ☐ Dubbo ☐ Service Mesh

+ Add all interface rules ?

+ Add specified interface rule ?

Please add all interface rules or specify interface rules


Default state


☒

OK

Cancel

Service authentication rule parameters:

Parameter	Description
Namespaces	The <b>region</b> and the <b>Namespaces</b> where the service is deployed.
Rule name	The name of the authentication rule. The name must be up to 64 characters in length, and can contain uppercase or lowercase letters, digits, underscores (_), and hyphens (-).
Callee framework	The framework of the called application. Select <b>Dubbo</b> .
The callee	The called application.
<b>Add all interface rules</b>	
 <b>Notice</b> You can add a common rule for all interfaces only once.	
Callee interface	By default, the value is <b>All services/all interfaces</b> and cannot be changed.

Parameter	Description
Authentication method	The service authentication method. Values: <b>Whitelist (allow calls)</b> and <b>Blacklist (call denied)</b> . Select an option as needed.
Caller	The caller application to be authenticated for calling the service. Click <b>Add caller</b> to select multiple applications.
<b>Add specified interface rule</b> <div>  <b>Notice</b> The rules added to the specified interface overwrite the common rules that apply to all interfaces. Proceed with caution.         </div>	
Callee interface	Specify the services and interfaces of the called application.
Authentication method	The service authentication method. Values: <b>Whitelist (allow calls)</b> and <b>Blacklist (call denied)</b> . Select an option as needed.
Caller	The caller application to be authenticated for calling the service. Click <b>Add caller</b> to select multiple applications.
Default state	Specifies whether to enable the rule. <ul style="list-style-type: none"> <li>On: enables the rule immediately after it is created. This is the default value.</li> <li>Off: disables the rule after it is created. To enable the rule, find it on the <b>Service Authentication</b> page and click <b>Enable</b> in the <b>Operation</b> column.</li> </ul>

## Verify the result

After the service authentication rule is created and enabled, check whether the rule takes effect.

## What's next

After you create a service authentication rule, you can click **Edit**, **Close**, or **Open** in the **Operation** column to manage the rule. If the service authentication rule is no longer required, you can click **Delete** to delete the rule.

## 1.8.4. HSF service governance

### 1.8.4.1. Query HSF services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of High-speed Service Framework (HSF) applications that are deployed in EDAS.


## View the service list

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
3. On the **Service Query** page, select a **region** and a **Namespaces** to view the HSF services under the current account. The following information of HSF services is displayed: **Service Name**,

### Version, Group, App Name, and Number of Instances.

If many services exist, you can filter services by **Service Name**, **IP**, or **App Name**. Filter keywords are case-insensitive. When you search for services by **IP**, pay attention to the following:

- Elastic Compute Service (ECS) cluster: Enter the IP address of the ECS instance.
- Container Service Kubernetes cluster: Enter the IP address of a pod.

 **Note** If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps: The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.

## View service details

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
3. On the **Service Query** page, select a **region** and a **Namespaces**. Click the **Spring Cloud**, **Dubbo**, or **HSF** tab. Click a specific service name in the service list.
4. On the **Service Detail** page, view the details of the target service.  
The **Service Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.

- **Basic Information**

Basic information			
Service name	com.alibaba.edas.testcase.api.M...	Version	2.0.0
Grouping	0121	Service type	HSF
Application Name	test-mesh-2		

- **Service Call Relationship**

The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address**, **port number**, **serialization method**, and **timeout period**.

- **Metadata**

**Interface Metadata** includes **Method Name**, **Parameters**, and **Return Type**.

## 1.8.4.2. Query HSF service traces

You can log on to the EDAS console to query the traces of HSF services that are deployed in EDAS. EDAS is integrated with Application Real-Time Monitoring Service (ARMS). You can use ARMS to query service traces and holographic troubleshooting events.

## 1.8.4.3. View HSF service reports

Service statistics show the runtime status of all the services of all the applications under the current tenant from the past 24 hours, including the service call volume, call time consumption, and number of call errors. These statistics allow you to easily compare all services in the system.

### Procedure

## Procedure

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Microservice Governance > Service Statistics**.
3. On the **Service Statistics** page, view the runtime data of services.

## 1.8.4.4. End-to-end canary release

### 1.8.4.4.1. Overview

Canary release help developers upgrade applications to a later version more smoothly. Enterprise Distributed Application Service (EDAS) supports canary traffic on a single application and end-to-end canary release for multiple applications.

### Scenario

You can implement single-application and multi-application canary release based on HTTP or HSF.

- **Upgrade of a single application**  
New versions are released continuously during application iteration. Before releasing a new version officially, you can use the canary traffic feature to verify the new version's function, performance, and stability on a small scale, and then perform a full upgrade if the verification succeeds.
- **Multi-application troubleshooting**  
When your HSF microservice-oriented application deployed in EDAS fails, you can use the end-to-end canary traffic feature to route the specified traffic to an application for troubleshooting. This ensures the normal operation of the entire microservice-oriented application.

### Terms

#### Ingress application and canary release rule

To perform end-to-end canary release, you must specify an ingress application and set the HTTP and HSF canary release rules separately. The following figure describes the functions of the ingress application and canary release rules:

- For HTTP traffic, the canary release traffic is identified only after it reaches the ingress application. If the traffic meets the rule, it is marked as the canary release traffic.

#### Notice

- Currently, end-to-end canary release does not support routing of HTTP canary release traffic. The system identifies canary traffic based on HTTP traffic at the ingress application and then routes the HSF canary traffic generated by other nodes to the canary instance group.
  - Based on the throttling rules of a single application, the system not only identifies the HTTP canary release traffic but also routes it to application groups.
- As for the HSF traffic, the canary release traffic is identified and routed before it reaches the ingress application. If the ingress application has an instance in the current canary instance group, the canary release traffic is directly routed to the canary instance group. Otherwise, the canary traffic is routed to the default group of this ingress application. In simple terms, the system identifies the canary release traffic and routes it.  
The method for setting canary release rules is similar to that for throttling of a single application, except that you can set multiple rules.

- You must specify the traffic protocol type for each rule, which can be HTTP or HSF.
- Each rule can have multiple rule conditions that are in the AND or OR relationship.

## Canary release environment

EDAS manages canary releases by defining a canary release environment. A canary release environment contains the ingress application and canary release rules. It is also a logical space that contains application instance groups. Therefore, you can add or remove an instance group (a non-default group) of an application to or from a canary release environment.

## Flexible features

The EDAS end-to-end canary release solution implements canary release and throttling through the console, and has the following flexible features:

- You only need to prepare instance resources for the applications that require canary release, instead of building an entire service system.
- It allows you to enable canary release for multiple applications and set different canary release throttling rules for different applications, and even allows one application to be involved in multiple canary release throttling operations at the same time.
- It supports link-based canary release. That is, it allows multiple applications to be in the same canary environment. The canary traffic identified by an upstream application instance can still pass through immediate application instances that do not require canary release, and then be routed to a downstream canary application instance.

### 1.8.4.4.2. Update a single application by using end-to-end traffic adjustment

In the application iteration process, you can use end-to-end canary traffic adjustment to verify a new version on a small number of instances. After the verification succeeds, you can upgrade all of your applications to the new version.

#### Scenario description

Web Application A has two instances that are of the V1 version and are deployed in an ECS cluster through WAR packages. After the development of the V2 version is complete, you need to first verify the version in one instance. After the V2 version passes the verification, you can upgrade the other instance from V1 to V2 to complete the upgrade of Application A.

#### Canary release process

1. Create a canary instance group.
2. In the canary instance group, configure and enable a traffic adjustment rule.
3. Deploy the new version (V2) in the canary instance group. Then, verify that the specified traffic is distributed to instances in the canary instance group.
4. Verify the new version based on the traffic that is distributed to the canary instance group.
5. After the new version passes the verification, upgrade the version of the application in the default group to V2.  
If problems are found during the verification, disable the traffic adjustment rule for the canary instance group and move the instance in the canary instance group to the default group. After the upgrade to the V2 version is complete, enable the traffic adjustment rule for the canary instance group again, and deploy the application and verify the canary traffic in the canary instance group.

6. Disable the traffic adjustment rule for the canary instance group and delete the canary instance group.

## Step 1: Create a canary instance group

In ECS clusters, different application versions are deployed and traffic adjustment rules are configured based on instance groups. Therefore, you must create a canary instance group first.

1. Log on to the [EDAS console](#).
2. In the left-side navigation pane, choose **Application Management > Applications**. On the **Applications** page, click the name of a specific application.
3. On the Application Details page, click the **Instance Deployment Information** tab. On the page that appears, click **Create Group** in the upper-right corner.
4. In the **Create Group** dialog box, enter a **canary instance group** in the *Group Name* field and click **Create**.

After the group is created, the message **The group is created** appears on the top of the page.

## Step 2: Configure and enable a traffic adjustment rule

You can configure rules for HSF traffic adjustment and HTTP traffic adjustment.

- To configure a rule for HTTP traffic adjustment, enable **Traffic Management** on the **Basic Information** tab.
- To configure a rule for HSF traffic adjustment, you do not need to enable **Traffic Management**. However, your application must use EDAS Container 3.5.3 and later.

The following section provides an example on how to configure and enable a traffic adjustment rule. A rule for HTTP traffic adjustment is used in the example.

1. In the **Application Settings** section on the **Basic Information** tab, click **Enable** on the right side of **Traffic Management**.
2. On the **Instance Deployment Information** tab, click **Traffic Adjustment** in the upper-right corner of the section for the canary instance group, and select **HTTP Traffic Adjustment** from the drop-down list.

The screenshot displays the EDAS console interface. At the top, there are two tabs: 'Basic Information' and 'Instance Information'. The 'Instance Information' tab is active, and a red circle '1' highlights the 'Create Group' button in the top right corner. Below the tabs, there is a section for 'Default Group' with details about the deployment package version and running instances. A table lists instance details, including Instance ID/Name, IP, Specifications, Network Type, Package Version/MD5, Running Status, and Actions. A red circle '2' highlights the 'Traffic Adjustment' button in the top right corner of the instance group section. A dropdown menu is open, showing 'Canary Environment Settings', 'HTTP Traffic Adjustment', and 'HSF Traffic Adjustment'. A red circle '3' highlights the 'HTTP Traffic Adjustment' option. Below this, another section for 'Canary Environment Settings' is visible, showing details about the canary environment and its traffic adjustment rules. A table lists the details of the canary environment, including Instance ID/Name, IP, Specifications, Network Type, Package Version/MD5, Running Status, and Actions.

3. In the **Traffic Adjustment** dialog box, specify the traffic adjustment parameters and select **Enable** the following configuration to control HTTP request traffic that enters the current application instance group. Then, click **Save**.  
You can select one of the following two canary release rules for inbound traffic: **Canary Release by Content** and **Canary Release by Ratio**.

- **Canary Release by Content**: distributes traffic that meets the configured canary release rule to the canary instances.
- **Canary Release by Ratio**: randomly distributes traffic to the canary instances based on the ratio.

The canary release by ratio method is easy to use. This topic describes how to specify a traffic adjustment rule by using the canary release by content method.

The screenshot shows the 'Traffic Adjustment' dialog box. At the top, there is a checkbox 'Enable the following configuration to control HTTP request traffic that enters the current application instance group'. Below this, there are two tabs: 'Canary Release by Content' (selected) and 'Canary Release by Ratio'. The 'Canary Release by Content' tab contains a 'Path' field with a placeholder text 'A relative HTTP path, for example, /a/b. Note that the paths must be matched exactly. If the value is left empty, any'. Below the path field, there are two radio buttons for 'Conditional Mode': 'Meet All Following Conditions' (selected) and 'Meet Any of Following Conditions'. Below the conditional mode, there is a table for 'Conditions' with columns 'Parameter Type', 'Parameter', 'Conditions', and 'Value'. The table is currently empty and shows 'No Data'. Below the table is a '+ Add Rule Condition' button. At the bottom right of the dialog are 'Save' and 'Cancel' buttons. To the right of the dialog, there is a diagram titled 'Canary Release by Content' showing 'Upstream App A' sending traffic to 'App B (new) V2 V2' and 'App B V1 V1 V1' based on the condition 'uid % 100 <= 40'.

Parameter description for traffic adjustment:

- **Conditional Mode**: Select **Meet All Following Conditions**.
- **Conditions**: The following three methods are available: Cookie, Header, and Parameter. In this topic, Parameter is used as an example.
  - **Parameters Type**: Select **Parameter**.
  - **Parameters**: Enter a *version*.
  - **Conditions**: Select **=**.
  - **Value**: Enter **1**.

**Notice** After the parameter settings are complete, select **Enable the following configuration to control HTTP request traffic that enters the current application instance group**. Then, the traffic adjustment rule takes effect.

If the canary instance group has no instance to receive the canary traffic after the canary release rule takes effect, the default group is automatically used.

### Step 3: Deploy a new version and verify traffic adjustment

1. Move the instance to the canary instance group.
  - i. On the **Instance Deployment Information** tab, click **Change Group** in the Actions column of the specified instance in the Default Group section.
  - ii. In the **Change Group** dialog box, select a **canary instance group** from the **Target Group** drop-down list and click **OK**.  
After the instance group is moved, the current V1 version is used on the instance by default.
2. Deploy a new version

You can deploy an application in the console or by using tools. In this example, the application is deployed in the console. If you need to use other deployment methods, see [Deploy an application \(applicable to ECS clusters\)](#).

- i. On the Application Details page, click **Deploy Application** in the upper-right corner.
- ii. On the page for **configuring the application to be deployed**, specify the deployment parameters and click **OK**.

**Note:** In this topic, the V1 version is deployed for the sample application by using the WAR package. Therefore, you must select the WAR package for redeployment. If you have deployed the application by using JAR packages, select the options that are related to the **JAR package**.

- **Deployment Method:** Select **WAR**.
- **File Uploading Method:** Select **Upload WAR Package** and click **Select File**. In the dialog box that appears, select the local WAR package of the V2 version.
- **Version:** Enter **V2**.
- **Group:** Select a **canary instance group**.
- **Batches per Group:** Select **1 Batches**.
- **Batch Mode:** Select **Automatic**.

After the application is deployed, the Change Details page appears in the console. You can view the deployment progress on this page. When **Success** appears in the **Change Status** column for the application, the application is deployed. If the deployment fails, the related log appears in the change details. You can check the log for troubleshooting.

Return to the Application Details page and click the **Instance Deployment Information** tab. On the Instance Deployment Information tab, the value of **Deployment Package Version** is updated to **V2** and **Running Status** is updated to **Normal**.

### 3. Verify traffic adjustment

- i. In the address bar of the browser, enter `http://<IP address of the instance in the default group>:<Service port>` and press Enter.  
The web page of the application that is deployed in the V1 version appears.
- ii. In the address bar of the browser, enter `http://<IP address of the instance in the default group>:<Service port>?version=1` and press Enter.  
The web page of the application that is deployed in the V2 version appears.

This indicates that the traffic adjustment rule has taken effect and the specified traffic is distributed to the instance in the canary instance group.

## Step 4: Verify the new version

You can verify the new version based on your actual business requirements.

If problems are found during the verification, disable the traffic adjustment rule for the canary instance group and move the instance in the canary instance group to the default group. After the upgrade to the V2 version is complete, enable the traffic adjustment rule for the canary instance group again, and deploy the application and verify the canary traffic in the canary instance group.

## Step 5: Upgrade the version of the application in the default group

After the new V2 version passes verification, you need to upgrade the application in the default group to V2.

If you upgrade the application, it indicates that the application is deployed again. For more information about the procedure, see [Step 3: Deploy a new version and verify traffic adjustment](#).

## Step 6: Disable the traffic adjustment rule and delete the canary instance group

After you upgrade the two instances in both groups to the V2 version, you must disable the traffic adjustment rule and release the canary instance group.

1. On the **Instance Deployment Information** tab, click **Traffic Adjustment** in the upper-right corner of the canary instance group section, and select **HTTP Traffic Adjustment** from the drop-down list.
2. In the **Traffic Adjustment** dialog box, clear **Enable the following configuration to control HTTP request traffic that enters the current application instance group** and click **Save**.
3. Return to the **Instance Deployment Information** tab, and click **Change Group** in the Actions column of the instance in the canary instance group.
4. In the **Change Group** dialog box, select **Default Group** from the **Target Group** drop-down list and click **OK**.
5. Return to the **Instance Deployment Information** tab, and click **Delete Group** in the upper-right corner of the canary instance group section.

### 1.8.4.4.3. Troubleshoot application problems by using end-to-end traffic adjustment

When your HSF microservice application that is deployed in EDAS fails, you can troubleshoot specific process problems for the application by using end-to-end traffic adjustment. This makes the troubleshooting more efficient and ensures that the entire microservice application is running as expected. This topic uses an example to describe how to troubleshoot process problems for applications by using end-to-end traffic adjustment.

#### Limits

You can use canary release to troubleshoot multiple applications. This method has the following limits:

- The cluster type must be ECS clusters.
- The applications must run in EDAS Container. This indicates that the main applications are HSF applications.
- Applications deployed by using images are not supported.

#### Sample scenario description

Applications A, B, C, and D are deployed in the HSF microservice application process and A1, B1, C1, and D1 versions are separately deployed for the four applications. An exception occurs for a service that is provided in the process. The preliminary troubleshooting indicates that errors exist in Applications B and D. The specified traffic is routed to the canary instance groups of Applications B and D through end-to-end traffic adjustment so that problems can be troubleshooted.

#### Troubleshooting process

The troubleshooting process is divided into two phases. In the first phase, the system troubleshoots Application D. In the second phase, the system troubleshoots Application B.


#### Procedure

The following section describes the troubleshooting procedure in details.

1. Create an instance group for Application D to receive the canary traffic.

2. Create a traffic adjustment environment. Specify Web Application A as an entry application and configure a traffic adjustment rule for the application. Add a traffic adjustment group for Application D and enable a canary release environment.

The test traffic can be identified on the HTTP entry application. This method is convenient and common. In this example, the HTTP entry application is Application A, but the application to be troubleshooted is Application D. The end-to-end traffic adjustment capability is required because the canary traffic needs to be identified and processed on different applications.

 **Note** When no instances are deployed in the canary instance group, the group cannot receive the canary traffic. In this case, canary degradation is triggered so that instances in the default group can receive the canary traffic. After an instance is added to the canary instance group, canary degradation is disabled, and the canary traffic flows to the instances in the canary instance group.

3. Add an instance to the canary instance group of Application D.  
Add the instance based on the version of the instance in the default group.
4. Verify whether the traffic distribution meets your expectation. Troubleshoot the problem in the canary instance group of Application D.
  - i. You can click **Flow Monitoring** next to the canary instance group to determine whether the canary release rule takes effect and whether the traffic distribution meets your expectation.
  - ii. Troubleshoot the problem in the canary instance group of Application D by using the canary traffic.
    - If the troubleshooting process is smooth, you can prepare to troubleshoot Application B by creating a canary instance group for Application B.  
The canary instance group of Application D is created after the traffic adjustment environment is created and the traffic adjustment rule is configured. You can activate the traffic adjustment rule and deploy canary instances for applications by using multiple policies because the order of these two operations is not limited.
    - If the configuration is invalid or you need to update the version, disable the canary release environment and remove instances from the canary instance group of Application D. After the new version is available or the configuration is modified, enable the canary release environment and redeploy the application for verification.
5. Create an instance group for Application B to receive the canary traffic.
6. Add a canary instance group of Application B to the traffic adjustment environment.
7. Add an instance to the canary instance group of Application B.
8. Check whether the traffic distribution meets your expectation and troubleshoot the problem in the canary instance groups of Applications B and D.
  - i. You can click **Flow Monitoring** next to the canary instance group to determine whether the canary release rule takes effect and whether the traffic distribution meets your expectation.

- ii. Troubleshoot the problem in the canary instance groups of Applications B and D by using the canary traffic.
  - If the troubleshooting process is complete, disable the canary release environment, delete the canary instance groups of Applications B and D, and then delete the traffic adjustment environment.
  - If the configuration is invalid or you need to update the version, update the application or the configuration. After the application is redeployed, verify whether the traffic distribution meets your expectation again.

The traffic adjustment environment is not disabled because Application D is being troubleshot. Only the instance is removed from the canary instance group of Application B.

## 1.8.4.4.4. Monitor traffic for an application in a canary release

After a canary release is implemented, you can monitor traffic for applications and instances. This helps you verify that the canary release is successful.

### Context

You can monitor traffic for a single application in a canary release. If multiple applications are involved in canary releases and end-to-end traffic adjustment is enabled, you can also monitor traffic for multiple applications.

You can implement a canary release for a single application by application instance group. After a canary release is implemented, you can monitor traffic for the application and application instances by application instance group. You can perform the following steps:

### Procedure

1. Log on to the Enterprise Distributed Application Service (EDAS) console. In the left-side navigation pane, choose **Application Management > Applications**.
2. On the **Applications** page, click the name of the application for which a canary release is implemented.
3. On the application details page, click the **Instance Deployment Information** tab. On the right side of the application instance group that you want to manage, click **Traffic Monitoring**.
4. In the **Flow Monitoring** dialog box, select **Instance Perspective** or **Service Perspective**, and select the start time and end time for traffic monitoring in the time window.
5. Monitor traffic for the application.
  - Instance Perspective: On the **Overview** tab, you can monitor the application and application instances, such as the upstream and downstream traffic, response time (RT), number of requests, and number of errors. You can select an application or instance in the list on the left side for monitoring. You can also monitor other common items, such as JVM monitoring, host monitoring, and interface snapshots.
  - Service Perspective: On the **Overview** tab, you can monitor a service provided by the application, such as the upstream and downstream traffic, RT, number of requests, and number of errors. You can select a service in the list on the left side for monitoring. You can also monitor other common items, such as interface snapshots.

### Result

Monitor application traffic when end-to-end traffic adjustment is enabled


Assume that multiple applications are involved in canary releases and end-to-end traffic adjustment is enabled. In this case, a canary release environment is provided for the applications. After canary releases are complete, you can monitor traffic for each application in the canary release environment. You can perform the following steps:

1. Log on to the EDAS console. In the left-side navigation pane, choose **Microservices Governance > HSF > End-to-end Traffic Adjustment**.
2. On the **End-to-end Traffic Adjustment** page, select a namespace and click the name of the canary release environment.
3. On the details page of the canary release environment, click the **Monitoring Details** tab.
4. On the **Monitoring Details** tab, select an application from the drop-down list next to **Gray Environment Application**, and select **Instance Perspective** or **Service Perspective**. Then, select the start time and end time for traffic monitoring in the time window.
5. Monitor traffic for the specified application in the canary release environment.  
You can monitor traffic for each application in the canary release environment.

### 1.8.4.4.5. Restrictions on canary release

End-to-end canary release provides flexible canary release methods and is also subject to some restrictions and conventions.

If an application instance group belongs to multiple canary environments, throttling conflicts may occur. Therefore, in the end-to-end canary release, one application instance group belongs to only one canary environment.

 **Note** After a High-speed Service Framework (HSF) traffic rule is enabled for an application group during the throttling of an application, a canary environment is also created for this application group.

#### Unique canary property for traffic

After a traffic request is marked as belonging to a certain canary environment, this request will never be marked as belonging to another canary environment even if this request complies with other canary release rules.

#### Priority of canary release rules

One application may be used as the entry application for multiple canary environments, and traffic may comply with multiple canary release rules at the same time. Due to the unique canary property for traffic, you must set priority policies for multiple canary release rules. Currently, a canary release rule that is created or modified later takes effect earlier.

#### Restrictions on the joint use of single application throttling and end-to-end canary release

An application may use both single application throttling and end-to-end canary release. As described in [Overview of end-to-end traffic adjustment](#), instance group C1 of application C sets a throttling rule, whereas instance group C2 participates in the end-to-end canary release that involve multiple applications.

When an HSF throttling rule is set in single application throttling, a canary environment is created. However, an application group can belong to only one canary environment. Therefore, when an application group, such as C2, has been added to an end-to-end canary environment, HSF throttling rules cannot be set for single application throttling. Similarly, when an application group, such as C1, has its own HSF throttling rules, it cannot be added to another end-to-end canary environment.

### Unique rule for the same ingress endpoint of an entry application in the single application throttling mode

In the single application throttling mode, the HTTP or HSF throttling rules of an application are subject to the uniqueness restriction. That is, one ingress endpoint of an application can be defined by only one HTTP or HSF throttling rule. The ingress endpoint is defined as follows:

- For the HTTP protocol, an endpoint refers to an application, and only one group can be set for an application.
- For the HSF protocol, an endpoint refers to a method in an interface.

In single application throttling, only one HTTP throttling rule can be customized for one application. Also, an interface method of an application can be used in only one HSF throttling rule.

### 1.8.4.4.6. Canary release policies

In canary release, you can first set throttling rules and then deploy a new version, or first deploy a new version and then set throttling rules. Flexible policies are available. The following table lists several available procedures.

Procedure	Advantage	Problem
Deploy the new version > Set throttling rules	You can verify the maximum traffic for a canary group. The sequence of deploying the application and setting canary release rules are executed for only once.	Before a canary release rule takes effect, the traffic that enters the instances deployed with the new version of the application may be not canary traffic.
Deploy the earlier version > Set throttling rules > Upgrade to the new version	You can verify the maximum traffic for a canary group.	The smooth upgrade is also important. You need to check whether the upgrade of an application on relevant instances affects the requests that are being processed during the upgrade. The application deployment is performed twice.
Set throttling rules > Deploy the new version	The sequence of deploying the application and setting canary release rules are executed for only once.	Before the deployment, canary traffic is downgraded and routed to the non-canary environment. However, after the canary release, the first batch of new-version application instances may be affected by all canary traffic.

Procedure	Advantage	Problem
Set invalid throttling rules > Deploy the new version > Set valid throttling rules	This method has the best controllability. The application instances deployed with the new version receive canary traffic only after the new valid canary release rule take effect.	The setting of canary release rules is performed twice.

#### Note

- We recommend that you use the third method. This method is simple and can ensure that only canary traffic enters the canary environment. However, only a few canary application instances exist at the start of deployment, and application instances may be affected by huge canary traffic after the canary release. Therefore, you need to protect the instances.
- The end-to-end canary release and the combination of throttling for single applications are similar, except that setting throttling rules is changed to creating a canary environment.

### 1.8.4.4.7. Parameters of traffic throttling rules

Canary release allows you to create traffic throttling rules for HTTP and High-Speed Service Framework (HSF) applications. The rule parameters vary based on the application type.

For HTTP requests, you can set throttling rules based on the cookie, HTTP header, and URL. You can determine whether to allow the traffic based on the remainder range or list that is returned by the mod operation (mod 100). This is relatively general and simple, and is not described in detail in this topic. When a parameter value contains a non-digit character, a hash algorithm is used to convert this non-digit character into a digit. If you have complex parameters, we recommend that you use lists to determine whether to perform traffic throttling.

This topic describes the parameters in traffic throttling rules for HSF applications.

The end-to-end canary release allows you to obtain a property of a parameter by using a parameter expression. Currently, the following expressions are supported.

Expression	Description	Remarks
args0	The value of the parameter.	None.
args0.name	The name property of the parameter.	It can be translated into the <code>arg.getName()</code> Java statement.
args0.isEnabled()	The enabled property of the parameter. The data type is BOOLEAN.	In the Java specification, the getter method for a BOOLEAN value uses the form of <code>isXXX()</code> .
args0[0]	Retrieves the first value in the arg array.	None.
args0.get(0)	Retrieves the first value in the arg list.	None.

Expression	Description	Remarks
<code>args0.get("key")</code>	Retrieves the value of the specified key in a map named arg.	None.

If you select the first parameter, Enterprise Distributed Application Service (EDAS) automatically generates an `args0` prefix on the page.

The preceding expressions can be combined, as shown in the following example:

The `args0.persons[0].meta.get("name")` expression retrieves the first parameter in the persons array, retrieves the meta property that is a map in the persons array, and then retrieves the value of the name key in the map.

## Supported operators

- `=` : supports the comparison between strings, numbers, BOOLEAN values, and CHAR values.
- `!=` : supports the comparison between strings, numbers, BOOLEAN values, and CHAR values.
- `>` : supports the comparison between numbers.
- `>=` : supports the comparison between numbers.
- `<` : supports the comparison between numbers.
- `<=` : supports the comparison between numbers.

## Supported value expressions

A value expression in a parameter matching condition in HSF represents a value in Java. Only the basic data types in Java are supported, such as the numeric data types, BOOLEAN, and CHAR. Complex and custom data types are not supported.

The following types of value expressions are supported:

### Standard Java string

A standard Java string is used to represent a string that is enclosed in double quotation marks ( `"` ).

Examples:

- `"tom"` : the tom string
- `"10"` : the 10 string
- `"abc"` : the abc string, which is followed by a space
- `"a"` : the a string
- `"\n"` : a line feed
- `""abc""` : the `"abc"` string
- `"a\bc"` : the a\bc string


This expression can be used to represent all strings based on the syntax of standard Java string expressions.

### Numeric types

To represent a numeric value, you only need to enter a number. Examples:

- 100
- 1.23

- -3.14
- 1.23f

 **Note** In Java, 1.23 is a DOUBLE type value by default. To represent 1.23 of the FLOAT type, use 1.23f. This is determined by the precision of data in Java-based systems.

## BOOLEAN type

The BOOLEAN type has only two values: true and false.

## CHAR type

A CHAR type value represents a character that is enclosed in single quotation marks (''). For example, a CHAR type value can be 'a'.

## Null type


A null type value indicates the null value in Java. You can directly enter null.


## String literal

A string literal is used to represent an exact string so that no characters in this string need to be escaped. The following table provides some examples.

String literal	Java String
tom	"tom"
"	"\""
\	"\""
a\b	"a\b"

The following table lists the value expressions for all types of values.

Value type	Value	Value expression (to be entered)
java.lang.String	"tom"	"tom" or tom
java.lang.String	"true"	"true"
java.lang.String	"10"	<p>"10"</p> <p> <b>Note</b> To represent the value 10 as a string, enclose the number 10 in double quotation marks ("). When double quotation marks (") are absent, this string is parsed as 10 of the numeric type.</p>
java.lang.String	Line feed	"\n"
java.lang.String	'	"\""
java.lang.String	"	"\""

Value type	Value	Value expression (to be entered)
java.lang.String	\	"\"
java.lang.String	aa'bb	"aa'bb"
int	10	10
java.lang.Integer	10	10
byte	10	10
boolean	true	true
java.lang.Boolean	true	true
short	10	10
long	100	100
java.lang.Long	100	100
float	1.23f	1.23f <div> <b>Note</b> If arg is a FLOAT value, you must suffix this value with the letter f. <code>1.23f==1.23</code> returns false.</div>
java.lang.Float	1.23f	1.23f
double	1.23	1.23
java.lang.Double	1.23	1.23
char	'a'	'a'
null	null	null

## Examples

### Parameter type: a string

You do not need to enter anything in the field. If the field is empty, the value expression represents the parameter itself.

### Parameter type: an array

Assume that the parameter is a string array.

You can enter `[0]` in the field to retrieve the first element of the array.

### Parameter type: a list

Assume that the parameter is a `List<String>` object.

You can enter `.get(0)` in the field to retrieve the first element of the list. Make sure that the period (.) is included in the expression.

## Complex parameters

Assume that the first parameter of a method is of the following type:


```
public class Person {  
    private String name;  
    private int age;  
    private String[] array;  
    private List<String> list;  
    private Map<String,String> map;  
}
```

# 1.9. Batch operations

In the EDAS console, you can run machine commands to perform batch operations on the ECS instances with EDAS Agent installed.

## Procedure

1. [Log on to the EDAS console.](#)
2. In the left-side navigation pane, choose **Batch Operations > Machine Commands**.
3. On the **Batch Operations** page, select a region and namespace.
4. In the **Machine Commands** section, click **By Clusters**, **By Applications**, or **By Instances** to determine the operation level.

 **Note** This topic describes operations at the cluster level. The procedures at the other two levels are similar.

5. Click **Add** next to **Select Cluster**. In the **Select Cluster** dialog box, select a cluster (or search for the target cluster by performing a keyword search for its name) in the field on the left. Click **>** to add the cluster to the **Selected** field on the right. Then click **OK**.
6. Enter a command in the **Command** field.
7. (Optional) Select an operation range.
  - **Skip this step** if all the selected items are **ECS clusters**, **common applications**, or **common single-server instances**. The system uses the admin account to log on to instances and run commands.
  - If the selected items include **Swarm or Kubernetes clusters**, **Docker or Kubernetes applications**, or **Docker single-server instances**, select **Execute in Host**, **Execute in Docker Container**, or **Execute in Host and Docker Container** (or select the three options). The system uses the admin account to log on to the host and run commands, and uses the root account to log on to the Docker container and run commands.
8. Click **Run**.

## Result

- View operation results and details

You are redirected to the **View Details** page after commands are executed. The **View Details** page includes the Overview, Basic Information, and Details tabs.

- The Overview tab page shows the comprehensive analysis results of the command execution for batch operations, the number of successful and failed execution instances, and the time consumption.
- The Basic Information tab page shows the batch operator, operating time, and executed commands.
- The Details tab page shows the IP addresses and statuses (successful or failed) of the ECS and Docker instances for batch operations, and the command execution details. The Execution Details section shows the detailed command execution processes on instances. If command execution fails, an error message that indicates the cause is returned.

In this case, select the instance and click **Retry**. You can rerun the command on the selected instance.

- View operation records

On the **Batch Operations** page, view the batch operation record in the lower section. The record contains the operator name, creation time, end time, commands, and status (indicated by the execution results).

- If the current account is the primary account, you can view all the batch commands that are executed by the primary account and all its RAM users.
- If the current account is a RAM user, you can view only the batch commands that are executed by this RAM user.

The entries in the operation record are sorted in descending order by time. You can sort the entries by operator name, creation time, or end time.

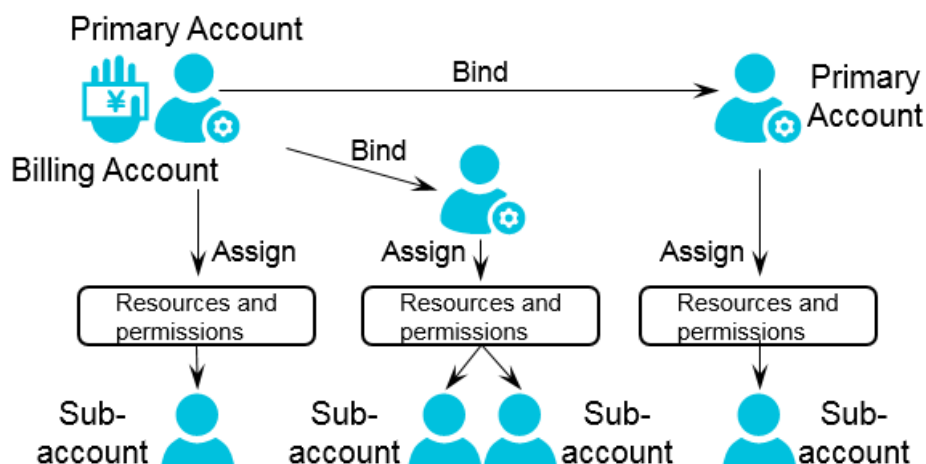
Click **View** in the Details column to go to the Details page.

## 1.10. System management

### 1.10.1. Introduction to the EDAS account system

EDAS provides a comprehensive primary and RAM user management system. A primary account can assign permissions and resources to multiple RAM users as needed in accordance with the minimum permission principle. This lowers the risks to enterprise information security and reduces the work burden on the primary account.

#### EDAS account system



## 1.10.2. Manage RAM users

Resource Access Management (RAM) user management consists of RAM user overview and the Apsara Stack tenant account's operations on the RAM users.

### 1.10.2.1. RAM user overview

When using Enterprise Distributed Application Service (EDAS), you often need to complete different types of tasks as different roles. You can allocate different roles and resources to RAM users under an Apsara Stack tenant account to complete different types of jobs with different user identities. This permission model between the Apsara Stack tenant account and RAM user works in a similar way to the system and common user model in a Linux operating system, where system users can grant or revoke permissions to or from common users.

Relationship between an Apsara Stack tenant account and a RAM user:

- In the EDAS system, you can bind your Apsara Stack tenant account to a RAM user to avoid sharing your AccessKey pair with other users, and assign minimum permissions to the RAM user to complete different types of jobs with different user identities for effective enterprise management.
- When an Apsara Stack tenant account is bound to a RAM user, their binding relationship is valid only within EDAS, and both are independent accounts in other environments.
- An Apsara Stack tenant account can be an Apsara Stack tenant account with RAM users or be a RAM user under another Apsara Stack tenant account.

### 1.10.2.2. Create a RAM role

To authorize a cloud service in a level-1 organization to use other resources in the organization, you must create a RAM role. This role contains the operations that the cloud service can perform on resources.

For more information, see the topic in *Apsara Stack Console User guide* *Apsara Uni-manage User Guide*.

### 1.10.2.3. Use a primary account for RAM user operations

You can use a primary account for RAM user operations, such as Manage Role, Authorize Application, Authorize Resource Group, and Unbind. The procedures for these operations are similar. The following describes how to manage roles in detail and how to perform the other three operations briefly.

#### Context

A primary account can assign a role to a RAM user to grant the role-associated permissions to this sub-account.

#### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **System Management > Sub-Account**.
3. Locate the row that contains the target RAM user, and click **Manage Roles** in the Actions column.
4. Select the target role and click **OK**.  
After the preceding settings, the role name appears in the Role field for the RAM user on the Sub-Accounts page.

 **Note**

- **Authorize an application**

A primary account can assign an application to a RAM user to grant the application ownership to this RAM user.

Application authorization only grants the application ownership to the RAM user. To grant application operation permissions (to start or delete the application, for example) to the RAM user, assign a role to the RAM user. Therefore, application authorization is typically followed by role authorization.

- **Authorize a resource group**

A primary account can assign a resource group to a RAM user, allowing the RAM user to use resources in the resource group. For the definition of a resource group, see [Resource management](#).

- **Unbind**

Through the unbinding operation, you can release the binding relationship between a RAM user and the primary account. The relationships with the assigned role, application, and resource group are also released. If you have not bought the EDAS service for the RAM user, you cannot log on to the EDAS console by using this RAM user after unbinding.

## 1.10.3. Manage roles

A primary account can define different operation permissions for its RAM users by creating different roles.

### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **System Management > Roles**.
3. Click **Create Role** in the upper-right corner of the page.
4. Enter a role name, add the permissions in the left-side field to the right, and click **OK**.  
After a role is added, you can perform actions on this role, such as **View Permissions**, **Manage Permissions**, and **Delete**.

## 1.10.4. View all permissions

You can list all permissions of the EDAS system in the console.

### Procedure

1. [Log on to the EDAS console](#).
2. In the left-side navigation pane, choose **System Management > All Permissions**.
3. Click a level to view the details of permissions at this level.

## 1.11. FAQ

This topic describes the common problems and solutions during product development and use.

### 1.11.1. Known issues and solutions

This topic describes some known issues of the features in the current version of EDAS and the corresponding solutions. If you encounter these issues, resolve them by using the following methods.

- **What can I do if the application monitoring data page is dimmed? Why am I unable to open the application monitoring data page?**  
Cause: This issue occurs if you do not purchase a certificate. In this case, an Application Real-Time Monitoring Service (ARMS) page uses a self-signed or invalid certificate over HTTPS. Therefore, the access is blocked by the browser.  
Solution: Open the blocked ARMS page on a separate tab, and then open it again in EDAS.
- **Why is no monitoring data displayed after an application is upgraded?**  
Cause: This issue occurs when an application is upgraded to a later version. Since the EDAS monitoring solution has changed in version 3.9.0, you must restart the application to enable monitoring.  
Solution: If you perform a hot upgrade for your application, restart the upgraded application so that you can view the basic monitoring data and the service monitoring data. If you cannot restart the application, remotely connect to an ECS instance and perform the following steps:
  - i. Reinstall EDAS Agent.
  - ii. Switch to the admin user and run the `edas refresh-apm` command.
- **After I unbind a vServer group of an SLB instance that is configured with routing methods from an application instance group, the SLB instance fails to forward traffic to the application and the HTTP 503 status code is returned. How can I fix this issue?**  
Cause: After you unbind a vServer group of an SLB instance from an application instance group in the EDAS console, EDAS does not delete the routing methods of the SLB instance. However, the backend servers in this vServer group is also unbound from the application. As a result, if the SLB instance still follows the routing methods to forward traffic, the HTTP error 503 occurs.  
Solution: Delete the routing methods of the unbound vServer group in the SLB console.
- **Why does an application to which an SLB instance is bound fail to be accessed by using the IP address of the SLB instance after traffic management is enabled? How can I fix this issue?**  
Cause: In this scenario, traffic management is enabled for an application and an SLB instance that supports HTTP is bound to the application. In this case, a limit exists. When the SLB instance uses HTTP connections to detect whether the backend nodes are alive, the "HEAD / HTTP/1.0" message is sent. Tengine returns the HTTP 400 status code. As a result, the health check on the SLB listener fails. Therefore, a 502 bad gateway error occurs when you attempt to access the application.  
Solution: Log on to the EDAS console. Disable traffic management on the Application Information page of the applications that do not require traffic management. This operation uninstalls Tengine, modifies the application configurations, and restarts the application. To retain this feature, select Code 4xx that is returned by health check on the Health Check page of the SLB console.
- **What can I do if a task is stuck and cannot be scheduled due to a change order lag?**  
When a change order lags, restart the two EDAS Enterprise Asset Management (EAM) containers and try again.
- **What can I do if the content of the mount script is cleared but the last content remains?**  
We recommend that you change the script to a null statement `echo ""` for bypass.
- **If you cancel HTTP rules and click Save when you configure traffic throttling, the server reports the error "execution failed". Why is the error reported?**  
Generally, the error message is reported when you cancel HTTP rules for the first time. However, you can cancel the HTTP rules by saving the configuration on the current page again.
- **After you use a RAM user to create an application, the error "no permission" is reported when you delete the application. However, the application is deleted. How can I fix this**

issue?

Cancel the error dialog box although this error message may confuse you. The application can be deleted, and the error will not appear later.

- I can view ECS instances in the cluster list by using a RAM user. Why am I unable to use it to view the corresponding ECS instances on the ECS management page or in the application scale-out list?

All ECS instances in the cluster are displayed in the cluster list. However, when you query the ECS instance list or scale out the application, the system checks whether the permission to access an ECS instance is granted to a RAM user. If the permission to access the ECS instance is not granted to the RAM user, the ECS instance cannot be used.

## 1.11.2. Development FAQ

The development FAQ covers Ali-Tomcat, lightweight configuration center, HSF, HSF error codes, and other development problems.

### 1.11.2.1. Ali-Tomcat FAQ


This topic describes the problems frequently encountered during the Ali-Tomcat development process and their solutions.

- **Problem locating procedure**

Ali-Tomcat may fail to start due to various errors. Check the catalina.out and localhost.log files to locate the error. If you use the Tomcat4E plug-in, you can view the detailed problem description in the Eclipse console.

- **How do I distinguish an EDAS error from a code error when an exception occurs?**

Check whether the last part of the error stack contains the code itself. Example: Caused by: com.yourcompany.yourpack.

Problem	Error message	Solution
Service authentication failure	<p>java.lang.Exception: Service authentication failed</p> <div><p> <b>Note</b> This problem only occurs in the EDAS production environment.</p></div>	<ul style="list-style-type: none"><li>• The AccessKeyId and AccessKeySecret used for installing EDAS Agent are incorrect or they became incorrect due to web-based installation or other reasons.<ol style="list-style-type: none"><li>i. Run <code>cat /home/admin/.spas_key/default</code>.</li><li>ii. Log on to the EDAS console. In the left-side navigation pane, choose <b>Resource Management</b> &gt; <b>ECS</b>. On the <b>Instances</b> page, click <b>Install Agent</b>.</li><li>iii. On the page that appears, check whether AccessKeyId and AccessKeySecret are set to the preceding values (case-sensitive). Web-based installation may cause case inconsistency.</li></ol></li><li>• The ECS instance has a delay of more than 30s. Adjust the time of the ECS instance.<ul style="list-style-type: none"><li>◦ Run the <code>date</code> command to check whether the date is accurate.</li></ul></li></ul>

Problem	Error message	Solution
Unknown host exception	Caused by: java.net.UnknownHostException: iZ25ax7xuf5Z	iZ25ax7xuf5Z indicates the current hostname. Check whether <code>/etc/hosts</code> contains the IP address and name of the current host. If not, configure them, for example, <code>192.168.1.10 iZ25ax7xuf5Z</code> .
Port in use	Caused by: java.net.BindException: Address already in use: JVM_Bind	The port is in use. The troubleshooting method is the same as the method for troubleshooting port conflict in the lightweight configuration center.
com.aliyun.unit.rule.Router initialization failure	SEVERE: Context initialization failed java.lang.NoClassDefFoundError: Could not initialize class com.aliyun.unit.rule.Router	Address server connection failure jmenv.tbsite.net. Bind the domain. Add the following content to the hosts file to bind the domain name server address: <code>192.168.1.10 jmenv.tbsite.net</code> . Change 192.168.1.10 to the IP address of your lightweight configuration center. The path to the hosts file is as follows: <ul style="list-style-type: none"> <li>Windows: <code>C:\Windows\System32\drivers\etc\hosts</code></li> <li>Linux: <code>/etc/hosts</code></li> </ul>
QoS port binding exception, resulting in a Pandora startup failure	Cannot start pandora qos due to qos port bind exception	The QoS port is in use. The troubleshooting method is the same as the method for troubleshooting port conflict in the lightweight configuration center.
Insufficient JVM memory	java.lang.OutOfMemoryError	Set the memory size. For more information about the solution, search <b>JVM memory settings</b> on the Internet.
A null pointer exception during WAR package deployment	deployWAR NullPointerException	Check whether the WAR package is normal. Run <code>jar xvf xxx.war</code> to check whether the WAR package can be decompressed properly.
com.taobao.diamond.client.impl.DiamondEnvRepository initialization failure	Could not initialize class com.taobao.diamond.client.impl.DiamondEnvRepository	If DiamondServer data on the address server is empty, check whether the address server is correctly configured and is running stably. Access <a href="http://jmenv.tbsite.net:8080/diamond-server/diamond">http://jmenv.tbsite.net:8080/diamond-server/diamond</a> and check whether a response is properly returned.

## 1.11.2.2. Lightweight configuration center FAQ

This topic describes the common problems related to the lightweight configuration center and their solutions.

Problem	Error message	Solution
Startup fails when startup.bat and startup.sh are executed.	Java version not supported, must be 1.6 or 1.6+	Check whether Java is properly installed. If Java is not installed, install Java 1.6 or a later version.
	Unable to start embedded Tomcat servlet container	Check whether port 8080 is in use. If the port is used by another application, stop the application and run the startup script. Perform the following operations: <b>Windows:</b>
	Tomcat connector in failed state	<ol style="list-style-type: none"> <li>1. Open the CMD window and run <code>netstat -aon findstr "8080"</code> . Record the last column of numbers in the queried data, that is, the process ID (PID), such as 2720.</li> <li>2. Run <code>tasklist findstr "2720"</code> . The application that corresponds to the current PID, such as javaw.exe, appears.</li> <li>3. Run <code>taskkill /PID 2720 /T /F</code> .</li> <li>4. Start the lightweight configuration center again.</li> </ol> <b>Linux:</b> <ol style="list-style-type: none"> <li>1. Run <code>netstat -antp grep 8080</code> . The PID of the process that uses port 8080 appears, for example "2720".</li> <li>2. Run <code>kill -9 2720</code> .</li> <li>3. Start the lightweight configuration center again.</li> </ol>
	Caused by: java.net.UnknownHostException: iZ25ax7xuf5Z	iZ25ax7xuf5Z indicates the current hostname. Check whether the IP address and name of the current host are configured in /etc/hosts. If not, configure them, for example, 192.168.1.10 iZ25ax7xuf5Z.

- **How do I specify the startup IP address for instances with multiple NICs?**

In the startup script startup.bat or startup.sh, add the startup parameter -Daddress.server.ip={accessible IP address}.

- **How do I customize service publishing IP addresses?**


In some cases, a service must be published on a vNIC or a non-physical IP address (for example, the EIP of an ECS instance) associated with the local host. If the virtual IP address is specified by using -Dhsf.server.ip, an error may occur when the service is started and the service cannot be published. This is because the virtual IP address cannot be found on the NIC of the local host during publishing. To solve this problem, EDAS provides the service IP address customization function for the provider that allows the provider to publish a service in the configuration center without specifying any IP address. After the service is successfully published, modify the IP address and then publish the service again. The consumer does not need to make any changes.

Perform the following operations:

- i. After the service is published, find it in **Configuration List** and click **Update** on the right of the service.

You can also find the published service on the **Services** tab page.

- ii. On the **Edit Configuration** page, modify the IP address in the Content field.

 **Notice** Do not modify the content after the IP address unless necessary. Otherwise, a service call error may occur.

- iii. Click **OK** to save the settings.
- iv. Restart the service. The service with the new IP address is registered again to enable the modification to take effect.

After modification, the consumer does not need to make any changes and can call the service in the normal way. You can query logs in `{user.home}\logs\configclient\config-client.log` to check the real IP address that is called by the consumer. Check the content next to the keyword **[Data-received]** in the logs to view the complete information about the called service.

### 1.11.2.3. HSF FAQ

- **Locate and solve HSF problems**

HSF problems are logged in `/home/admin/logs/hsf/hsf.log`. If any HSF problem occurs, check this file to locate the error. HSF errors have corresponding error codes. You can use these error codes to find the appropriate solution.

- **Set the timeout period for an HSF service**

Use the HSF tags `methodSpecials` and `clientTimeout` to configure the timeout period.

- **methodSpecials**: sets the timeout period (unit: ms) for a single method.
- **clientTimeout**: sets the general timeout period (unit: ms) for all methods in the interface.

The timeout period settings are sorted in descending order of priority as follows:

Consumer `methodSpecials` > Consumer `clientTimeout` > Provider `methodSpecials` > Provider `clientTimeout`

An example of the Consumer tag settings is as follows:

```
<hsf:consumer id="service" interface="com.taobao.edas.service.SimpleService"
version="1.1.0" group="test1" clientTimeout="3000"
target="10.1.6.57:12200?_TIMEOUT=1000" maxWaitTimeForCsAddress="5000">
<hsf:methodSpecials>
  <hsf:methodSpecial name="sum" timeout="2000" ></hsf:methodSpecial>
</hsf:methodSpecials>
</hsf:consumer>
```

- **HSF invalid call is removed**

**Error message:**

invalid call is removed because of connection closed

**Causes:**

- **Transient network disconnection**: After the provider and consumer establish a connection, the consumer initiates a call request. An error is returned if the provider is still processing this request within the timeout period of the consumer and the consumer is disconnected due to network and other problems.
- **Provider restart**: After the consumer initiates a request, it waits for a response from the provider. If the consumer is restarted at this time, the socket is disconnected and the consumer receives an operating system connection closed callback. In this case, an error is returned.

**Solution**

If the service is idempotent, retry the service. Check the HSF provider network. This problem is often caused by network disconnection (transient disconnection).

- **Binding an IP address and port fails upon HSF start up**

**Problem:** An error is returned when HSF is started. The error message is as follows:

**Java.net.BindException: Can't assign requested address**

**Cause:** The current IP address and port cannot be obtained.

**Solution:** Set the following JVM parameter:

-Dhsf.server.ip=IP address of your local network adapter -Dhsf.server.port=12200

- **Keep user logs from being overwritten**

**Problem:** After EDAS is used, the log4j log cannot be generated.

**Cause:** The log4j log is overwritten and thus cannot be generated.

**Solution:** Set the JVM parameter Dlog4j.defaultInitOverride to false to generate user logs.

- **HSF Others**

**Error message:** The following error is reported during start up:

java.lang.IllegalArgumentException: HSFApiConsumerBean.ServiceMetadata.ifClazz is null.

**Solution:** The class for the interface cannot be loaded. Check that the interface class is loaded to class path.

**Error message:** failure to connect 10.10.1.1

**Solution:**

Check whether the HSF services are in the same VPC and the same region. If not, they cannot be connected.

Check whether the HSF services are in the same security group. If not, enable port 12200.

Run `telnet 10.10.1.1 12200` to check whether the port can be connected. If the port cannot be connected, check the firewall settings of the ECS instance with the IP address 10.10.1.1.

## 1.11.2.4. HSF error codes

### Error code: HSF-0001

**Error message:**

HSFServiceAddressNotFoundException: This error message is returned when the address of the target service to be called is not found.

**Description:**

The target service to be called is xxxx, which is in the xxxx group.

**Solution:**

1. In the case of name mismatch, check whether the service name, version, and group (case-sensitive, without leading or trailing spaces) are set consistently for the provider and consumer.
2. Check whether an error is reported when the Tomcat container is started. Go to the Tomcat installation directory and check whether /logs/catalina.out localhost.log. 2016-07-01 (current date) contains any errors. If yes, fix the errors.
3. No service group is created. Log on to the EDAS console. In the left-side navigation pane, choose **Service Marketplace > Service Groups** to check whether a service group is created for the application. Example:

```
<hsf:provider
  id="sampleServiceProvider" interface="com.alibaba.edas.SampleService" ref="target"
  version="for-test" group="your-namespace" ></hsf:provider>
```

The corresponding group named *your-namespace* must exist in the service group list.

4. In the case of failed authentication, go to the ECS instance that corresponds to the **provider** and check whether `/home/admin/configclient/logs/config.client.log` contains the `spas-authentication-failed` error. If this error exists:
  - No service group is created.
  - The `AccessKeyId` and `AccessKeySecret` used for installing EDAS Agent are incorrect or they became incorrect due to web-based installation or other reasons.
    - a. Run `cat /home/admin/.spas_key/default` .
    - b. Log on to the EDAS console. In the left-side navigation pane, choose **Resource Management > ECS** and click **Install Agent** .
    - c. On the page that appears, check whether `AccessKeyId` and `AccessKeySecret` are set to the preceding values (case-sensitive). Web-based installation may cause case inconsistency.
    - d. The IP address of the provider cannot be pinged. If multiple NICs exist, publish the IP address that is inaccessible from the consumer. Use `-Dhsf.server.ip` to specify the IP address of the provider.
5. The service call is initiated too early. A call is initiated before `ConfigServer` pushes the address, resulting in an error. Add `maxWaitTimeForCsAddress` to the consumer configuration file. For more information, see *Developer Guide*.
6. In the case of a data push error, contact a developer for troubleshooting.

## Error code: HSF-0002

Error message:

Consumer error: HSFTimeOutException

Solution:

- Check whether the network of the ECS instance is healthy. Check whether the IP address of the provider can be pinged.
- If the processing time of the provider is greater than 3s, find the service execution timeout logs in `hsf.log` of the provider to locate the specific class and method:
  - A serialization error has occurred for the provider. Check the codes. The stream type, files, and oversized objects may cause a serialization error, and they cannot be transferred.
  - The code performance is inadequate. Optimize the code.
  - The logic of the provider is complex, and service processing requires more than 3s. Modify the timeout period. (See the *Developer Guide*.)
- Timeout occurs occasionally, and GC occurs for both the provider and consumer. Check the GC logs of the provider and consumer. GC that requires a long time may result in timeout. For more information about troubleshooting methods, search **Java GC optimization** on the Internet.
- The consumer is heavily loaded and fails to send the request, resulting in timeout. Add more instances for the consumer.

## Error code: HSF-0003

Error message:

Consumer error: `java.io.FileNotFoundException: /home/admin/logs/hsf.log` (The specified path is not found.)

**Solution:** The default HSF log path cannot be found or is under access control. Load -DHSF.LOG.PATH=xxx during startup to modify the default path.

## Error code: HSF-0005

**Error message:**

Startup error:

java.lang.IllegalArgumentException: This error message is returned when the object to be published as a service is not configured. The service name is com.taobao.hsf.jar.test.HelloWorldService:1.0.zhouli.

**Solution:**

The target attribute is missing from the bean of the provider. Check the configuration file.

The implementation class of the service specified by target does not exist. Check the configuration file.

## Error code: HSF-0007

**Error message:**

java.lang.IllegalArgumentException: This error message is returned during startup when the serialization type is not supported.

**Solution:** The serializeType or preferSerializeType attribute is incorrectly configured for the bean of the provider. Check the configuration file. We recommend that you use Hessian or Hessian 2.0.

## Error code: HSF-0008

**Error message:** java.lang.IllegalArgumentException, which is returned when the service type specified by ProviderBean is not [com.taobao.hsf.jar.test.HelloWorldServiceImpl].

**Solution:** serviceInterface configured for the bean of the provider is not an interface. serviceInterface must be set to an interface name. Check the configuration file.

## Error code: HSF-0009

**Error message:** java.lang.IllegalArgumentException, which is returned when the real service object [com.taobao.hsf.jar.test.HelloWorldServiceImpl@10f0a3e8] does not implement the specified interface [com.taobao.hsf.jar.test.HelloWorldService].

**Solution:** No interface is implemented for the bean specified by target of the provider. Check that the corresponding interface is implemented in the interface class.

## Error code: HSF-0014

**Error message:** java.lang.IllegalArgumentException, which is returned when the interface class specified by ProviderBean does not contain [com.taobao.hsf.jar.test.HelloWorldService1].

**Solution:** The serviceInterface attribute of the provider is incorrectly configured, and the specified interface does not exist.

## Error code: HSF-0016

**Error message:**

Startup error: Failed to start the HSF provider.

**Solution:**

- Check whether port 12200 is already occupied. A server binding failure may cause a startup failure.
- If multiple NICs and an instance with a public network IP address exist, specify the local IP address by using -Dhsf.server.ip.

## Error code: HSF-0017

### Error message:

Start up error: java.lang.RuntimeException: [ThreadPool Manager] Thread pool allocated failed for service [com.taobao.hsf.jar.test.HelloWorldService:1.0.zhouli]: balance [600] require [800]

**Solution:** The allocated thread pool is insufficient. By default, the maximum thread pool size of HSF is 600. You can set the JVM parameter `-Dhsf.server.max.poolsize=xxx` to modify the default global maximum thread pool size.

## Error code: HSF-0020

### Error message:

WARN taobao.hsf - HSF service: com.taobao.hsf.jar.test.HelloWorldService:1.0.zhouli, which is returned when initialization is repeated.

**Solution:** In one HSF process, a service is uniquely identified by the service name and version. Services with the same name and version but of different groups cannot be published or subscribed to in a single process. Check the configuration file. For example, the service `com.taobao.hsf.jar.test.HelloWorldService` cannot be published or subscribed to in a single process if the following two configurations exist in the configuration file:

```
com.taobao.hsf.jar.test.HelloWorldService 1.0 groupA
```

```
com.taobao.hsf.jar.test.HelloWorldService 1.0 groupB
```

## Error code: HSF-0021

### Error message:

Start up error:

java.lang.IllegalArgumentException, which is returned when the interface class specified by ProviderBean does not contain [com.taobao.hsf.jar.test.HelloWorldService1].

java.lang.IllegalArgumentException: This error message is returned when the interface class specified by ConsumerBean does not contain [com.taobao.hsf.jar.test.HelloWorldService1].

**Solution:** The serviceInterface attribute of HSFSpringProviderBean is incorrectly configured, the specified interface does not exist (HSF-0014), or the interface specified by the interfaceName field in HSFSpringConsumerBean does not exist (HSF-0021).

## Error code: HSF-0027

**Error message:** [HSF-Provider] HSF thread pool is full

### Solution:

The processing speed of a service on the HSF provider is too slow, and requests from the client cannot be processed in time. As a result, the thread pool of the HSF provider for service execution reaches the maximum value. By default, HSF dumps the `/home/admin/logs/hsf/HSF_JStack.log` file (default path). View the **HSFBizProcessor-xxx** thread stack information about the file and analyze the performance bottleneck.

The maximum number of threads of HSF is 600 by default. To increase the number, change the value of the `-Dhsf.server.max.poolsize=xxx` JVM parameter.

## Error code: HSF-0030

**Error message:** [HSF-Provider] cannot find the method to be called.

### Solution:

- The method is not provided by the provider. Log on to the EDAS console. In the left-side navigation pane, choose **Application Management** and click the name of the application that corresponds to the service provider to go to the Application Details page. In the left-side navigation pane, choose **Services** and check whether the corresponding service is successfully published.
- An earlier version and a later version coexist. The wrong version of a service is called. View the details of the service by using the preceding method.
- The interfaces of the provider and the consumer are inconsistent. For example, the provider provides java.lang.Double, whereas the consumer uses double to call the provider.
- Inconsistent interface classes are loaded for the provider and consumer. Check whether the MD5 values in the interface-contained JAR packages of the provider and consumer are consistent.

### Error code: HSF-0031

**Error message:** [HSF-Provider] takes xxx ms to execute the xxx method of the xxx HSF service. The time approximates the timeout period.

**Solution:** The provider prints this log when the timeout period minus the actual time elapsed is less than 100 ms. The timeout period is 3s by default.

- If the timeout period is short, for example, less than 100 ms, this log is printed in each call, and you can ignore it.
- If this log is still printed for a long timeout period, it indicates service execution is slow. Analyze the performance bottleneck in service execution.

### Error code: HSF-0032

**Error message:** please check log on server side that unknown server error happens.

**Solution:** An uncaptured error occurs when the provider processes a request. Check the hsf.log file of the provider.

### Error code: HSF-0033

**Error message:** Serialization error during serialize response.

**Solution:**

An error occurs when the provider returns data during serialization. Check the hsf.log file of the provider.

If the log file contains "must implement java.io.Serializable", implement a serializable interface on the DO.

### Error code: HSF-0038

**Error message:** Multiple NICs are configured for the HSF provider, and the HSF provider is bound to an incorrect IP address.

**Solution:** Add -Dhsf.server.ip=xxx.xxx.xx.xx to the JVM startup parameters to specify the desired IP address.

### Error code: HSF-0035

**Error message:** RPCProtocolTemplateComponent invalid address.

**Solution:** A TCP connection cannot be established between the current instance and the corresponding address. Check whether the corresponding remote address and port can be connected.

## 1.11.2.5. Other development problems

- Q: How do I develop an HSF application by using a framework other than Spring?  
A: We recommend that you use Spring to develop HSF applications. If you use another framework, you can develop applications by using LightAPI. For more information, see the *Developer Guide*.
- Q: Can I access the services in a production environment directly from a development environment?  
A: No. The production environment is isolated for security.
- Q: Does EDAS provide APIs? What functions do they have?  
A: EDAS provides APIs to implement resource query, application lifecycle management, and account management.
- Q: Does EDAS support other languages in addition to Java?  
A: HSF is developed in Java by default. HSF clients are also available in C++ and PHP, allowing you to access the backend HSF services provided by Java.

## 1.11.3. Usage FAQ

Common problems during development are related to accounts, resources, application lifecycle, and monitoring and alarms.

### 1.11.3.1. Account management

- Q: Can I create multiple RAM users?  
A: Yes.
- Q: Who can grant application operation permissions for RAM users?  
EDAS allows you to grant application operation permissions to RAM users only by using the primary account.

### 1.11.3.2. Resource management

- Q: Why doesn't the a prompt appear after EDAS Agent installation and EDAS Agent version is not displayed?  
A: Perform the following steps to troubleshoot the problem:
  - i. Log on to the ECS instance and check `/home/admin/edas-agent/logs/agent.log`. If `UnauthorizedException` exists, check whether:
    - The `AccessKeyId` and `AccessKeySecret` used for installing EDAS Agent are incorrect or they became incorrect due to web-based installation or other reasons.
      - a. Run `cat /home/admin/.spas_key/default`.
      - b. Log on to the EDAS console. In the left-side navigation pane, choose **Resource Management** > **ECS**. On the **Instances** page, click **Inst all Agent**.
      - c. On the page that appears, check whether `AccessKeyId` and `AccessKeySecret` are set to the preceding values (case-sensitive). Web-based installation may cause case inconsistency.
    - The region script used for installation is incorrect.
  - ii. Check `/home/admin/edas-agent/logs/std.log`. If "Java not found" or other error messages exist, run `java -version` to check whether the Java version is 1.7. If the version is Java 1.5, run `rpm -e corresponding installed RPM name` to remove it and reinstall EDAS Agent.
- Q: Why is the status Unknown or Abnormal after EDAS Agent is installed?

A: Check the std.log and agent.log files in the `/home/admin/edas-agent/logs` directory of the ECS instance.

- std.log is the log of EDAS Agent installation.
- agent.log is the runtime log of EDAS Agent.

The possible causes are as follows:

- If "*Permission denied*" or "*Not such file*" is found in those logs, the possible cause is the lack of required file and directory permissions. In this case, check whether the admin account has permissions for all files in the `/home/admin` directory, and reinstall EDAS Agent.
- Check whether the ECS hostname is the same as that in the `/etc/hosts` file. If not, modify the name and restart EDAS Agent.

```
/home/admin/edas-agent/bin/shutdown.sh
/home/admin/edas-agent/bin/startup.sh
```

- Q: Which version of Java is EDAS using? Can I choose another version?

A: EDAS provides Java 7 and Java 8. Java 7 is used by default. You can select a Java version when installing EDAS Agent. Run the following command to select a Java version:

```
install.sh -ak -sk [-java <7(default)|8>]
```

- Q: What can happen if the heartbeat process of EDAS Agent stops?

A: If no application is installed on that ECS instance, no services are affected. If an application is installed on that ECS instance, the *real-time status* of the ECS instance in the ECS instance list of the application (which appears in the lower part of the page after you select the application on the **Application Management** page and go to the **Basic Information** page) changes to **Agent Abnormal**. Any commands for the ECS instance, such as `deploy`, `start`, and `stop`, are ineffective. Log on to the ECS instance and run `sudo -u admin /home/admin/edas-agent/bin/startup.sh` to start EDAS Agent. Troubleshoot the EDAS Agent crash as follows:

Check whether error messages are logged in `/home/admin/edas-agent/logs/agent.log`.

Check whether the system memory is sufficient. If the system memory is insufficient, the OOM Killer may be triggered. For more information, search for Linux OOM Killer on the Internet. If the OOM Killer is triggered, we recommend that you check the system memory usage and adjust memory allocation.

- Q: What should I do if the Ali-Tomcat container suddenly exits?

A: Log on to the EDAS console to start the corresponding application. Troubleshoot the crash of Ali-Tomcat as follows:

- Check whether error messages are logged in `/home/admin/tomcat (installation directory)/logs/catalina.out`.
- Check whether the system memory is sufficient. If the system memory is insufficient, the OOM Killer may be triggered. For more information, search for Linux OOM Killer on the Internet. If the OOM Killer is triggered, we recommend that you check the system memory usage and adjust the memory allocation policy.

- Q: Why doesn't EDAS Agent start after the system is restarted?

A: Currently, EDAS Agent of the CentOS 6.5 version supports automatic startup. Testing is not performed in other systems for the moment. If EDAS Agent is not started, run the following program:

```
sudo -u admin /home/admin/edas-agent/bin/startup.sh
/usr/alisis/dragon/bin/DragonAgent
```

### 1.11.3.3. Application lifecycle

- Does EDAS Agent automatically restart after the required ECS instance is restarted?  
Yes, EDAS Agent automatically restarts after the target ECS instance is restarted. However, your Tomcat does not restart.
- Why am I unable to start EDAS Agent?  
Run the following ping command on the server to check whether the EDAS console server can be accessed.

```
ping edas-internal.console.aliyun.com
```

Then, check whether the specified security token file is valid.

```
cat /home/admin/.spas_key/default
```

- Can I deploy multiple applications on the same ECS instance in EDAS?  
EDAS allows you to deploy only one application on a single ECS instance.
- Can I specify the URL of an application deployment package as an address that I want?  
Make sure that the application deployment package can be downloaded on your server from this URL.
- Why does an application operation (such as starting, stopping, or deploying an application) fail?  
Check whether EDAS Agent runs as expected on the server where the failure occurs. An application operation usually fails because EDAS Agent does not run as expected.
- Why do the ECS instances for my account fail to appear in the instance selection dialog box when I create an application?  
Check whether EDAS Agent is validly installed on the server. For more information, see the steps in Resource Management > ECS > Install EDAS Agent.



**Notice** Select a valid region when you install EDAS Agent.

- Why is the ECS instance status Unknown in the EDAS console?  
EDAS Agent periodically reports heartbeat data to the EDAS console. If EDAS Agent stops reporting data for some time, the EDAS console determines that the instance is in Unknown state. When EDAS Agent stops, the issue occurs. The cause is general.
- The service list does not appear, but services can be called as expected. Why does this issue occur?  
APIs have generics, but the generics do not have a specific type. This results in a failure to resolve the service list. In this case, modify the corresponding code.
- What can I do when a service appears as Normal in the service list, but I cannot call it?
  - i. Check whether the group that corresponds to the service provider has been created. If the group has not been created, the service provider fails to be authenticated.
  - ii. Check `/home/admin/logs/hsf/hsf.log` to view the error code, and query [HSF FAQ](#) based on the error code.
- Can I restore an application after I delete it?  
No, you cannot restore an application after you delete it. Application deletion is irrevocable. All the application data is cleared after the application is deleted.
- How do I perform batch or beta publishing?
  - If an application has multiple ECS instances, select batch publishing and set the number of batches to a value greater than 1 to publish the application in batches.

- If an application has multiple ECS instances, set some of these instances as beta instances. You can publish the application only to the beta instances. When the application is published, only beta instances are updated. Other instances are not updated.
- How do I share cluster sessions after my application is deployed on multiple ECS instances?  
EDAS does not support distributed session management. You can use a distributed cache system, such as ApsaraDB for MongoDB (OCS) or ApsaraDB for Redis (Redis), to manage distributed sessions.
- How do I specify the health check URL?  
When an application is released, the WAR package that you provide is automatically deployed in the Tomcat directory. Therefore, the WAR package name must be added to the health check URL by default. In addition, the files in the WAR package must return an HTTP code that ranges from 200 to 400. For example, if your WAR package is *order.war* and contains the *index.jsp* file, the health check URL can be set to `http://127.0.0.1:8080/order/index.jsp`.
- Can I use SLB for load balancing after I deploy my application on multiple servers?
  - i. HTTP-based web applications in EDAS use SLB for load balancing. You can configure SLB on the application configuration page of EDAS.
  - ii. You do not need to consider load balancing for applications that belong to the RPC providers of EDAS. EDAS supports load balancing for RPC providers.

### 1.11.3.4. Troubleshoot EDAS Agent issues

#### Condition

Enterprise Distributed Application Service (EDAS) Agent becomes abnormal or a container exits.


#### Cause

EDAS Agent becomes abnormal. The system time of the Elastic Compute Service (ECS) instance is not synchronized with that of EDAS Agent, which has a deviation of 30s. The network resolution times out.

#### Remedy


#### Procedure

1. Log on to the ECS instance where the application is deployed and go to the `/home/admin/edas-agent/` directory. View the `agent.log` log to check whether the ECS instance where EDAS Agent is deployed is properly connected to EDAS.
2. Check whether the system time of the ECS instance where the application is deployed is synchronized with that of the components such as `edasServer`, `Dauth`, `edasConsole`, and `edasAdmin`.

 **Notice** In normal cases, a component consists of multiple nodes. If the system time of the nodes that constitute a component is not synchronized, an issue occurs.

3. Check the AccessKey ID and AccessKey secret.
  - i. Log on to the ECS instance where the application is deployed. Run the `cat /home/admin/.spas_key/default` command on the ECS instance to query the AccessKey ID and AccessKey secret.
  - ii. Log on to the EDAS console. In the left-side navigation pane, choose **Resource Management** > **ECS**.
  - iii. On the ECS page, click **Import ECS**. On the page that appears, click **Switch to Manual Installation**.

- iv. On the **Manually Install EDAS Agent on Single Instance** page, view the AccessKey ID and AccessKey secret that are used for installing EDAS Agent.
- v. Check whether the AccessKey ID and AccessKey secret used on the ECS instance are the same as those on the **Manually Install EDAS Agent on Single Instance** page.

 **Notice** Take note of the case sensitivity. If you use a web terminal to install EDAS Agent, this may cause case inconsistency.

- vi. Log on to the ECS instance where the application is deployed. Run the `dmidecode -s system-serial-number | grep -v '^#'` and `dmidecode | grep -i uuid | awk '{print tolower($2)}'` commands to check whether the UUIDs are the same.
  - If the UUIDs are consistent, this issue is not caused by inconsistent AccessKey IDs or AccessKey secrets. No further action is required.
  - If the UUIDs are inconsistent, run `cp /usr/sbin/staragent_sn /usr/sbin/staragent_sn.bak && dmidecode | grep -i uuid | awk '{print tolower($2)}' > /usr/sbin/staragent_sn` and `pkill -9 staragent && /home/staragent/bin/agent.sh start`. Then, execute the following SQL statement to verify the result:

```
-select * from ecs where serial-num="";  
/home/staragent/conf/staragent.conf
```

## 1.11.3.5. Troubleshoot change process issues

### Condition

The deployment of an application suspends and cannot proceed to the next step.

### Cause

- Enterprise Distributed Application Service (EDAS) Agent that is deployed on an Elastic Compute Service (ECS) instance becomes abnormal and the client fails to deliver commands. This results in a failed job whose execution is dependent on EDAS Agent.
- A job of a service type fails to run because the management components of EDAS become abnormal. The following jobs are of a service type: Bring Tengine online or offline. Update Tengine. Bring Server Load Balancer (SLB) online or offline. Set the weight for load balancing.
- A health check reports an error because the application package is abnormal.

### Remedy

### Procedure

1. The number of online health checks that failed accounts for more than half of the total number of failed jobs. In normal cases, if the configuration of a health check is invalid or an issue exists in your application package, the health check fails. If a URL health check fails, check whether the URL for the health check is correctly configured.
  - If the health check configuration is invalid or an issue exists in your application package, configure the URL for the health check again or fix the issue of the application package.
  - If you verify that this issue is not caused by the health check configuration or application package, proceed to the next step.
2. Log on to the ECS instance where the application is deployed. Check the startup logs of Tomcat and application logs to troubleshoot the issue.

3. Start the change process again and check whether a command delivery timeout results in this issue.

## 2. API Gateway

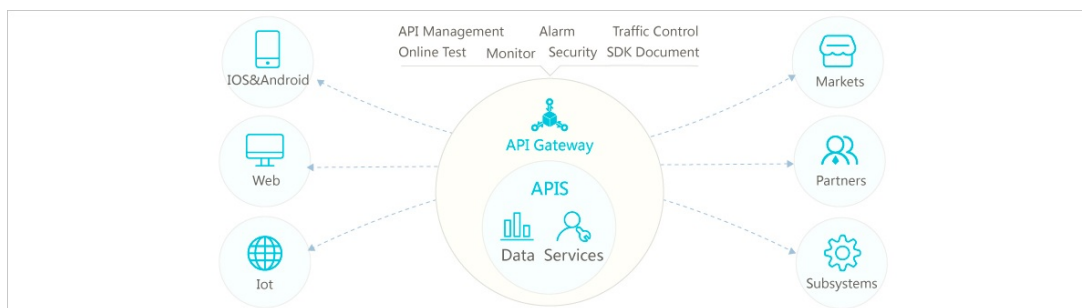
### 2.1. What is API Gateway?

API Gateway provides a comprehensive suite of API hosting services that help you share capabilities, services, and data with partners in the form of APIs.

- API Gateway provides multiple security mechanisms to secure APIs and reduce the risks arising from open APIs. These mechanisms include protection against replay attacks, request encryption, identity authentication, permission management, and throttling.
- API Gateway provides API lifecycle management that allows you to define, publish, and unpublish APIs. This improves API management and iteration efficiency.

API Gateway allows enterprises to reuse and share their capabilities with each other so that they can focus on their core business.

API Gateway



### 2.2. Log on to the API Gateway console


This topic describes how to log on to the API Gateway console.

#### Prerequisites

- The URL of the Apsara Uni-manager Management Console is obtained from the deployment personnel before you log on to the Apsara Uni-manager Management Console.
- A browser is available. We recommend that you use the Google Chrome browser.

#### Procedure

1. In the address bar, enter the URL of the Apsara Uni-manager Management Console. Press the Enter key.
2. Enter your username and password.  
Obtain the username and password that you can use to log on to the console from the operations administrator.

 **Note** When you log on to the Apsara Uni-manager Management Console for the first time, you must change the password of your username. Your password must meet complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:

- Uppercase or lowercase letters
- Digits
- Special characters, which include ! @ # \$ %

3. Click **Login**.

4. In the top navigation bar, choose **Products > Application Services > API Gateway**.

## 2.3. Quick start

### 2.3.1. Create an API with HTTP as the backend service

This topic describes how to create and publish an API with HTTP as the backend service in API Gateway. This topic also describes how to call the API by using the AppKey and AppSecret pair of an app, with Security Certification set to Alibaba Cloud APP. You need to perform the following operations in sequence: **create an API group, create an API, create an app and an API authorization, debug the API, and call the API.**

#### Create an API group

APIs are managed in API groups. Before you create an API, you must create an API group.

1. **Create a group:** In the left-side navigation pane of the API Gateway console, choose **Publish APIs > API Groups**. On the Group List page, click **Create Group** in the upper-right corner. On the Create Group page, specify **Organization**, **Resource Set**, and **Region**, set **Name** to testAppkeyGroup, and click **Submit**. After you specify Organization, Instance is automatically set to Shared Instance(Classic Network).

## Create Group

The screenshot shows the 'Create Group' form. It has two main sections: 'Region' and 'Basic Settings'. In the 'Region' section, there are three fields: '\*Organization:' with a dropdown menu showing 'dstest', '\*Resource Set:' with a dropdown menu showing 'ResourceSet(dstest)', and '\*Region:' with a text input showing 'cn-qingdao-env17-d01'. In the 'Basic Settings' section, there are three fields: '\*Instance:' with a dropdown menu showing 'Shared Instance(Classic Network)', '\*Name:' with a text input showing 'testGroup' and a note below it stating 'The group name must be unique. It must be 4 to 50 characters in length and c', and 'Description:' with a text input showing 'test' and a note below it stating 'The description can be up to 180 characters in length.'

2. **View group information:** In the Submitted message, click Back to Console. On the **Group List** page, click **Refresh** in the upper-right corner. The group you created is displayed. You can click the group name to go to the Group Details page and perform operations such as binding a domain name and modifying basic information. A second-level domain is automatically created for the API group. It can be used in Apsara Stack to call all APIs under this group. In this example, the domain name is used for tests.

## Create an API

In the left-side navigation pane of the API Gateway console, choose **Publish APIs > APIs**. On the API List page, click **Create API** in the upper-right corner. On the Create API page, perform the following steps:

1. **Specify basic information.** In this step, specify basic information, including Group, API Name, Security Certification, and Description. In this example, set Group to testAppkeyGroup, Security Certification to Alibaba Cloud APP, and AppCode Certification to Disable AppCode authentication, set other parameters as required, and click Next.

Create API [Back to API list](#)

Basic Information Define API Request Define API Backend Service

Name And Description

Group testGroup

API Name testAPI3

Security Certification Alibaba Cloud APP

AppCode Certification Disable AppCode authentication

Signature Method HmacSHA256

API Options ☐ Force Nonce Check (Anti Replay by X-Ca-Nonce)

Description test

Next

2. **Define an API request.** In this step, define how a client, such as a browser, a mobile app, or a business system, sends a request for the API. The parameters that need to be specified in this step include Request Type, Protocol, Request Path, HTTP Method, Request Mode, and those in the Input Parameter Definition section. Then click Next. In this example, enter /web/cloudapi in the Request Path field and configure a path parameter, a query parameter, and a header parameter in the Input Parameter Definition section.

Basic Information Define API Request Define API Backend Service Define Response

Basic Request Definition

Request Type ☒ COMMON ☐ REGISTER(WEBSOCKET) ☐ UNREGISTER(WEBSOCKET) ☐ NOTIFY(WEBSOCKET)

Protocol ☒ HTTP ☐ HTTPS ☐ WEBSOCKET

Custom Domain Name Bind domain name to the group

Subdomain Name 976e79c0d0164eac8152956f414ea063.apigateway.inter.env17e.shuguang.com

Request Path /api/test/{type} ☐ Match All Child Paths

The request path must contain the Parameter Path in the request parameter within brackets []. For example: /getUserInfo/{userId}

HTTP Method GET

Request Mode Request Parameter Mapping(Filter Unknown Parameter)

All request parameters must have unique names, including the dynamic parameters in the path, headers parameters, query parameters, body parameters (form parameters).

Input Parameter Definition

Order	Param Name	Param Location	Type	Required	Default Value	Example	Description	Operation
1	type	Parameter Path	String	<input checked="" type="checkbox"/>				<a href="#">More</a> <a href="#">Remove</a>
2	headerparam	Head	String	<input type="checkbox"/>				<a href="#">More</a> <a href="#">Remove</a>
3	queryparam	Query	String	<input type="checkbox"/>				<a href="#">More</a> <a href="#">Remove</a>

[+ Add](#)

Prev Next

3. **Specify API backend service information.** In this step, configure a backend service type and a backend service address of the API and the mappings between request and response parameters. In this example, set Backend Service Type to HTTP(s) Service and Backend Service Address to an address that you can use to access API Gateway. For information about other backend service types, see API Gateway documentation. Set other parameters such as Backend Request Path as

prompted, and click Next.

Basic Backend Definition

Backend Service Type

HTTP(s) Service

VPC

Mock

Backend Service Address

http://10.152.1.1:8080

A backend service address is the domain name or IP address used by the API gateway to call underlying services, not including the path

Backend Request Path

/web/cloudapi{[type]}

Match All Child Paths

The backend request path must contain the Parameter Path in the backend service parameter within brackets []. For example: /getUserInfo/{userId}

HTTP Method

GET

Backend Timeout

10000

ms

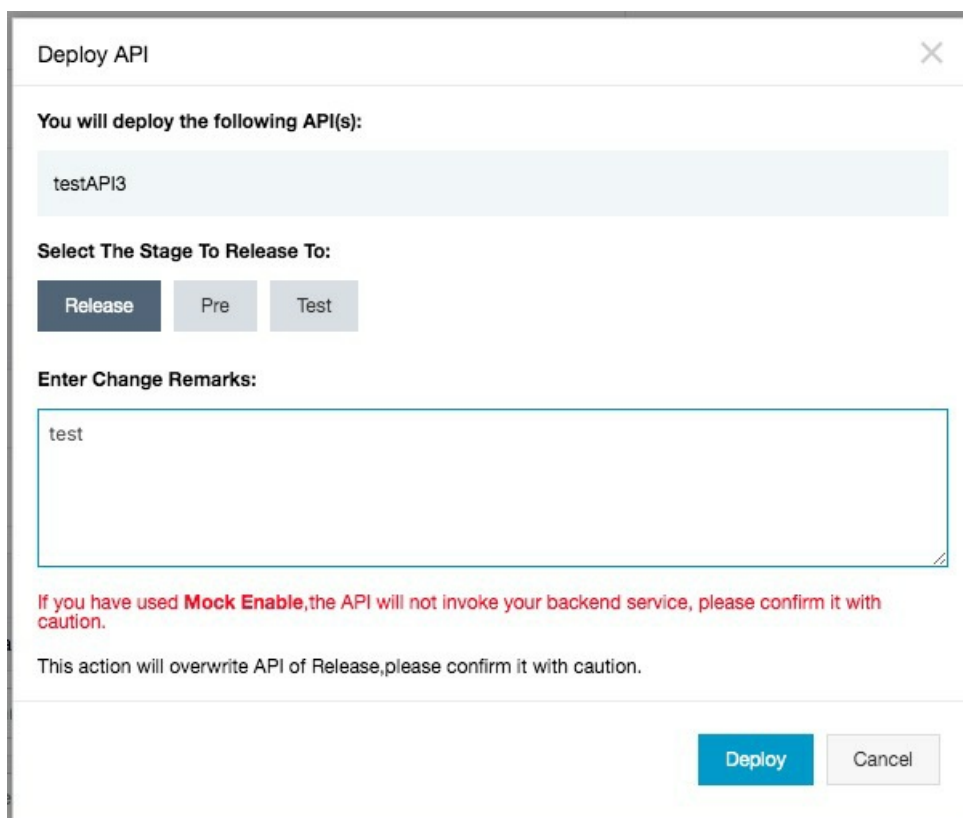
Backend Service Parameter Configuration

Order	Backend Param Name	Backend Param Location	Frontend Param Name	Frontend Param Location	Frontend Param Type
<div><div>↓</div><div>↑</div></div>	<div>type</div>	<div>Parameter Path</div>	type	Parameter Path	String
<div><div>↓</div><div>↑</div></div>	<div>headerparam</div>	<div>Head</div>	headerparam	Head	String
<div><div>↓</div><div>↑</div></div>	<div>queryparam</div>	<div>Query</div>	queryparam	Query	String

Error Code Definition

Error Code	Error Message	Description	Model	Operation
<div>Create Model For Response</div>				
<div><div>+ Add</div></div>				

4. **Define return results.** In this step, configure response information to generate API documentation. The documentation helps API callers better understand APIs. You can set parameters such as ContentType of Response, Sample of Returned Results, and Sample of Returned Failure. The configurations in this step are not involved in this example. Click **Create**.
5. **Publish the API.** API Gateway provides three environments to which you can publish an API: Release, Pre, and Test. All configurations you perform on an API can take effect only after you publish the API to a required environment. In this example, click Deploy in the message that indicates successful API creation. Alternatively, find the created API on the API List page and click Deploy in the Operation column. In the Deploy API dialog box, set Select The Stage To Release To to Release, specify Enter Change Remarks, and click Deploy.



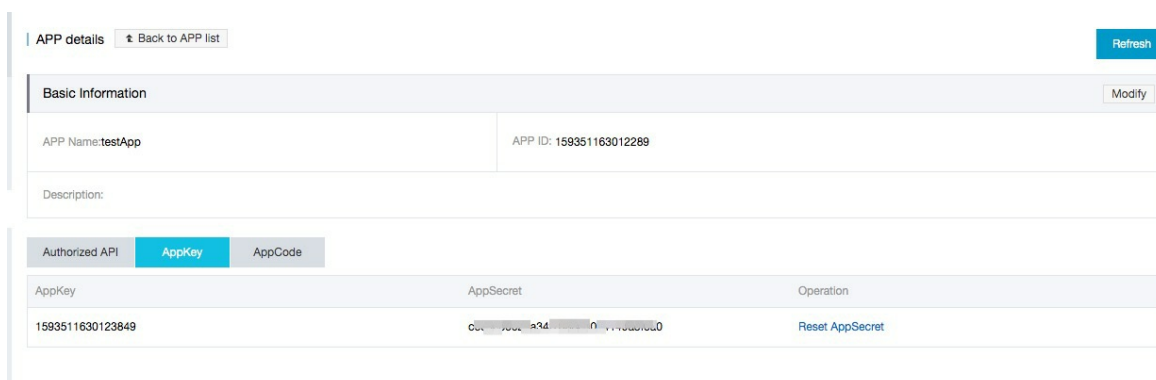
The 'Deploy API' dialog box contains the following elements:

- Deploy API** (Title bar)
- You will deploy the following API(s):**
  - testAPI3
- Select The Stage To Release To:**
  - Release (Selected)
  - Pre
  - Test
- Enter Change Remarks:**
  - test
- Warning messages:**
  - If you have used **Mock Enable**, the API will not invoke your backend service, please confirm it with caution.
  - This action will overwrite API of Release, please confirm it with caution.
- Buttons:** Deploy, Cancel

## Create an app and an API authorization.

Apps are the identities that you use to call APIs. In Step 1 of the "Create an API" section, Security Certification is set to Alibaba Cloud APP. Therefore, after you publish the API, you must create an app and authorize the app to call the API.

1. **Create an app:** In the left-side navigation pane of the API Gateway console, click **Consume APIs** and then **APPs**. On the APP List page, click **Create APP** in the upper-right corner to create an app. Then click the name of the created app to go to the APP details page, as shown in the following figure. Two authentication modes are provided: an AppKey and AppSecret pair and AppCode. Each app has an AppKey and AppSecret pair. It works in a way similar to an account and password pair. When you call an API, you must pass in the AppKey as an input parameter. AppSecret is used to calculate the signature string. API Gateway authenticates the key pair to verify your identity.



The 'APP details' page shows the following information:

- APP details** (Header) | [Back to APP list](#) | [Refresh](#)
- Basic Information** (Tab) | [Modify](#)
  - APP Name: testApp | APP ID: 159351163012289
  - Description:
- Authorized API** (Tab) | **AppKey** (Selected) | AppCode


AppKey	AppSecret	Operation
1593511630123849	00000000000000000000000000000000	<a href="#">Reset AppSecret</a>

2. **Authorize an API:** On the **API List** page, find the created API and click **Authorize** in the Operation column. In the Authorize dialog box, set **Select The Stage For Authorization** to the environment in which you published the API. In this example, set this parameter to **Release**. Click **Search** to search for the created app and click **+Add** in the Operation column to add this app to

the pane on the right. Then click **OK**. A success message is displayed.

**Authorize**

You will authorize the following API(s):

testAPI3

Select The Stage For Authorization: **Release** Pre Test

Authorization Valid Time: ☐ Short-term to  ☒ Long-term

Please note: If there is an authorization relationship between the API and the application, this operation will overwrite the original authorization.

Select The APP For Authorization:

My APP  Enter APP Name **Search**

<input type="checkbox"/>	APP ID	APP Name	Operation
<input type="checkbox"/>	159351163012289	testApp	<b>+ Add</b>
<input type="checkbox"/>	159006778589959	test	<b>+ Add</b>

**Add Selected** 2 entries in total < 1 >

(1) APPs have added

testApp **X Remove**

**OK** Cancel

## Debug an API

API Gateway supports online debugging. We recommend that you use this feature to check whether an API is correctly configured before you call this API at the client side.

1. In the left-side navigation pane of the API Gateway console, click **Consume APIs** and then **APPs**. On the APP List page, click the name of the app that has been authorized to call the created API. On the APP details page, click **Authorized API**, find the target API, and click **Debug API** in the Operation column. On the API debugging page, if you have defined input parameters for this API, you can enter different values for the input parameters to check whether the API is correctly configured.

testAPI3 - Debug API

Request Parameters

API Domain Name

HTTP://976e79c0d0164eac8152956f414ea063-cn-qing

Http Method:GET

Path Format: /api/test/{type}

Path

type = test

Headers

headerparam = sf

Query

queryparam = asdfsdf

Certificate

Authorization Type: Use AppSecret

Stage = RELEASE

AppKey = 1593511630123849

AppSecret = .....

Send Request

Debug Information

Request:

Url: http://976e79c0d0164eac8152956f414ea063.apigateway.inter.env17e.shuguang.com/api/test?queryparam=asdfsdf

Header: {"X-Ca-Timestamp":"1593511950236","gateway\_channel":"http","X-Ca-Key":"1593511630123849","x-ca-nonce":"95b483f5-b90a-417a-8e7c-0e3b286b9e0b","X-Ca-Request-Mode":"DEBUG","X-Ca-Stage":"RELEASE","Host":"976e79c0d0164eac8152956f414ea063.apigateway.inter.env17e.shuguang.com","X-Ca-Signature":"OBsajebWoiel1EUL0x4bLXAFZ63n38l40pD/bWssJyM=","headerparam":"sf","Content-Type":"application/json; charset=utf-8","X-Ca-Signature-Headers":"X-Ca-Timestamp,X-Ca-Key,X-Ca-Request-Mode,X-Ca-Stage"}

Response:

200

Date: Tue, 30 Jun 2020 10:12:30 GMT

Content-Type: application/octet-stream

Content-Length: 618

Connection: keep-alive

Keep-Alive: timeout=25

X-Ca-Request-Id: 4C95EE2A-B2F9-42BF-BA60-37ED6AFE4466

Content-Disposition: attachment; filename=ApiResponseForInnerDomain

{

"Body":",

"Headers":{

"content-length":"0",

"x-forwarded-proto":"http",

"host":"10.17.35.190:8080",

"x-ca-request-id":"4C95EE2A-B2F9-42BF-BA60-37ED6AFE4466",

"content-type":"application/x-www-form-urlencoded; charset=UTF-8",

"connection":"Keep-Alive",

"headerparam":"sf",

"x-forwarded-for":"10.17.50.220",

"user-agent":"Apache-HttpClient/4.5.6 (java/1.8.0\_102)",

"via":"163b308fc121472ca41eaf413c0470e1"

},

"Method":"GET",

"Params":{

"queryparam":"asdfsdf"

},

"Path":"/web/cloudapi/test",

Only the APIs that are published to a required environment and can be called by authorized apps are displayed after you click **Authorized API**.

## Call an API

After you debug and publish an API to a Release environment, you can use SDKs for API Gateway to call the API in your business system.

1. In the left-side navigation pane of the API Gateway console, click **Consume APIs** and then **Authorized APIs SDK**. On the Authorized APIs SDK Auto-Generation page, find the target app and click the required programming language in the Authorized APIs SDK Auto-Generation column to download the relevant SDK package. The SDK package contains the API documentation and the SDK for the created API. For information about how to use the SDK, see the Readme file in the SDK package. Only the SDKs for APIs that are published to a Release environment are supported.

## 2.4. Call an API

### 2.4.1. Manage applications

#### 2.4.1.1. Create an app

Apps are the identities that you use to call APIs. You can own multiple apps. Your apps can be authorized to call different APIs based on your business requirements. User accounts cannot be authorized to call APIs. In the API Gateway console, you can create, modify, or delete apps, view the details of apps, manage key pairs, and view the APIs that can be called by authorized apps.

Each app has an **AppKey** and **AppSecret** pair. It works in a way similar to an account and password pair. When you call an API, you must pass in the **AppKey** as an input parameter. **AppSecret** is used to calculate the signature string. API Gateway authenticates the key pair to verify your identity. An app must be authorized to call an API. Both authorization and authentication are intended for apps.

You can create apps on the **APP List** page in the API Gateway console.

## Procedure

1. [Log on to the API Gateway console.](#)
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, click **Create APP** in the upper-right corner. The app name must be globally unique. It must be 4 to 26 characters in length and can contain only letters, digits, and underscores (\_). It must start with a letter.  
After an app is created, the system automatically assigns an **AppKey** and **AppSecret** pair to the app. You must use the **AppSecret** to calculate a signature string. When you call an API, you must include the signature string in the request. API Gateway verifies your identity based on the signature string.  
On the **APP List** page, click the app name to go to the APP details page that displays the **AppKey** and **AppSecret** information. If the key pair is missing, you can reset it.

### 2.4.1.2. View app details

You can view details of created apps.

## Procedure

1. [Log on to the API Gateway console.](#)
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, click the name of the app that you want to view. On the APP details page, you can view basic app information. You can also click **AppKey** or **Authorized API** to view key pair information and APIs that can be called by authorized apps.

### 2.4.1.3. Edit an app

You can edit a created app.

## Procedure

1. [Log on to the API Gateway console.](#)
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, find the target app and click **Edit** in the Operation column.
4. In the Modify APP dialog box, modify app information and click **OK**.

### 2.4.1.4. Delete an app

You can delete a created app.

## Procedure

1. [Log on to the API Gateway console.](#)
2. In the left-side navigation pane, click **Consume APIs** and then **APPs**.
3. On the APP List page, find the target app and click **Delete** in the Operation column.
4. In the Confirm Deletion message, click **OK**.

## 2.4.2. View created APIs

You can view created APIs in the API Gateway console.

## Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.

### 2.4.3. Authorize an application

Authorization is the process of authorizing an application to call an API. Your applications must be authorized before they can call APIs.

You must provide your application IDs to the API provider for authorization. After authorization, you can view the APIs that your applications have been authorized to call in the API Gateway console.

The APIs that your applications have been authorized to call are displayed in the **Callable APIs** section on the application details page.

After the API provider authorizes your applications to call APIs, you do not need to and cannot authorize your applications.

### 2.4.4. Encrypt a signature

When you call an API, you must construct a signature string and add the calculated signature string to the request header. API Gateway uses symmetric encryption to verify the identity of the request sender.

- Add the calculated signature string to the request header.
- Organize the request parameters into a string-to-sign based on [Request signatures](#). Then, use the algorithm provided in the SDK sample to calculate the signature. The result is the calculated signature string.
- Both HTTP and HTTPS requests must be signed.

For more information about how to construct a string-to-sign, see [Request signatures](#). Replace the AppKey and AppSecret in the SDK sample with your own AppKey and AppSecret. Then, construct a string-to-sign based on Request signatures. After creating the string-to-sign, you can use it to initiate a request.

### 2.4.5. Request signatures

#### Endpoint

- Each API belongs to an API group, and each API group has a unique endpoint. An endpoint is an independent domain name that is bound to an API group by the API provider. API Gateway uses endpoints to locate API groups.
- An endpoint must be in the *www.[Independent domain name].com/[Path]?[HTTPMethod]* format.
- API Gateway locates a unique API group by endpoint, and locates a unique API in the group through the combination of Path and HTTPMethod.
- After you purchase an API, you can obtain the API documentation from the **Purchased APIs** list in the API Gateway console. If you have not purchased an API, you must obtain authorization from the API provider for your applications to call the API. After authorization, you can obtain the API documentation from the **Callable APIs** list on the application details page.

#### System-level header parameters

- (Required) X-Ca-Key: AppKey.

- (Required) X-Ca-Signature: the signature string.
- (Optional) X-Ca-Timestamp: the timestamp passed in by the API caller. This value is a UNIX timestamp representing the number of milliseconds that have elapsed since January 1, 1970 00:00:00 UTC. The timestamp is valid for 15 minutes by default.
- (Optional) X-Ca-Nonce: the UUID generated by the API caller. To prevent replay attacks, you must specify both the X-Ca-Nonce header and the X-Ca-Timestamp header.
- (Optional) Content-MD5: When the request body is not a form, you can calculate the MD5 value of the request body. Then, you can send the value to API Gateway for MD5 verification.
- (Optional) X-Ca-Stage: the stage of the API. Valid values: TEST, PRE, and RELEASE. Default value: RELEASE. If the API that you intend to call has not been published to the release environment, you must specify the value of this parameter. Otherwise, a URL error will be reported.

## Signature validation

Construct the signature calculation strings

```
String stringToSign=
HTTPMethod + "\n" +
Accept + "\n" +      // We recommend that you specify the Accept header in the request. If the request header is not set, some HTTP clients will use the default value */*, causing signature verification to fail.
Content-MD5 + "\n"
Content-Type + "\n" +
Date + "\n" + ,
Headers +
Url
```

An HTTP method must be uppercase, such as POST.

If Accept, Content-MD5, Content-Type, and Date are empty, add a line break `\n` after each of them.

If Headers is empty, `\n` is not required.

### Content-MD5

Content-MD5 indicates the MD5 value of the request body. The value is calculated as follows:

```
String content-MD5 = Base64.encodeBase64(MD5(bodyStream.getBytes("UTF-8")));
```

bodyStream indicates a byte array.

### Headers

Headers indicates the string constructed by the keys and values of the header parameters that are used for Headers signature calculation. We recommend that you use the parameters starting with X-Ca and custom header parameters for signature calculation.

#### Notice

The following parameters are not used for Headers signature calculation: X-Ca-Signature, X-Ca-Signature-Headers, Accept, Content-MD5, Content-Type, and Date.

Headers construction method:

Sort the header keys used for Headers signature calculation in **alphabetical order**. Construct the string based on the following rules: If the value of a header parameter is empty, use

`HeaderKey + ":" + "\n"` for signature calculation. The key and colon (:) must be retained.

```
String headers =  
HeaderKey1 + ":" + HeaderValue1 + "\n\" +  
HeaderKey2 + ":" + HeaderValue2 + "\n\" +  
...  
HeaderKeyN + ":" + HeaderValueN + "\n"
```

The keys of the header parameters used for Headers signature calculation must be separated with commas (,), and placed in the request headers. The key is X-Ca-Signature-Headers.

### Url

Url indicates the **Form** parameter in **Path + Query + Body**. For **Query + Form**, sort keys specified by **Key** in alphabetical order and construct the string based on the following rules: If **Query** or **Form** is empty, no question marks `?` are required for `Url = Path`. If **Value** of a parameter is empty, only **Key** is used for signature calculation and an equal sign (=) is not required.

```
String url =  
Path +  
"?" +  
Key1 + "=" + Value1 +  
&" + Key2 + "=" + Value2 +  
...  
&" + KeyN + "=" + ValueN
```



#### Notice

Note: The **Query** parameter or the **Form** parameter may have multiple values specified by **Value**. If both parameters have multiple values, only the first value of each parameter is used for signature calculation.

## Signature calculation

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");  
byte[] keyBytes = secret.getBytes("UTF-8");  
hmacSha256.init(new SecretKeySpec(keyBytes, 0, keyBytes.length, "HmacSHA256"));  
String sign = new String(Base64.encodeBase64(Sha256.doFinal(stringToSign.getBytes("UTF-8")), "UTF-8"));
```

`secret` indicates the AppSecret.

## Signature passing

Add the calculated signature to the request header. The key is X-Ca-Signature.

## Signature troubleshooting

If signature verification fails, API Gateway places the returned `stringToSign` value in the HTTP response header and sends the response to the client. The key is X-Ca-Error-Message. Compare the `stringToSign` value calculated by the client with the one returned by the server.

If the stringToSign values from the client and server are the same, check the AppSecret used for signature calculation.

HTTP headers do not support line breaks. Line breaks in stringToSign values are filtered out. Ignore the line breaks when you make a comparison.

## Signature demo

For detailed demo (Java) of signature calculation, please refer to the API Gateway console.

## 2.4.6. API call examples

You can edit an HTTP or HTTPS request to call an API. The API Gateway console provides API call examples of multiple programming languages for you to test the call.

### Part 1: Request

Request URL

When you call an API over an internal network, the second-level domain of the API group to which this API belongs is used by default. To view a second-level domain, choose **Publish APIs > API Groups** in the left-side navigation pane of the API Gateway console. Click the name of the target group to go to the Group Details page. If this group is bound with an independent domain, you can use this independent domain to initiate an access request.

```
http://e710888d3ccb4638a723ff8d03837095-cn-qingdao.aliapi.com/demo/post
```

Request method

```
POST
```

Request body

```
FormParam1=FormParamValue1&FormParam2=FormParamValue2 //HTTP request body
```

Request header

```
Host: e710888d3ccb4638a723ff8d03837095-cn-qingdao.aliapi.com
Date: Mon, 22 Aug 2016 11:21:04 GMT
User-Agent: Apache-HttpClient/4.1.2 (java 1.6)
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
// The request body type. Set the request body type based on the actual request you want to make.
Accept: application/json
// The response body type. Some APIs can return data in the appropriate format based on the specified response type. We recommend that you manually specify the request header. If the request header is not specified, some HTTP clients will use the default value */*, which causes a signature error.
X-Ca-Request-Mode: debug
// Specifies whether to enable the debug mode. This parameter is not case-sensitive. If it is not specified, the debug mode is disabled. Enable this mode in the API debugging phase.
X-Ca-Version: 1
// The API version number. Currently, all APIs support only version 1. You can leave this request header unspecified. The default version number is 1.
X-Ca-Signature-Headers: X-Ca-Request-Mode,X-Ca-Version,X-Ca-Stage,X-Ca-Key,X-Ca-Timestamp
// The custom request headers involved in signature calculation. The server reads the request headers based on this configuration to sign the request. This configuration does not include the Content-Type, Accept, Content-MD5, and Date request headers, which are already included in the basic signature structure. For more information about the signature, see Request signatures.
X-Ca-Stage: RELEASE
// The stage of the API. Valid values: TEST, PRE, and RELEASE. This parameter is not case-sensitive. The API provider can select the stage to which the API is published. The API can be called only after it is published to the specified stage. Otherwise, the system will prompt that the API cannot be found or that the request URL is invalid.
X-Ca-Key: 60022326
// The AppKey of the request. You must obtain the AppKey in the API Gateway console. Apps can call APIs only after they have been authorized.
X-Ca-Timestamp: 1471864864235
// The request timestamp. This value is a UNIX timestamp that represents the number of milliseconds that have elapsed since January 1, 1970 00:00:00 UTC. The timestamp is valid for 15 minutes by default.
X-Ca-Nonce:b931bc77-645a-4299-b24b-f3669be577ac
// The unique ID of the request. AppKey, API, and Nonce must be unique within the last 15 minutes. To prevent replay attacks, you must specify both the X-Ca-Nonce header and the X-Ca-Timestamp header.
X-Ca-Signature: FJleSrCYPGCU7dMLTG+UD3Bc5Elh3TV3CWHtSKh1Ys=
// The request signature.
CustomHeader: CustomHeaderValue
// The custom request headers. CustomHeaderValue is used as an example. You can configure multiple custom request headers in requests based on the definition of the API that is being called.
```

## Part 2: Response

### Status code

```
400 // The status code of the response. If the value is greater than or equal to 200 but less than 300, the call succeeds. If the value is greater than or equal to 400 but less than 500, a client-side error has occurred. If the value is greater than 500, a server-side error has occurred.
```

### Response header

```

X-Ca-Request-Id: 7AD052CB-EE8B-4DFD-BBAF-EFB340E0A5AF
// The unique ID of the request. When API Gateway receives a request, it generates a request ID and returns the request ID to the client in the X-Ca-Request-Id header. We recommend that you record the request ID in both the client and backend server for troubleshooting and tracking.
X-Ca-Error-Message: Invalid Url
// The error message returned by API Gateway. If a request fails, API Gateway returns the error message to the client in the X-Ca-Error-Message header.
X-Ca-Debug-Info: {"ServiceLatency":0,"TotalLatency":2}
// The message returned only when the debug mode is enabled. The message is used only for reference at the debugging stage.

```

Regardless of whether you call an API by using HTTP or HTTPS, the request must include the signature information. For information about how to calculate and deliver an encrypted signature, see [Request signatures](#).

## 2.5. APIs

### 2.5.1. Manage groups

#### 2.5.1.1. Create an API group

You can create an API group in the API Gateway console.

##### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, click **Create Group** in the upper-right corner.
4. On the Create Group page, specify **Organization**, **Resource Set**, and **Region** in the Region section. Then specify **Name** and **Description** in the Basic Settings section and click **Submit**. The group name must be unique. It must be 4 to 50 characters in length and can contain only letters, digits, and underscores (\_). It must start with a letter.

#### 2.5.1.2. Manage domain names

In Apsara Stack, you can use the second-level domain of a group to directly call an API that belongs to this group. You can also bind your domain name to the group so that you can use your domain name to call APIs that belong to the group.

##### Context

If you want to use your domain name to directly call APIs that belong to a group, you must bind the domain name to the group and add a DNS record to your domain name. The domain name must be resolved to the second-level domain of the group or the IP address that corresponds to the second-level domain.

##### Bind an independent domain

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the target group and click **Bind Domain** in the **Operation** column.

4. In the Bind Domain Name dialog box, specify **Domain Name** and click **OK**.

## Delete an independent domain

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, find the target group and click its name to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Delete Domain** in the Operation column.
5. In the Confirm Deletion message, click **OK**.

### 2.5.1.3. Manage certificates

To use HTTPS on an independent domain, you must upload an SSL certificate.

#### Context

To perform HTTPS API calls, you must use a domain name that supports HTTPS and set Protocol to HTTPS in the Basic Request Definition section when you define an API request.

#### Upload a certificate

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, click the name of the target group to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Create Certificate** in the SSL Certificate column.
5. In the **Create Certificate** dialog box, specify **Certificate Name**, **Certificate Content**, and **Private Key**, and click **OK**.

#### Delete a certificate


1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, click the name of the target group to go to the **Group Details** page.
4. In the **Custom Domain Name** section, click **Delete Certificate** in the Operation column.
5. In the Confirm Deletion message, click **OK**.

### 2.5.1.4. Delete an API group

You can delete a created API group.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the **Group List** page, find the target group and click **Delete** in the Operation column.
4. In the Delete Group message, click **Delete**.

 **Note** Before you delete a group, you must delete APIs that belong to this group.

## 2.5.1.5. Manage environments

To understand environment management, you must be familiar with two concepts: environment and environment variable.

- An environment is an API group configuration. You can configure several environments for a group. APIs that have not been published are considered defined APIs. An API can provide external services only after it is published to an environment.
- Environment variables are environment-specific variables that you can create and manage. For example, you can create an environment variable named `Path` in the Release environment. The value of this variable is `/stage/release`.

When you define an API request, you can set **Backend Service Address** to `http(s):// #Path# . #Path#` indicates a variable named `Path`.

When you publish the API to the Release environment, the value of `#Path#` is `/stage/release`.


When you publish the API to another environment that does not have the environment variable `#Path#`, the variable value cannot be obtained and the API cannot be called.

Environment variables allow backend services to run in different runtime environments. You can access various backend services by configuring the same API definition but different backend service endpoints and paths across different environments. When you use environment variables, consider the following limits:

- Variable names are case-sensitive.
- If you configure a variable in the API definition, you must configure the name and value of the variable for the environment to which the API is published. Otherwise, no value is assigned to the variable and the API cannot be called.

### Create an environment variable

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the target group and click **View Stages** in the Operation column.
4. On the Stage Management page, click **Add Variable** in the upper-right corner. In the Add Variable dialog box, specify *Name* and *Value* and click **Add**.

 **Notice** The variable names for the Release, Pre, and Test environments must be the same. However, the variable values for the three environments can be different. When an API is published to a specified environment, the variable value will be automatically replaced.

### Delete an environment variable

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find the target group and click **View Stages** in the Operation column.
4. On the Stage Management page, select the runtime environment, find the target variable, and click **Delete** in the Operation column.
5. In the Confirm Deletion message, click **OK**.

## 2.5.2. Create an API

### 2.5.2.1. Overview

Creating an API is the process of defining the API in the API Gateway console. When creating an API, you must define the basic information, back-end service information, API request information, and response information of the API.

- API Gateway enables you to configure verification rules for input parameters. API Gateway can be configured to pre-verify and forward API requests that contain valid parameters.
- API Gateway enables you to configure mappings between front-end and back-end parameters. API Gateway can map a front-end parameter at one location to a back-end parameter at a different location. For example, you can configure API Gateway to map a `Query` parameter in an API request to a `Header` parameter in a back-end service request. In this way, you can encapsulate your back-end services into standard API operations.
- API Gateway enables you to configure constant and system parameters. These parameters are not visible to your users. API Gateway can add these parameters to requests based on your business requirements before sending the requests to your back-end services. If you want API Gateway to attach the keyword `apigateway` to each request that API Gateway forwards to your back-end services, you can configure `apigateway` as a constant parameter and specify where it is received.

### 2.5.2.2. Create an API

#### Procedure

1. [Log on to the API Gateway console.](#)
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, click **Create API** in the upper-right corner.
4. Specify basic information of the API and click **Next**.

Parameter	Description
Group	The basic management unit of APIs. Before you create an API, you must create an API group. When you select a group, a region is selected for the API.
API Name	The name of the API to be created.
Security Certification	<p>The authentication mode of API requests. Valid values: <i>Alibaba Cloud APP</i> and <i>No Certification</i>.</p> <ul style="list-style-type: none"><li>◦ <i>Alibaba Cloud APP</i>: This mode requires the requester to pass the app authentication to call an API.</li><li>◦ <i>No Certification</i>: This mode allows all users who know the request definition of an API to initiate a request. API Gateway directly forwards the request to your backend service without the need to verify the identity of a requester.</li></ul>


Parameter	Description
Signature Method	<p>The algorithm that is used to sign API requests. Valid values:</p> <ul style="list-style-type: none"> <li>◦ HmacSHA256</li> <li>◦ HmacSHA1 and HmacSHA256: If you set this parameter to this value, both the algorithms are supported.</li> </ul>
Description	The description of the API.

5. Define an API request. In this step, define how users call your API, with the following parameters specified: Request Type, Protocol, Request Path, HTTP Method, and Request Mode.

Parameter	Description
Request Type	<p>The request type. Only the COMMON request type is supported. Valid values: COMMON, REGISTER(WEBSOCKET), UNREGISTER(WEBSOCKET), and NOTIFY(WEBSOCKET).</p> <ul style="list-style-type: none"> <li>◦ <i>COMMON</i>: indicates common HTTP or HTTPS requests.</li> <li>◦ <i>REGISTER(WEBSOCKET)</i>: indicates the bidirectional control signaling to register devices. It is sent from the client to the server.</li> <li>◦ <i>UNREGISTER(WEBSOCKET)</i>: indicates the bidirectional control signaling to deregister devices. It is sent from the client to the server. After devices are deregistered, server-to-client notifications are no longer received.</li> <li>◦ <i>NOTIFY(WEBSOCKET)</i>: After receiving the registration signaling sent from the client, the backend service records the device ID and sends a server-to-client notification to API Gateway. Then, API Gateway sends the notification to the device. If the device is online, API Gateway sends the server-to-client notification to the device.</li> </ul>
Protocol	The supported protocol. Valid values: <i>HTTP</i> , <i>HTTPS</i> , and <i>WEBSOCKET</i> .
Request Path	The API request path. The request path can be different from the actual backend service path. You must specify a valid and semantically accurate path as the request path. You can configure dynamic parameters in the request path. This requires that you specify path parameters in the request. In addition, the path parameters can be mapped to query and header parameters that are received by the backend service.
HTTP Method	The HTTP request method. Valid values: <i>PUT</i> , <i>GET</i> , <i>POST</i> , <i>PATCH</i> , <i>DELETE</i> , and <i>HEAD</i> .

Parameter	Description
Request Mode	<p>The request mode. Valid values: <i>Request Parameter Mapping(Filter Unknown Parameters)</i>, <i>Request Parameter Mapping(Passthrough Unknown Parameters)</i>, and <i>Request Parameter Passthrough</i>.</p> <ul style="list-style-type: none"> <li>Request Parameter Mapping(Filter Unknown Parameters): You must configure request and response data mappings for query, path, and body form parameters. API Gateway transparently passes only the configured parameters to the backend service. Other parameters are filtered out.</li> <li>Request Parameter Mapping(Passthrough Unknown Parameters): API Gateway maps and verifies only configured request parameters and transparently passes unknown parameters in a request to the backend service.</li> <li>Request Parameter Passthrough: You do not need to configure query and body form parameters, but must configure path parameters in the Input Parameter Definition section. All parameters sent from the client are transparently passed by API Gateway to the backend service.</li> </ul>

- Define request parameters. In this step, define the request parameters of your API. You can specify different request parameters for different parameter paths. You can select Head, Query, Body, or Parameter Path from the Param Location drop-down list. When you configure a dynamic path parameter, you must provide a description of this dynamic parameter in the Input Parameter Definition section. The following data types are supported: String, Int, and Boolean.
  - Note that the names of all parameters must be unique.
  - You can use the shortcut keys in the Order column to adjust the parameter order.
  - To delete a parameter that is no longer required, you can click **Remove** in the Operation column that corresponds to the parameter.
- Configure parameter verification rules. To configure verification rules of a parameter, you can click **More** in the Operation column that corresponds to the parameter. For example, you can specify Max Length and Enumeration. API Gateway pre-verifies requests based on the verification rules. Requests with invalid parameters are not sent to your backend service. This significantly reduces the workload on your backend service.
- Configure the backend service and click **Next**. In this step, define mappings between request and response parameters, and specify the API configurations of your backend service. Backend service configurations include Backend Service Address, Backend Request Path, Backend Timeout, and configurations in the Backend Service Parameter Configuration, Constant Parameter, and System Parameter sections. After receiving a request, API Gateway converts the format of the request into the format that is required by your backend service based on the backend service configuration. Then, API Gateway forwards the request to your backend service.

 **Note** You can configure the following parameters: dynamic path parameters, header parameters, query parameters, body parameters (non-binary), constant parameters, and system parameters. Each parameter name must be globally unique. For example, you cannot specify a header parameter and a query parameter that have the same name.

## i. Specify related parameters in the Basic Backend Definition section.

Parameter	Description
Backend Service Type	<ul style="list-style-type: none"> <li>■ HTTP(s) Service: This option is selected by default. It indicates that API Gateway accesses the backend service over HTTP or HTTPS. If API Gateway can directly communicate with the backend service, select this option.</li> <li>■ VPC: If the backend service is deployed in a VPC, select this option.</li> <li>■ Mock: If you want to simulate expected return results, select this option.</li> </ul>
VPC ID	The ID of the VPC where your backend service is deployed. This parameter is required when Backend Service Type is set to VPC.
Backend Service Address	<p>The host of the backend service.</p> <ul style="list-style-type: none"> <li>■ If Backend Service Type is HTTP(s) Service, set this parameter to a domain name or a value in the <code>http(s)://host:port</code> format.</li> <li>■ If Backend Service Type is VPC, set this parameter to a value in the <code>http://ip:port</code> format.</li> </ul>
Backend Request Path	The actual request path of your API on your backend server. If you want to receive dynamic parameters in the backend path, you must specify the locations and names of the corresponding request parameters to declare parameter mappings.
HTTP Method	The HTTP request method. Valid values: <i>PUT</i> , <i>GET</i> , <i>POST</i> , <i>PATCH</i> , <i>DELETE</i> , and <i>HEAD</i> .
Backend Timeout	The response time for API Gateway to access the backend service after API Gateway receives an API request. The response time starts from the time when API Gateway sends an API request to the backend service and ends at the time when API Gateway receives a response returned by the backend service. The response time cannot exceed 30s. If API Gateway does not receive a response from the backend service within 30s, API Gateway stops accessing the backend service and returns an error message.

## ii. Configure parameters in the Backend Service Parameter Configuration section.

API Gateway can set up mappings between request and response parameters, including name mappings and parameter location mappings. API Gateway can map a path, header, query, or body request parameter to a response parameter at a different location. This way, you can package your backend service into a standardized and professional API form. This part declares the mappings between request and response parameters.

 **Note** The request and response parameters must be globally unique.

iii. Configure constant parameters in the Constant Parameter section.

If you want API Gateway to attach the `apigateway` tag to each request that API Gateway forwards to your backend service, you can configure this tag as a constant parameter. Constant parameters are not visible to your users. After API Gateway receives requests, it automatically adds constant parameters to the specified locations and then forwards the requests to your backend service.

iv. Configure system parameters in the System Parameter section.

By default, API Gateway does not send its system parameters to your backend service. If you require the system parameters, you can configure the related locations and names. The following table lists the system parameters.

Parameter	Description
CaClientIp	The IP address of the client that sends a request.
CaDomain	The domain name from which a request is sent.
CaRequestHandleTime	The time when a request is sent. It must be in GMT.
CaAppId	The ID of the app that sends a request.
CaRequestId	The unique ID of the request.
CaApiName	The name of the API.
CaHttpSchema	The protocol that is used to call an API. The protocol can be HTTP or HTTPS.
CaProxy	The proxy (AliCloudApiGateway).

9. Define responses and click **Create**. In this step, specify `ContentType` of Response, `Sample of Returned Results`, and `Sample of Returned Failure`, and add configurations in the `Error Code Definition` section. API Gateway does not parse responses, but forwards the responses to API requesters.

### 2.5.2.3. Security authentication

The security authentication methods that are supported by API Gateway include Alibaba Cloud applications and none.

- Alibaba cloud applications: An application must be authorized by the API provider to call an API. An API caller must provide an `AppKey` and encrypted signature. Otherwise, the API request validation will fail. For more information about the signature method, see [Encrypt a signature](#).
- None: The API can be called without authorization after it is published. The `AppKey` and encrypted signature are not required when you make an API request.

### 2.5.2.4. Configure a network protocol

HTTPS domain names are not supported in the API Gateway console. To use an HTTPS domain name, you can call the API operations of API Gateway.

To configure a network protocol, perform the following operations: Find the target API on the API List page in the API Gateway console, and click **Manage** in the Operation column. On the **API Definition** page, click **Edit** in the upper-right corner. On the page that appears, specify Protocol in the **Define API Request** step.

Valid values of Protocol:

- HTTP
- HTTPS
- WEBSOCKET

### 2.5.2.5. Configure a request body

You can configure a request body when the HTTP method is POST, PUT, or PATCH. You can use the following methods to configure the request body. The methods are mutually exclusive.

- **Form-based request body:** Add a request parameter in the Input Parameter Definition section of the Define API Request step on the Create API page, and select **Body** from the Param Location drop-down list. The configured request body can only be used to transmit form data.
- **Non-form-based request body:** If the body content to be transmitted is in the JSON or XML format, select **Non-Form data**, such as JSON, Binary data in the Request Body section of the Define API Request step on the Create API page. The size of a request body cannot exceed 8 MB.

### 2.5.2.6. Configure an API in Mock mode

In most cases, business partners can work in combination to develop a project. The project development process is hindered due to the interdependence among business partners. Misunderstandings can also arise and affect the development progress or even cause severe delays to the project. The Mock mode is used to simulate the predetermined API responses in the project development process. This reduces misunderstandings and improves development efficiency.

API Gateway provides a simple configuration process of an API in Mock mode.

#### Configure an API in Mock mode

Log on to the API Gateway console. In the left-side navigation pane, choose **Publish APIs > APIs**. On the API List page, find the target API and click **Manage** in the Operation column. On the API Definition page, click **Edit** in the upper-right corner.

On the page that appears, configure the Mock mode in the **Define API Backend Service** step.

1. Set **Backend Service Type** to **Mock**.
2. Specify Mock Result in the Mock Configuration section.  
Enter your responses as the Mock-based response body. Responses can be in the JSON, XML, or text format. Example:

```
{
  "result": {
    "title": "Mock test for API Gateway",
  }
}
```

Save the settings and then publish the API to the Test or Release environment for testing.

3. Specify **HTTP Status Code** based on HTTP status code specifications. Enter 200 to indicate a

successful API request.

4. Specify **Mock Header**. You can click +Add Item to add a Mock response header based on your business requirements.


### 2.5.2.7. Return the Content-Type header

The value of the Content-Type header is only used to generate API documentation. It does not affect responses returned by the back-end service. The Content-Type header is returned by the back-end service.

## 2.5.3. API management

### 2.5.3.1. View and modify an API

You can view and modify an API as required.

 **Note** If you modify an API that is published, the modifications are not immediately applied. You must republish the modified API to synchronize the changes to the Release environment.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API.
  - Click **Manage** in the Operation column. On the API Definition page, you can view the information of the API.
  - Click **Edit** in the upper-right corner to edit the API as required.

The procedure of creating an API is similar to that of modifying an API. For more information about how to create an API, see [Create an API](#). If you want to cancel the modifications before the modifications are submitted, click Cancel Edit in the upper-right corner of the edit page.

### 2.5.3.2. Publish an API

After you create an API, you must publish the API to the Test, Pre, or Release environment before it can be called.

- When you use a second-level or independent domain to access an API that is published to a specified environment, you must specify the environment in the request header.
- If you publish an API that already has a running version in the Test or Release environment, the running version is automatically overwritten by the new version within 15s. However, all historical versions and definitions are recorded. This allows you to roll the API back to an earlier version.
- You can unpublish an API in the Test or Release environment. The plugin binding relationship or the app authorization relationship is retained after you unpublish an API. These relationships take effect again if the API is republished. You can also perform related operations to remove the authorization or unbind a required plugin.

#### Step 1: Publish the API

After the test is complete, you can publish the API.

API Gateway allows you to manage different versions of APIs in the Test or Release environment. You can publish or unpublish the API, and switch the version of the API. The version switch takes effect in real time.

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Deploy** in the Operation column.
4. In the Deploy API dialog box, specify Enter Change Remarks and click **Deploy**.

## Step 2: Test the API

To simulate API requests, you can create an app and authorize the app to call your API.

You can compile code based on actual scenarios, or use the SDK samples provided by API Gateway to call your API.

You can publish the API to the Test or Release environment. If no independent domain is bound to the group to which the API belongs, you can test or call the API by using a second-level domain. When you make an API request, set the X-Ca-Stage header to TEST, PRE, or RELEASE to specify the environment of the API. If you do not specify the header, the API will be invoked to the Release environment.


### 2.5.3.3. Authorize an app

You must authorize an app before it can call an API. After you publish an API to the Release environment, you must authorize apps to call the API. You can grant or revoke the authorization of an app to call an API. API Gateway verifies the authorization relationship.

#### Note

- You can authorize one or more apps to call one or more APIs.
- If an API is published to both the Test and Release environments and an app is authorized to call the API in the Test environment, the app can call only the API in the Test environment.
- You can find an app based on its ID.
- If you want to revoke the authorization of an app to call an API, go to the Authorization page of the API. Then select the required app and click Revoke Authorization in the lower-left corner.

An app indicates the identity of a requester. Before testing or calling an API, you or your users must create an app that is used as the identity of a requester. Then, you must authorize the app to call the API.

 **Note** Authorizations are environment-specific. If you want to use an app to call an API in both the Test and Release environments, you must authorize the app in both environments. Otherwise, errors may occur due to the inconsistency between the authorized environment and the requested environment.

## Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Authorize** in the Operation column.

4. In the Authorize dialog box, specify **Select The Stage For Authorization** and **Select The APP For Authorization**. **My APP** is automatically selected from the drop-down list on the left. Click **Search**. Apps created under your account appear. If you want to authorize an app created under a different account, select **APP ID** from the drop-down list on the left, enter the app ID in the search bar, and click **Search**. To view the ID of an app, click **Consume APIs** and then **APPs** in the left-side navigation pane. On the APP List page, click the name of the target app to go to the APP details page.
5. Select an app to be authorized and click **+Add** in the Operation column to add this app to the right pane. Alternatively, you can select multiple apps to be authorized at a time and click **Add Selected** in the lower-left corner of the page to add these apps to the right pane.
6. Click **OK** to complete the authorization.
7. Click **Manage** in the Operation column that corresponds to the target API. On the API Definition page, click **Authorization** in the left-side navigation pane to view the authorized apps.

### 2.5.3.4. Revoke an authorization

You can revoke the authorization of an app to call an API.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, click the name of the target API for which you want to revoke the authorization. On the API Definition page, click **Authorization** in the left-side navigation pane.
4. Select target apps and click **Revoke Authorization** in the lower-left corner.
5. In the Confirm authorization revocation message, click **OK**.

### 2.5.3.5. Unpublish an API

You can unpublish an API.

You can unpublish an API in the Test or Release environment. The binding or authorization relationships of policies, keys, and apps are retained after you unpublish an API. These relationships will take effect again if the API is republished. To remove these relationships, you must delete the API.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Undeploy** in the Operation column.
4. In the Undeploy api message, click **OK**.

### 2.5.3.6. View the version history of an API

You can view the version history of an API, including the version number, description, environment, publish time, and specific definition of each version.

#### Procedure

1. [Log on to the API Gateway console](#).

2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Manage** in the Operation column to go to the API Definition page.
4. Click **Deployment History** in the left-side navigation pane. You can view the version history of this API.
5. On the Deployment History page, find the target version and click **View** in the Operation column.

### 2.5.3.7. Change the version of an API

When you view the version history of an API, you can select a different version to switch the API to this version. The selected version then replaces the previous version and takes effect in the specified environment.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > APIs**.
3. On the API List page, find the target API and click **Manage** in the Operation column to go to the API Definition page.
4. Click **Deployment History** in the left-side navigation pane.
5. Find the target version and click **Switch to this version** in the Operation column.
6. In the API Version Switch dialog box, enter the description and click **Switch**.

## 2.5.4. Plugin management

### 2.5.4.1. Use parameters and conditional expressions

In an `access control plugin`, `throttling plugin`, `backend routing plugin`, or `error code mapping plugin`, you can obtain parameters from requests, responses, and system context. Then, you can use conditional expressions to evaluate these parameters. This topic describes how to define parameters and write conditional expressions.

#### 1. Define parameters

1. Definition method Before you use a conditional expression, you must explicitly define all the parameters required in this conditional expression in the parameters field. Example:

```
---
parameters:
  method: "Method"
  appId: "System:CaAppId"
  action: "Query:action"
  userId: "Token:UserId"
```

The parameters specified in `parameters` are key-value pairs of the string type.

- `key` indicates the name of a variable to be used in a conditional expression. The name must be unique and must conform to the following regular expression: `[a-zA-Z_][a-zA-Z0-9]+`.
- `value` indicates the location of a parameter. It is specified in the `{location}` or `{location}:{name}` format.

- **location** indicates the location of a parameter. For more information, see the following table.
- **name** indicates the name of a parameter, which is used to locate the parameter at a specific location. For example, **Query:q1** indicates the first value of the query string named q1.

2. Parameter locations Before you use a conditional expression, you must define the parameters that are required in this conditional expression. The following table describes parameters at specific locations that can be used by various plugins.

Location	Included in	Description
Method	Request	The HTTP request method, in uppercase, such as GET or POST.
Path	Request	The complete HTTP request path, such as <code>/path/to/query</code> .
StatusCode	Response	The HTTP status code in a backend response, such as 200 or 400.
ErrorCode	Response	<a href="#">Error codes</a> .
Header	Request/Response	Use <code>Header:{Name}</code> to obtain the first value of the HTTP header that is specified by <code>{Name}</code> .
Query	Request	Use <code>Query:{Name}</code> to obtain the first value of the query string that is specified by <code>{Name}</code> .
Form	Request	Use <code>Form:{Name}</code> to obtain the first value of the form that is specified by <code>{Name}</code> .
Host	Request	Use <code>Host:{Name}</code> to obtain the template parameters of the matched wildcard domain names.
Parameter	Request	Use <code>Parameter:{Name}</code> to obtain the first value of the custom API parameter that is specified by <code>{Name}</code> .
BodyJsonField	Response	Use <code>BodyJsonField:{JPath}</code> to obtain the JSON string in the body of an API request or a backend response in JSONPath mode.

Location	Included in	Description
System	Request/Response	Use <code>System:{Name}</code> to obtain the value of the system parameter that is specified by <code>{Name}</code> .
Token	Request/Response	If JWT is used for authentication, use <code>Token:{Name}</code> to obtain the value of the parameter that is specified by <code>{Name}</code> in a token.

#### Rules for use:

- You can use the following plugins at the request phase: `access control plugin`, `throttling plugin`, and `backend routing plugin`. These plugins support only the parameters at the following locations: `Method`, `Path`, `Header`, `Query`, `Form`, `Parameter`, `System`, and `Token`.
  - You can also use the `error code mapping plugin` at the response phase. This plugin supports only the parameters at the following locations: `StatusCode`, `ErrorCode`, `Header`, `BodyJsonField`, `System`, and `Token`.
  - Parameters at the `Method`, `Path`, `StatusCode`, and `ErrorCode` locations are defined in the `{location}` format.
  - If you use parameters at the `Header` location in a plugin at the request phase, headers from client requests are read. If you use these parameters at the response phase, headers from backend responses are read.
  - Parameters at the `Parameter` location are available only for plugins at the request phase. A `frontend parameter`, instead of a `backend parameter`, is used to search for the parameter with the same name in the API definition. If no parameter with the same name exists, a null value is returned.
  - A complete request path is returned from `Path`. If you require a parameter at the `Path` location, use the corresponding parameter at the `Parameter` location.
  - Parameters at the `BodyJsonField` location are available only for the `error code mapping plugin`. Obtain the JSON string in the body of a backend response in `JSONPath` mode. For more information, see Usage notes of `JSONPath`.
  - If `JWT` is used for authentication, use `Token:{ClaimName}` to obtain the value of the parameter specified by `{ClaimName}` in a token. For more information, see the plugin documentation.
3. Usage notes of `JSONPath`  
`JSONPath` is available only for the `error code mapping plugin` at the `BodyJsonField` location. It is used to extract the `JSON` string in the body of a backend response. For more information about `JSONPath`, see the `JSONPath` overview documentation.  
 Example: When you use the expression `code:"BodyJsonField:$.result_code"`, you can obtain the value of `result_code` from the following body. `code:ok` is parsed from the following body.

```
{ "result_code": "ok", "message": ... }
```

#### 4. System parameters

Parameter	Description	Value
CaClientIp	The IP address of the request client.	Example value: 37.78.3.3.
CaDomain	The full domain name in a request, with a Host header.	Example value: api.foo.com.
CaAppId	The ID of the application that sends the request.	Example value: 49382332.
CaAppKey	The key of the application that sends the request.	Example value: 12983883923.
CaRequestId	The unique ID of the request generated by API Gateway.	Example value: CCE4DEE6-26EF-46CB-B5EB-327A9FE20ED1.
CaApiName	The API name.	Example value: TestAPI.
CaHttpSchema	The protocol used by the client to call operations.	Valid values: http, https, and ws.
CaClientUa	The UserAgent header of the client.	Used to transparently pass values uploaded by the client.
CaStage	The running environment of API Gateway.	Valid values: TEST, PRE, and RELEASE.

## 2. Write conditional expressions

You can use conditional expressions in plugins or other scenarios to evaluate parameters in a wide variety of scenarios.

### 1. Basic syntax

- Conditional expressions are similar to SQL statements. Example: `$A > 100 and '$B = 'B'`.
- An expression is in the following format: `{Parameter}{Operator}{Parameter}`. In the preceding example, you can specify a **variable** or a **constant** for `$A > 100`.
- A **variable** starts with \$ and references a parameter defined in the context. For example, `q1:"Query:q1"` is defined in **parameters**. You can use the variable `$q1` in your expression. The value of this variable is the value of the `q1` query parameter in the request.
- A **constant** can be a **string**, **number**, or **Boolean value**. Examples: `"Hello", 'foo', 100, -1, 0.1, and true`. For more information, see **Value types and evaluation rules**.
- The following **operators** are supported:
  - `=` and `==`: equal to.
  - `<>` and `!=`: not equal to.
  - `>`, `>=`, `<`, and `<=`: comparison.
  - `like` and `!like`: check whether a specific string matches a specified pattern. The percent sign `%` is used as a wildcard in the evaluation. Example: `$Query like 'Prefix%'`.
  - `in_cidr` and `!in_cidr`: specify the mask of an IP address. Example: `$ClientIp in_cidr '47.89.0.0/24'`.

- You can use `null` to check whether a parameter is empty. Example: `$A == null` or `$A != null`.
- You can use the operators `and`, `or`, and `xor` to combine different expressions in a right-to-left order by default.
- You can use parentheses `()` to specify the priority of conditional expressions.
- You can use `!()` to perform the logical negation operation on the enclosed expression. For example, the result of `!(1=1)` is false.
- The following built-in functions are used for evaluation in some special scenarios:
  - `Random()`: generates a parameter of the floating-point number type. The parameter value ranges from 0 to 1. This parameter is used in scenarios where random input is required, such as blue-green release.
  - `Timestamp()`: returns a UNIX timestamp representing the number of milliseconds that have elapsed since the epoch time January 1, 1970, 00:00:00 UTC.
  - `TimeOfDay()`: returns the number of milliseconds from the current time to 00:00 of the current day in GMT.

## 2. Value types and evaluation rules

- The following value types are supported in expressions:
  - `STRING`: The value can be a string. Single quotation marks ( `' '` ) or double quotation marks ( `" "` ) can be used to enclose a string. Examples: `"Hello"` and `'Hello'`.
  - `NUMBER`: The value can be an integer or a floating-point number. Examples: `1001`, `-1`, `0.1`, and `-100.0`.
  - `BOOLEAN`: The value can be a Boolean value. Valid values: `true` and `false`.
- For the operator types `equal to`, `not equal to`, and `comparison`, the following evaluation rules apply:
  - `STRING` type: uses the string order for evaluation. Examples:
    - `'123' > '10000'`: The result is true.
    - `'A123' > 'A120'`: The result is true.
    - `" < 'a'`: The result is true.
  - `NUMBER` type: uses numerical values for evaluation. Examples:
    - `123 > 1000`: The result is false.
    - `100.0 == 100`: The result is true.
  - `BOOLEAN` type: For Boolean values, true is greater than false. Examples:
    - `true == true`: The result is true.
    - `false == false`: The result is true.
    - `true > false`: The result is true.
- For the operator types `equal to`, `not equal to`, and `comparison`, if the value types before and after an operator are different, the following evaluation rules apply:

- Assume that a value before an operator is of the `STRING` type and that after the operator is of the `NUMBER` type. If the value type before the operator can be changed to `NUMBER`, use numerical values for evaluation. Otherwise, use the string order for evaluation. Examples:
  - `'100' == 100.0` : The result is true.
  - `'-100' > 0` : The result is false.
- Assume that a value before an operator is of the `STRING` type and that after the operator is of the `BOOLEAN` type. If the value type before the operator can be changed to `BOOLEAN` and the value is not case-sensitive, use `BOOLEAN` values for evaluation. Otherwise, except for the evaluation result of `!=`, all the other evaluation results are `false`. Examples:
  - `'True' == true` : The result is true.
  - `'False' == false` : The result is true.
  - `'bad' == false` : The result is false.
  - `'bad' != false` : The result is true. If the value before the operator is not `true` or `false`, only the result for `!=` is true.
  - `'bad' != true` : The result is true.
  - `'0' > false` : The result is false.
  - `'0' <= false` : The result is false.
- Assume that a value before an operator is of the `NUMBER` type and that after the operator is of the `BOOLEAN` type. The result is false.
- The `null` value is used to check whether a parameter is empty. For the operator types `equal to`, `not equal to`, and `comparison`, the following evaluation rules apply:
  - If the `$A` parameter is empty, the result of `$A == null` is true, and the result of `$A != null` is false.
  - If the empty string `"` is not equal to `null`, the result of `" == null` is false, and the result of `" == "` is true.
  - For the `comparison` operator type, if the value on either side of the operator is `null`, the result is false.
- `like` and `!like` operators are used to match the prefix, suffix, and inclusion of a string. The following evaluation rules apply:
  - In an expression, the value after the operator must be a constant of the `STRING` type. Example: `$Path like '/users/%'`.
  - The `'%'` wildcard character in the value after the operator is used to match the prefix, suffix, or inclusion of a string. Examples:
    - Prefix matching: `$Path like '/users/%'` and `$Path !like '/admin/%'`
    - Suffix matching: `$q1 like '%search'` and `$q1 !like '%.do'`
    - Inclusion relation matching: `$ErrorCode like '%400%'` and `$ErrorCode !like '%200%'`
  - If the value type before an operator is not `NUMBER` or `BOOLEAN`, change the type to `STRING` and then perform the evaluation.
  - If the value before an operator is `null`, the result is `false`.
- `in_cidr` and `!in_cidr` operators are used to identify the mask of a CIDR block. The following evaluation rules apply:

- The value after an operator must be a constant of the **STRING** type and must be an IPv4 or IPv6 CIDR block. Examples:
  - `$ClientIP in_cidr '10.0.0.0/8'`
  - `$ClientIP !in_cidr '0:0:0:0:FFFF::/96'`
- If the value type before an operator is **STRING**, the value is considered an IPv4 CIDR block for evaluation.
- If the value type before an operator is **NUMBER** or **BOOLEAN** or the value is empty, the result is **false**.
- The `System:CaClientIp` parameter specifies the IP address of the client, which is used for evaluation.

### 3. Use cases

- The following expression indicates that the probability is less than 5%:

```
Random() < 0.05
```

- The following expression indicates that the requested API is published to the Test environment:

```
parameters:
  stage: "System:CaStage"
```

```
$CaStage='TEST'
```

- The following expression indicates that the custom parameter Username is set to Admin and the source IP address is `47.47.74.0/24`:

```
parameters:
  Username: "Token:UserName"
  ClientIp: "System:CaClientIp"
```

```
$UserName = 'Admin' and $CaClientIp in_cidr '47.47.74.0/24'
```

- The following expression indicates that the AppId parameter is set to 1001, 1098, or 2011, and the protocol that is used by the API request is HTTPS:

```
parameters:
  CaAppId: "System:CaAppId"
  HttpSchema: "System:CaHttpSchema"
```

```
$CaHttpScheme = 'HTTPS' and ($CaAppId = 1001 or $CaAppId = 1098 or $CaAppId = 2011)
```

- The following expression indicates that the JSON string in a body contains result\_code that is not ok when StatusCode in a response is 200:

```
parameters:
  StatusCode: "StatusCode"
  ResultCode: "BodyJsonField:$result_code"
```

```
$StatusCode = 200 and ($ResultCode <> null and $ResultCode <> 'ok')
```

### 4. Limits

- A maximum of 16 parameters can be specified in a plugin.
- A conditional expression can contain a maximum of 512 characters.
- The size of a request or response body specified by BodyJsonField cannot exceed 16 KB. Otherwise, the settings will not take effect.

## 2.5.4.2. Create a plugin

### 2.5.4.2.1. Create an IP address-based access control plugin

IP address-based access control helps API providers configure an IP address whitelist or blacklist for API calls. This topic describes how to create an IP address-based access control plugin.

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**, and set Type to **IP Access Control**. A plugin definition template in the YAML format is automatically loaded in the **Script Configurations** field. Modify template content.

## Create Plug-in

Region

\*Organization:

ebstest

\*Resource Set:

ResourceSet(ebstest)

\*Region:

cn-qingdao-env17-d01

Basic Settings

\*Name:

testIp

The name of the plug-in. The name must be 4 to 50 characters in length, and can contain letters,

\*Type:

IP Access Control

\*Script Configurations:

---  
type: ALLOW    # Ip control type: 'ALLOW' or 'REFUSE'  
items:  
- blocks:        # IP Blocks  
  - 78.11.12.2    # config via ip address v4  
  - 61.3.9.0/24   # config via cidr  
  appld: 219810   # (optional) if config appld, this item will only affected to configured APP  
- blocks:        # IP Blocks  
  - 79.11.12.2   # config via ip address v4

Submit

Parameter	Description
-----------	-------------

Parameter	Description
type	<ul style="list-style-type: none"><li>◦ ALLOW: You can configure a whitelist to allow the API requests that meet specific requirements. The following types of whitelists are supported:<ul style="list-style-type: none"><li>■ You can configure a whitelist that includes only IP addresses. In this case, only API requests from the IP addresses in the whitelist are allowed.</li><li>■ You can configure a whitelist that contains apps and their IP addresses. In this case, each app can send API requests only from its IP addresses in the whitelist.</li></ul></li><li>◦ REFUSE: You can configure an IP address blacklist. API Gateway rejects all API requests from the IP addresses in the blacklist.</li></ul>

Script template of the IP address-based access control plugin

```
---
type: ALLOW    # The type of access control. You can set this parameter to ALLOW to apply a whitelist or to REFUSE to apply a blacklist.
items:
- blocks:      # The IP address segment.
- 78.11.12.2   # Specifies an IP address.
- 61.3.9.0/24  # Specifies a CIDR block.
appId: 219810  # Optional. If you specify this parameter, this IP address-based access control policy applies only to the app specified by this parameter.
- blocks:      # The IP address segment.
- 79.11.12.2   # Specifies an IP address.
```

4. Click **Submit**.

## 2.5.4.2.2. Create a throttling plugin

You can use a throttling plugin to limit the number of API requests. A throttling plugin helps prevent a backend service from being overwhelmed by a large number of API requests. This topic describes how to create a throttling plugin.

### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, and **Region**. Then specify **Name**, set **Type** to **Traffic Control**, and modify configurations in the **Script Configurations** field.

## Create Plug-in

Region

\*Organization: ebstest

\*Resource Set: ResourceSet(ebstest)

\*Region: cn-qingdao-env17-d01

Basic Settings

\*Name: testTraffic

The name of the plug-in. The name must be 4 to 50 characters in length, and can contain letters.

\*Type: Traffic Control

\*Script Configurations:
 

```

---
unit: SECOND      # Traffic control period:
SECOND, MINUTE, HOUR, DAY
apiDefault: 1000  # Total limit for attached API.
userDefault: 30   # (optional) Limit vary by
consumer User, can't larger than total limit.
appDefault: 30    # (optional) Limit vary by
consumer App, can't larger than total limit.
specials:         # (optional) Limit for specific
consumer, support "APP" or "USER"
- type: "APP"     # Vary by 'APP', each APP has a
unique AppID,
policies:
- key: 10123123  # value of AppID, refer to `Web
Console -> Consume APIs -> APPs`
value: 10        # Special limit, can't larger than
total limit
- key: 10123123  # value of AppID, refer to `Web
Console -> Consume APIs -> APPs`
value: 10        # Special limit. can't larger than
          
```

Submit

Parameter	Description
unit	The unit of time. Valid values: SECOND, MINUTE, HOUR, and DAY.
apiDefault	The default API-level throttling threshold. It indicates the maximum number of times that an API bound with a throttling policy can be called within a specific unit of time. This parameter is set based on the backend service capability. This parameter is required.

Parameter	Description
userDefault	The default user-level throttling threshold. It indicates the maximum number of times that each user can call an API that is bound with a throttling policy within a specific unit of time. The user-level throttling threshold cannot be greater than the API-level throttling threshold. This parameter is optional.
appDefault	The default app-level throttling threshold. It indicates the maximum number of times that each app can call an API that is bound with a throttling policy within a specific unit of time. The app-level throttling threshold cannot be greater than the user-level throttling threshold. This parameter is optional.
specials	The special throttling settings. This parameter is optional. You can set throttling thresholds for special apps or users in a throttling policy. After this parameter is specified, the special throttling settings prevail for special apps or users.

#### Script template

```
---
unit: SECOND    # The unit of time. Valid values: SECOND, MINUTE, HOUR, and DAY.
apiDefault: 1000 # The default API-level throttling threshold.
userDefault: 30  # Optional. The default user-level throttling threshold. If you set this threshold to 0, user-level throttling is not performed. The user-level throttling threshold cannot be greater than the API-level throttling threshold.
appDefault: 30   # Optional. The default app-level throttling threshold. If you set this threshold to 0, app-level throttling is not performed. The app-level throttling threshold cannot be greater than the user-level throttling threshold.
specials:        # Optional. The special throttling settings. You can set throttling thresholds for special apps or users in a throttling policy.
  - type: "APP"   # The special throttling type. The value APP indicates that throttling is performed for special apps based on their AppKeys.
    policies:
      - key: 10123123 # The app ID. You can obtain the ID of an app from the app details page. To go to this page, click Consume APIs and then APPs in the left-side navigation pane of the API Gateway console and click the name of the app.
        value: 10     # The special throttling threshold for the app. This threshold cannot be greater than the user-level throttling threshold in the throttling policy.
      - key: 10123121 # The app ID.
        value: 10     # The special throttling threshold for the app. This threshold cannot be greater than the user-level throttling threshold in the throttling policy.
      - type: "USER" # The special throttling type. The value USER indicates that throttling is performed for special Apsara Stack tenant accounts.
        policies:
          - key: 123455 # The ID of an Apsara Stack tenant account. You can move the pointer over the profile picture in the upper-right corner of the Alibaba Cloud Management Console to obtain the ID.
            value: 100  # The special throttling threshold for the Apsara Stack tenant account. This threshold cannot be greater than the API-level throttling threshold in the throttling policy.
```

#### 4. Click Submit .

### 2.5.4.2.3. Create a backend signature plugin

A backend signature plugin is used for signature verification between API Gateway and your backend service. A backend signature is a key-secret pair that you create and issue to API Gateway. It works in a way similar to an account and password pair. When API Gateway sends a request to your backend service, API Gateway uses the backend signature to calculate a signature string and pass it to your backend service. Your backend service obtains the signature string and authenticates API Gateway by using symmetric calculation. Perform the following steps to create a backend signature plugin:

#### Procedure

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**. Set Type to **Backend Signature**.

## Create Plug-in

Region

\*Organization:

ebstest

▼

\*Resource Set:

ResourceSet(ebstest)

▼

\*Region:

cn-qingdao-env17-d01

Basic Settings

\*Name:

testSignature

The name of the plug-in. The name must be 4 to 50 characters in length, and c

\*Type:

Backend Signature

▼

\*Script Configurations:

---  
type: APIGW\_BACKEND  
key: SampleKey  
secret: SampleSecret

Submit

Configure the plugin parameters as required.

4. Click **Submit**.

### 2.5.4.2.4. Create a CORS plugin

This topic describes how to create a cross-origin resource sharing (CORS) plugin. If a resource requests another resource from a different domain or port of a different server, the former resource initiates a cross-domain HTTP request. For security purposes, the browser blocks the request and reports an error message. In this case, you need to use a CORS plugin to troubleshoot the issue.

## Procedure

1. Log on to the API Gateway console.
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**. Set **Type** to **CORS**.

## Create Plug-in

Region

\*Organization: ebstest

\*Resource Set: ResourceSet(ebstest)

\*Region: cn-qingdao-env17-d01

Basic Settings

\*Name: testCors

The name of the plug-in. The name must be 4 to 50 characters in length, and can contain letters, digits, and hyphens.

\*Type: CORS

\*Script Configurations:

```

---
allowOrigins: "api.foo.com"
allowMethods:
"GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH"
allowHeaders: "Authorization,Accept,Accept-
Ranges,Cache-Control,Range,Date,Content-
Type,Content-Length,Content-MD5,User-Agent,X-
Ca-Signature,X-Ca-Signature-Headers,X-Ca-
Signature-Method,X-Ca-Key,X-Ca-Timestamp,X-Ca-
Nonce,X-Ca-Stage,X-Ca-Request-Mode,x-ca-
deviceid"
exposeHeaders: "Content-
MD5,Server,Date,Latency,X-Ca-Request-Id,X-Ca-
Error-Code,X-Ca-Error-Message"
maxAge: 172800
allowCredentials: true

```

Submit

Cross-domain access template

```
---
allowOrigins: api.foo.com,api2.foo.com # The allowed origins. Separate origins with commas (,). Default value: *.
allowMethods: GET,POST,PUT           # The allowed HTTP methods. Separate methods with commas (,).
allowHeaders: X-Ca-RequestId          # The allowed request headers. Separate headers with commas (,).
exposeHeaders: X-RC1,X-RC2           # The headers that can be exposed to the XMLHttpRequest object. Separate headers with commas (,).
allowCredentials: true                # Controls whether cookies are allowed.
maxAge: 172800
```

Configure the plugin parameters as required.

4. Click **Submit**.

## 2.5.4.2.5. Create a backend routing plugin

A backend routing plugin is used to route API requests to different backend services by changing the backend service type, backend service address, backend request path, and response parameters based on request and system parameters in API requests. Backend routing plugins can be used for multi-tenant routing and blue-green release. They can also be used to distinguish between different environments.

### Create a backend routing plugin

1. [Log on to the API Gateway console](#).
2. In the left-side navigation pane, choose **Publish APIs > Plugin**.
3. On the Plugins list page, click **Create Plugin** in the upper-right corner. On the Create plugin page, specify **Organization**, **Resource Set**, **Region**, and **Name**. Set **Type** to **Routing**.

## Create Plug-in

Region

\*Organization: ebstest

\*Resource Set: ResourceSet(ebstest)

\*Region: cn-qingdao-env17-d01

Basic Settings

\*Name: testRouting

The name of the plug-in. The name must be 4 to 50 characters in length, and can

\*Type: Routing

\*Script Configurations:

```

---
routes:
- name: Vip
  condition: "$CaAppId = 123456"
  backend:
    type: "HTTP-VPC"
    vpcAccessName: "slbAccessForVip"
- name: MockForOldClient
  condition: "$ClientVersion < '2.0.5'"
  backend:
    type: "MOCK"
    statusCode: 400
    mockBody: "This version is not supported!!!"
- name: BlueGreenPercent05
  condition: "Random() < 0.05"
  backend:
    type: "HTTP"
    address: "https://beta-version.api.foo.com"
    constant-parameters:
      - name: x-route-blue-green
          
```

Submit

4. Modify configurations in the Script Configurations field and click **Submit**.

## Configurations

### Configuration template

1. You can configure a backend routing plugin in the JSON or YAML format because these two formats have the same schema. You can use the `yaml to json` tool to convert the plugin configuration format. The following example describes a plugin configuration template in the YAML format:

```
---
routes:
  # Responses that are no longer supported are returned to clients of an earlier version. ClientVersion is a
  # custom parameter in the API.
  - name: MockForOldClient
    condition: "$ClientVersion < '2.0.5'"
    backend:
      type: "MOCK"
      statusCode: 400
      mockBody: "This version is not supported!!!"
  # Blue-green release scenarios: Five percent of requests are routed to the backend of a blue-green release.
  - name: BlueGreenPercent05
    condition: "Random() < 0.05"
    backend:
      type: "HTTP"
      address: "https://beta-version.api.foo.com"
    constant-parameters:
      - name: x-route-blue-green
        location: header
        value: "route-blue-green"
```

The template has a root object `routes` that contains multiple route objects. Each route object is used to specify a routing rule. Each routing rule consists of the following parts:

- `name`: the name of the routing rule. The name must be unique within each plugin and can contain only letters and digits. If an API request hits the rule, an HTTP header `X-Ca-Routing-Name` that contains the name of the rule is added to the request before the request is routed to your backend service.
- `condition`: the conditional expression of the routing rule. If an API request meets the condition, the request hits the routing rule. The backend routing plugin checks the routing rules based on the order in which they are configured. The API request is routed to your backend service in the first routing rule that the request hits. After this occurs, the plugin does not check the remaining routing rules. If you configure multiple routing rules, make sure that they are configured in the order that meets your service expectations.
- `backend`: the description of your backend service. The description must be consistent with the Swagger specification files for API Gateway. The backend configurations for an API in a backend routing plugin override the original backend configurations in the API. If the backend configurations are incomplete after being overridden, the `X-Ca-Error-Code: I504RB` error is reported to the client. If this error is returned, check whether your backend configurations are complete.
- `constant-parameters`: the constant parameters that you can customize in the routing rule. Constant parameters are included in an API request before the request is routed to your backend service. These parameters are used in the business logic of your backend service. A constant parameter can be a query or header parameter.

## Conditional expressions

### Basic syntax

- The syntax of conditional expressions in backend routing plugins are similar to that of SQL statements. The basic format is `$A = 'A'` and `$B = 'B'`.
- Each parameter starts with `$`. You can reference the request parameters that are defined in an API to which a plugin is bound. The request mode of the API can be set to Request Parameter

Mapping(Filter Unknown Parameters), Request Parameter Mapping(Passthrough Unknown Parameters), or Request Parameter Passthrough. If you define a request parameter named `query1` when you configure an API, you can use `$query1` to reference this parameter in conditional expressions.

- The following constant parameter types are supported:
  - **STRING**: the string data type. Single or double quotation marks can be used to enclose a string, for example, "Hello".
  - **INTEGER**: the integer data type, for example, 1001 and -1.
  - **NUMBER**: the floating point data type, for example, 0.1 and 100.0.
  - **BOOLEAN**: the Boolean data type. Valid values: true and false.
- You can use `and` and `or` operators to connect different expressions.
- You can use parentheses `()` to specify the priority of conditional expressions.
- As a built-in function, `Random()` generates a **NUMBER**-type parameter that returns a random number in the range of `[0, 1)`.
- You can use `$CaAppId` to reference system parameters of the current request. You can reference system parameters without the need to define them in an API. However, if you have defined a parameter in the API with the same name as a system parameter, the value of the system parameter is overwritten by that of the defined parameter. The following system parameters apply to backend routing plugins:
  - **CaStage**: the environment to which the requested API is published. Valid values: **RELEASE**, **PRE**, and **TEST**.
  - **CaDomain**: the domain name of the API group to which the requested API belongs.
  - **CaRequestHandleTime**: the time in UTC at which the current request is received.
  - **CaAppId**: the value of the **AppId** parameter in the current request.
  - **CaAppKey**: the value of the **AppKey** parameter in the current request.
  - **CaClientIp**: the IP address of the client from which the current request is sent.
  - **CaApiName**: the name of the requested API.
  - **CaHttpScheme**: the protocol used by the current request. Valid values: **HTTP**, **HTTPS**, and **WS**.
  - **CaClientUa**: the **UserAgent** field uploaded from the client.
- If you use an unknown parameter in a conditional expression, such as `$UnknownParameter = 1`, the result of the expression is false.

#### Conditional expression examples

- The following expression indicates that the probability is less than 5%:

```
Random() < 0.05
```

- The following expression indicates that the requested API is published to the Test environment:

```
$CaStage = 'TEST'
```

- The following expression indicates that the custom parameter **UserName** is set to **Admin** and the source IP address is **47.47.74.77**.

```
$UserName = 'Admin' and $CaClientIp = '47.47.74.77'
```

- The following expression indicates that the **AppId** parameter is set to **1001**, **1098**, or **2011**, and the

protocol that is used by the API request is HTTPS:

```
$CaHttpScheme = 'HTTPS' and ($CaAppId = 1001 or $CaAppId = 1098 or $CaAppId = 2011)
```

## Backend configuration and overriding rules

The structure of a backend service is consistent with the Swagger definitions imported into API Gateway. The following examples describe the supported backend service types and configuration samples. The backend configurations in a backend routing plugin override the backend configurations in an API that is bound to the plugin. If you do not need to change the backend service type, specify only the parameters whose values you want to change.

- HTTP

```
---
backend:
  type: HTTP
  address: "http://10.10.100.2:8000"
  path: "/users/{userId}"
  method: GET
  timeout: 7000
```

- HTTP-VPC

```
---
backend:
  type: HTTP-VPC
  vpcId: vpc-xxxx
  vpcInstance: 172.168.1.1
  vpcInstancePort: 80
  path: "/users/{userId}"
  method: GET
  timeout: 10000
```

- MOCK

```
---
backend:
  type: MOCK
  mockResult: "mock resul sample"
  mockStatusCode: 200
  mockHeaders:
    - name: server
      value: mock
    - name: proxy
      value: GW
```

## Limits

- The metadata of a backend routing plugin can be a maximum of 16,384 bytes in size. If this limit is exceeded, the `InvalidPluginData.TooLarge` error is reported.
- A maximum of 16 routing rules can be configured in a backend routing plugin. If this limit is exceeded, the `InvalidPluginData.TooManyRoutes` error is reported.
- The size of a single conditional expression cannot exceed 512 bytes. If this limit is exceeded, the `InvalidPluginData.ConditionTooLong` error is reported.

- Configuration updates in a plugin are synchronized in real time to all the APIs bound to the plugin. An interval of at least 45s is required between two updates. If you update a plugin twice within less than 45s, the `InvalidPluginData.UpdateTooBusy` error is reported.

## Typical scenarios

- Configure multi-tenant routing. Different backend service addresses are allocated based on the AppId settings. Assume that users whose app ID is 10098 or 10099 are VIP customers. API requests from these two users are required to be routed to an independent server cluster.

```
---
-routes:
# If the AppId value for an API caller is 10098 or 10099, requests to the API are routed to an independent address.
# In this example, the VPC access name is set to slbAddressForVip.
- name: http1
  condition: "$CaAppId = 10098 or $CaAppId = 10099"
  backend:
    type: "HTTP"
    address: "https://test-env.foo.com"
```

- Configure routing based on environment settings (Test, Pre, and Release). All requests for the APIs that are published to the same environment are required to be routed to the same server.

```
---
routes:
# Route all requests for APIs that are published to the Test environment to the test server on the Internet.
- name: Vip
  condition: "$CaStage = 'TEST'"
  backend:
    type: "HTTP"
    address: "https://test-env.foo.com"
```

- Five percent of requests are required to be directed to a group of beta servers to perform a blue-green release.

```
---
routes:
# Blue-green release scenarios: Five percent of requests are routed to the backend of a blue-green release
.
- name: BlueGreenPercent05
  condition: "Random() < 0.05"
  backend:
    type: "HTTP"
    address: "https://beta-version.api.foo.com"
```

### 2.5.4.2.6. Create a caching plugin

You can bind a caching plugin to an API to cache the responses from your backend service. This reduces the load on the backend service and shortens the response time.

#### 1. Usage notes

- Caching plugins can cache only the responses to API requests that use the GET method.
- When you configure a caching plugin, you can use the following parameters to sort responses in a

cache:

- `varyByApp`: controls whether to match and serve cached responses based on the app IDs of API callers.
  - `varyByParameters`: controls whether to match and serve cached responses based on the values of specific parameters. The plugin uses the same request parameters of APIs that are bound to the plugin to sort the responses to API requests.
  - `varyByHeaders`: controls whether to match and serve cached responses based on different request headers. For example, match and serve cached responses based on the `Accept` or `Accept-Language` header.
- API Gateway provides each user with 5 MB of cache space in each region. Caches are cleared after expiration. If a cache reaches its space limit, no more responses are stored in the cache.
  - If `Cache-Control` is specified in a response from your backend service, the response is stored in a cache based on the specified cache policy. If `Cache-Control` is not specified in a response, after the response expires, the response is stored in a cache based on the default cache policy and is stored for the period of time specified by the duration parameter.
  - A response can be stored in a cache for a maximum of 48 hours (172,800 seconds) after it expires. Configurations made after the 48 hours are invalid.
  - API Gateway determines how to process the `Cache-Control` headers of client requests based on the client `CacheControl` settings. By default, API Gateway does not process the `Cache-Control` headers. You can set `clientCacheControl` to the following modes:
    - `off`: API Gateway ignores the `Cache-Control` headers of all client requests.
    - `all`: API Gateway processes the `Cache-Control` headers of all client requests.
    - `app`: API Gateway processes only the `Cache-Control` headers of client requests whose `app IDs` are included in the configured `apps` list.
  - By default, API Gateway caches only the `Content-Type`, `Content-Encoding`, and `Content-Language` headers in responses. If you need to cache more headers, add the headers in the `cacheableHeaders` parameter of the caching plugin.

## 2. Configurations

You can configure a caching plugin in the JSON or YAML format because these two formats have the same schema. You can use the `yaml to json` tool to convert the plugin configuration format. The following example describes a plugin configuration template in the YAML format:

```

---
varyByApp: false # Controls whether to match and serve cached responses based on the app IDs of API calls. Default value: false.
varyByParameters: # Controls whether to match and serve cached responses based on the values of specific parameters.
- userId # The name of a backend parameter. If the backend parameter is mapped to a parameter with a different name, set this parameter to the mapped parameter name.
varyByHeaders: # Controls whether to match and serve cached responses based on different request headers.
- Accept # Cached responses are matched and served based on the Accept header.
clientCacheControl: # API Gateway determines how to process the Cache-Control headers of client requests based on the clientCacheControl settings.
  mode: "app" # Valid values: off, all, and apps. Default value: off. off indicates that API Gateway ignores the Cache-Control headers of all client requests. all indicates that API Gateway processes the Cache-Control headers of all client requests. apps indicates that API Gateway processes only the Cache-Control headers of client requests whose app IDs are included in the configured apps list.
  apps: # A list of app IDs. If mode is set to app, API Gateway processes only the Cache-Control headers of client requests whose app IDs are in this list.
    - 1992323 # A sample app ID. It is not an AppKey.
    - 1239922 # A sample app ID. It is not an AppKey.
cacheableHeaders: # The cacheable response headers. By default, API Gateway caches only the Content-Type and Content-Length headers of backend responses.
- X-Customer-Token # The name of the cacheable response header.
duration: 3600 # The default grace period, in seconds.

```

### 3. Working mechanism

If an API request hits the cache of an API, the `X-Ca-Caching: true` header is included in the response to the API request.

### 4. Limits

- The metadata of a caching plugin can be a maximum of 16,380 bytes in size.
- A response body that exceeds 128 KB in size cannot be cached.
- Each user has a maximum of 30 MB of total cache space in each region.

#### 2.5.4.2.7. JWT authentication plugin

RFC 7519-compliant JSON Web Token (JWT) is a simple method used by API Gateway to authenticate requests. API Gateway hosts the public JSON Web Keys (JWKs) of users and uses these JWKs to sign and authenticate JWTs in requests. Then, API Gateway forwards claims to backend services as backend parameters. This simplifies the development of backend applications.

Compared to the OpenID Connect feature, the JWT authentication plugin can implement the functions of this feature and bring the following benefits:

- You do not need to configure an additional authorization API. JWTs can be generated and distributed in multiple ways. API Gateway is only responsible for JWT authentication by using public JWKs.
- JWKs without kid specified are supported.
- Multiple JWKs can be configured.
- You can read token information from the header of a request or a query parameter.
- If you want to transmit a JWT in an Authorization header, such as `Authorization bearer {token}`,

you can set `parameter` to Authorization and `parameterLocation` to header, so the token information can be correctly read.

- The `jit` claim-based anti-replay check is supported if you set `preventJtiReplay` to true.
- Requests that do not include tokens can be forwarded to backend services without verification if you set `bypassEmptyToken` to true.
- The verification on the `exp` setting for tokens can be skipped if you set `ignoreExpirationCheck` to true.

If you configure a `JWT authentication plugin` and bind it to an `API` for which the `OpenID Connect` feature is configured, the `JWT authentication plugin` takes effect in place of the `OpenID Connect` feature.

## 1. Obtain a JWK

RFC 7517-compliant JWK is used to sign and authenticate JWTs. If you want to configure a `JWT authentication plugin`, you must generate a valid `JWK` manually or by using an online `JWK generator` such as `mkjwk.org`. The following example shows a valid `JWK`. In the JWK example, the private key is used to sign the token, and the public key is configured in the `JWT authentication plugin` to authenticate the signature.

```
{
  "kty": "RSA",
  "e": "AQAB",
  "kid": "O9fpdhrViq2zaaaBEWZITz",
  "use": "sig",
  "alg": "RS256",
  "n": "qSVxcKnOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-9FPGy8NCoIO4MfLXzJ3mJ7xqglZp3NIOGXz-GIABcf13ii7kSSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2ImRh0h8ImK-vl42dwID_hOIzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ"
}
```

*The preceding JWK is in the JSON format. If you want to configure a `JWT authentication plugin` in the YAML format, you must use a JWK in the YAML format.\**

- For a `JWT authentication plugin`, you only need to configure a `public key`. Keep your `private key` safe. The following table lists the signature algorithms supported by the `JWT authentication plugin`.

Signature algorithm	Supported <code>alg</code> setting
RSASSA-PKCS1-V1_5 with SHA-2	RS256, RS384, RS512
Elliptic Curve (ECDSA) with SHA-2	ES256, ES384, ES512
HMAC using SHA-2	HS256, HS384, HS512

*When you configure a key of the HS256, HS384, or HS512 type, the key value is base64url encoded. If the signature is invalid, check whether your key is in the same format as the key used to generate the token.*

## 2. Plugin configurations

You can configure a JWT authentication plugin in the JSON or YAML format because these two formats have the same schema. You can use the `yaml to json` tool to convert the plugin configuration format. The following example describes a plugin configuration template in the YAML format:

```
---
parameter: X-Token      # The parameter from which the JWT is read. It corresponds to a parameter in an API request.
parameterLocation: header # The location from which the JWT is read. Valid values: query and header. This parameter is optional if Request Mode for the bound API is set to Request Parameter Mapping(Filter Unknown Parameters) or Request Parameter Mapping(Passthrough Unknown Parameters). This parameter is required if Request Mode for the bound API is set to Request Parameter Passthrough.
preventJtiReplay: false  # Controls whether to enable the anti-replay check for jti. Default value: false.
bypassEmptyToken: false  # Controls whether to forward requests that do not include tokens to backend services without verification.
ignoreExpirationCheck: false # Controls whether to ignore the verification of the exp setting.
claimParameters:        # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud         # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud    # The name of the backend parameter, to which the JWT claim is mapped.
  location: header        # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId      # The name of the JWT claim, which can be public or private.
  parameterName: userId   # The name of the backend parameter, to which the JWT claim is mapped.
  location: query         # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
#
# Public key in the JWK
jwk:
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-BqvT6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-9FPGy8NCoIO4MfLXzJ3mJ7xqglZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRV84tIZpFVh2lmRh0h8lmK-vl42dwlD_hOlzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
#
# You can configure multiple JWKs and use them together with the jwk field.
# If multiple JWKs are configured, kid is required. If the JWT does not include kid, the consistency check on kid fails.
jwks:
- kid: O9fpdhrViq2zaaaBEWZITz # If only one JWK is configured, kid is optional. If the JWT includes kid, API Gateway checks the consistency of kid.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v....
- kid: 10fpdhrViq2zaaaBEWZITz # If only one JWK is configured, kid is optional. If the JWT includes kid, API Gateway checks the consistency of kid.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v...
```

- The JWT authentication plugin retrieves JWTs based on the settings of `parameter` and `parameterLocation`. For example, if `parameter` is set to X-Token and `parameterLocation` is set to header, the JWT is read from the X-Token header.
- If the parameter configured in an API has the same name as the parameter specified by `parameter`, do not specify `parameterLocation`. Otherwise, an error is reported when the API is called.
- If you want to transmit a token in an Authorization header, such as `Authorization bearer {token}`, you can set `parameter` to Authorization and `parameterLocation` to header, so the token information can be correctly read.
- If `preventJtiReplay` is set to true, the JWT authentication plugin uses `jti` in `claims` to perform an anti-replay check.
- If `bypassEmptyToken` is set to true and a token is not included in a request, API Gateway skips the check and directly forwards the request to a backend service.
- If `ignoreExpirationCheck` is set to true, API Gateway skips the verification of the `exp` setting. Otherwise, API Gateway checks whether a token expires.
- If API Gateway is required to forward `claims` in tokens to backend services, you can set `tokenParameters` to configure the following parameters to be forwarded:
  - `claimName`: the name of the claim in a token, which can be `kid`.
  - `parameterName`: the name of the parameter forwarded to a backend service.
  - `location`: the location of the parameter forwarded to a backend service. Valid values: `header`, `query`, `path`, and `formData`.
    - If this parameter is set to `path`, the backend path must contain a parameter with the same name, such as `/path/{userId}`.
    - If this parameter is set to `formData`, the body of a received request in a backend service must be of the `Form` type.
- You can configure only one key in the `jwt` field. You can also configure multiple keys in the `jwt` field.
  - You can configure only one key with `kid` not specified.
  - You can configure multiple keys with `kid` specified. `kid` must be unique.

### 3. Verification rules

- A JWT authentication plugin obtains tokens based on the settings of `parameter` and `parameterToken`. If API Gateway is required to forward requests to backend services even when tokens are not included in the requests, set `bypassEmptyToken` to true.
- If you want to configure multiple keys, abide by the following principles:
  - Preferentially select a key whose ID is the same as the value of `kid` in a token for signature and authentication.
  - You can configure only one key with `kid` not specified. If there is no key whose ID is the same as the value of `kid` in a token, use the key with `kid` not specified for signature and authentication.
  - If all the configured keys have specified `kid` settings, and the token in a request does not contain `kid` or no keys match `kid`, an `A403JK` error is reported.
- If a token contains `iat`, `nbf`, and `exp`, the JWT authentication plugin verifies the validity of their time formats.
- By default, API Gateway verifies the setting of `exp`. If you want to skip the verification, set `ignoreExpirationCheck` to true.

- `tokenParameters` is configured to extract the required parameters from the `claims` of a token. These parameters are forwarded to backend services.

## 4. Configuration examples

### 4.1 Configure a single JWK

```
---
parameter: X-Token      # The parameter from which the JWT is read. It corresponds to a parameter in an API
                        request.
parameterLocation: header # The location from which the JWT is read. Valid values: query and header. This
                        parameter is optional if Request Mode for the bound API is set to Request Parameter Mapping(Filter Unknown Parameters) or Request Parameter Mapping(Passthrough Unknown Parameters). This parameter is required if Request Mode for the bound API is set to Request Parameter Passthrough.
claimParameters:        # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud         # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud   # The name of the backend parameter, to which the JWT claim is mapped.
  location: header       # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId      # The name of the JWT claim, which can be public or private.
  parameterName: userId  # The name of the backend parameter, to which the JWT claim is mapped.
  location: query        # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
preventJtiReplay: false # Controls whether to enable the anti-replay check for jti. Default value: false.
#
# Public key in the JWK
jwk:
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5vGsOmaorPDzHUubBmZZ4UXj-9do7w9X1uKFXAnqfto4TepSNuYU2bA_-tzSLAGBsR-Bqv
  T6w9SjxakeiyQpVmexxnDw5WZwpWenUAcYrfSPEoNU-0hAQwFYgqZwJQMN8ptxkd0170PFauwACox4Hfr-9F
  PGy8NCoIO4MfLXzJ3mJ7xqglZp3NIOGXz-GIAbCf13ii7kSStpYqN3L_zzpvXUAos1FJ9IPXRv84tIZpFVh2lmRh0h8
  lmK-vl42dwID_hOlzayL1Xno2R0T-d5AwTSdnep7g-Fwu8-sj4cCRWq3bd61Zs2QOJ8iustH0vSRMYdP5oYQ
```

### 4.2. Configure multiple JWKs

```

---
parameter: Authorization # The parameter from which the token is obtained.
parameterLocation: header # The location from which the token is obtained.
claimParameters:      # The claims to be converted into parameters. API Gateway maps JWT claims to backend parameters.
- claimName: aud      # The name of the JWT claim, which can be public or private.
  parameterName: X-Aud # The name of the backend parameter, to which the JWT claim is mapped.
  location: header     # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
- claimName: userId   # The name of the JWT claim, which can be public or private.
  parameterName: userId # The name of the backend parameter, to which the JWT claim is mapped.
  location: query      # The location of the backend parameter, to which the JWT claim is mapped. Valid values: query, header, path, and formData.
preventJtiReplay: true # Controls whether to enable the anti-replay check for jti. Default value: false.
jwks:
- kid: O9fpdhrViq2zaaaBEWZITz # kid must be set to different values for different JWKS.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v...
- kid: 10fpdhrViq2zaaaBEWZITz # kid must be set to different values for different JWKS.
  kty: RSA
  e: AQAB
  use: sig
  alg: RS256
  n: qSVxcknOm0uCq5v...

```

## 5. Error codes

Status	Code	Message	Description
400	I400JR	JWT required	No JWT-related parameters are found.
403	S403JI	Claim <code>jti</code> is required when <code>preventJtiReplay:true</code>	No valid <code>jti</code> claims are included in the request when <code>preventJtiReplay</code> is set to true in a <code>JWT authentication plugin</code> .
403	S403JU	Claim <code>jti</code> in JWT is used	The <code>jti</code> claim that is included in the request has been used when <code>preventJtiReplay</code> is set to true in a <code>JWT authentication plugin</code> .
403	A403JT	Invalid JWT: \${Reason}	The <code>JWT</code> that is read from the request is invalid.

Status	Code	Message	Description
400	I400JD	JWT Deserialize Failed: <code>#{Token}</code>	The <code>JWT</code> that is read from the request fails to be parsed.
403	A403JK	No matching JWK, <code>kid:#{kid}</code> not found	No <code>JWK</code> matches <code>kid</code> configured in the <code>JWT</code> included in the request.
403	A403JE	JWT is expired at <code>#{Date}</code>	The <code>JWT</code> that is read from the request expires.
400	I400JP	Invalid JWT plugin config: <code>#{JWT}</code>	The <code>JWT</code> authentication plugin is incorrectly configured.

If an HTTP response message includes an unexpected response code specified by `ErrorCode` in the `X-Ca-Error-Code` header, such as `A403JT` or `I400JD`, you can visit the [jwt.io](https://jwt.io) website to check the token validity and format.

## 6. Limits

- The metadata of a JWT authentication plugin can contain a maximum of **16,380** characters.
- You can configure a maximum of **16** parameters to be forwarded. Both the `claimName` and `parameterName` parameters cannot exceed 32 characters in length. Only the following regular expression is supported: `[A-Za-z0-9-_.]`.
- `alg` can be set to RS256, RS384, RS512, ES256, ES384, ES512, HS256, HS384, or HS512 for JWKs.

### 2.5.4.2.8. Access control plugin

#### 1. Overview

In an access control plugin, you can define conditions based on the request parameters or context of an API to which the plugin is bound. This allows you to determine whether to deliver an API request to a backend service. For information about how to define parameters and use conditional expressions, see [Use parameters and conditional expressions](#).

#### 2. Configurations

Assume that the API request path is `/#{userId}/...`. JWT authentication is enabled for APIs. Two claims, `userId` and `userType`, are available in the JWT. The following plugin verification conditions apply:

- If `userType` is set to `admin`, requests in all paths are allowed.
- If `userType` is set to `user`, only the requests in the same `/#{userId}` path are allowed.

```

---
#
# Assume that the API request path is /{userId}/... in this example.
# JWT authentication is enabled for APIs. Two claims, userId and userType, are available in the JWT.
# The following plugin verification conditions apply:
# - If userType is set to admin, requests in all paths are allowed.
# - If userType is set to user, only the requests in the same /{userId} path are allowed.
parameters:
  userId: "Token:userId"
  userType: "Token:userType"
  pathUserId: "path:userId"
#
# Rules are defined based on the preceding parameters. For each API request, the plugin checks the rules in sequence. If a condition in a rule is met, the result is true and the action that is specified by ifTrue is performed. If a condition in a rule is not met, the result is false and the action that is specified by ifFalse is performed.
# The action ALLOW indicates that the request is allowed. The action DENY indicates that the request is denied and an error code is returned to the client. After the ALLOW or DENY action is performed, the plugin does not check the remaining conditions.
# If neither the ALLOW nor DENY action is performed, the plugin proceeds to check the next condition.
rules:
  - name: admin
    condition: "$userType = 'admin'"
    ifTrue: "ALLOW"
  - name: user
    condition: "$userId = $pathUserId"
    ifFalse: "DENY"
    statusCode: 403
    errorMessage: "Path not match ${userId} vs /${pathUserId}"
    responseHeaders:
      Content-Type: application/xml
    responseBody:
      <Reason>Path not match ${userId} vs /${pathUserId}</Reason>

```

### 3. Relevant errors

Error code	HTTP status code	Message	Description
A403AC	403	Access Control Forbidden by \${RuleName}	The error message returned because the request is rejected by the access control plugin that is bound to the API.

### 4. Limits

- A maximum of **16** parameters can be specified in an access control plugin.
- Each conditional expression can contain a maximum of **512** characters.
- The metadata of an access control plugin can contain a maximum of **16,380** characters.
- A maximum of 16 **rules** can be configured in each access control plugin.

## 2.5.4.2.9. Error code mapping plugin

An **error code mapping plugin** is used to map backend error responses to expected error responses based on mapping rules that are defined by clients.

### 1. Overview

An error code mapping plugin is used to map backend error responses to expected error responses based on mapping rules that are defined by clients.

### 2. Quick start

The following example shows an error response that is returned by a backend service. The HTTP status code is 200, but the response body contains an error message in a JSON string.

```
HTTP 200 OK
Content-Type:application/json
{"req_msg_id":"d02afa56394f4588832bed46614e1772","result_code":"ROLE_NOT_EXISTS"}
```

- Assume that clients want to receive an HTTP status code other than 200 but do not want to modify backend configurations. For example, clients expect the following error response:

```
HTTP 404
X-Ca-Error-Message: Role Not Exists, ResultId=d02afa56394f4588832bed46614e1772
```

In this case, you can use the following sample to configure an error code mapping plugin and bind the plugin to relevant APIs:

```
---
# The parameters that are involved in mapping.
parameters:
  statusCode: "StatusCode"
  resultCode: "BodyJsonField:$.result_code"
  resultId: "BodyJsonField:$.req_msg_id"
# The mapping condition.
errorCondition: "$statusCode = 200 and $resultCode <> 'OK'"
# The parameter in an error response that is used to specify the error code and hit mapping rules.
errorCode: "resultCode"
# Mapping rules.
mappings:
  - code: "ROLE_NOT_EXISTS"
    statusCode: 404
    errorMessage: "Role Not Exists, RequestId=${resultId}"
  - code: "INVALID_PARAMETER"
    statusCode: 400
    errorMessage: "Invalid Parameter, RequestId=${resultId}"
# Optional. The default mapping rule.
defaultMapping:
  statusCode: 500
  errorMessage: "Unknown Error, ${resultCode}, RequestId=${resultId}"
```

In this example, the HTTP status code and the `result_code` parameter in an error response are used to define the mapping condition. If the HTTP status code of an error response is 200 and the value of the `result_code` parameter is not 'OK', the mapping starts. The `result_code` parameter is used to define the mapping rules. If the value of the `result_code` parameter is `ROLE_NOT_EXISTS`, the original HTTP status code is mapped to 404. If the value of the `result_code` parameter is `INVALID_PARAMETER`, the original HTTP status code is mapped to 400. If the value of the `result_code` parameter is neither of the preceding values, the original HTTP status code is mapped to 500.

### 3. Plugin configurations and mapping rules

#### 3.1 Plugin configurations

You can configure an error code mapping plugin in the `JSON` or `YAML` format. The following parameters can be specified:

- `parameters` : required. The parameters that are involved in mapping. These parameters are specified as key-value pairs in the `map` format. For information about how to define parameters and write conditional expressions, see [Use parameters and conditional expressions](#).
- `errorCondition` : required. The condition under which a response is considered an error response. If the result of the conditional expression is `true`, the mapping starts.
- `errorCode` : optional. The parameter that is used to specify the error code in an error response and hit mapping rules. The error code that is specified by this parameter is compared with the value of the `code` parameter in the mapping rules specified by `mappings`.
- `mappings` : required. The mapping rules. API Gateway reconstructs error responses based on the setting of `errorCode` or `errorCondition`. A mapping rule may contain the following parameters:
  - `code` : optional. The value of this parameter must be unique among all mapping rules. If the error code of an error response is the same as the value of the `code` parameter in the current mapping rule, the error response is mapped based on the current mapping rule.
  - `condition` : optional. The condition under which an error response needs to be mapped based on the current mapping rule. If the result of the conditional expression is `true`, the error response is mapped based on the current mapping rule.
  - `statusCode` : required. The HTTP status code that replaces the original HTTP status code of an error response if the error response needs to be mapped based on the current mapping rule.
  - `errorMessage` : optional. The error message that is returned to the client after mapping. The value of this parameter is obtained from the parameters in the original backend error response and is also stored in the `errorMessage` parameter in error logs. In the error response after mapping, this parameter is displayed as the value of the `X-Ca-Error-Message` header.

- **responseHeaders** : optional. The response headers that are included in an error response after mapping if the current mapping rule is hit. This parameter is specified as key-value pairs in the **map** format.
- **responseBody** : optional. The response body that overwrites the original response body of an error response if the error response needs to be mapped based on the current mapping rule.
- **defaultMapping** : optional. The default mapping rule. If all the rules that are defined in **mappings** are not hit by an error response, the error response is mapped based on this default mapping rule
- **statusCode** : required. The HTTP status code that replaces the original HTTP status code of an error response if the error response needs to be mapped based on the current mapping rule.
- **errorMessage** : optional. The error message that is returned to the client after mapping. The value of this parameter is obtained from the parameters in the original backend error response and is also stored in the **errorMessage** parameter in error logs. In the error response after mapping, this parameter is displayed as the value of the **X-Ca-Error-Message** header.
- **responseHeaders** : optional. The response headers that are included in an error response after mapping if the current mapping rule is hit. This parameter is specified as key-value pairs in the **map** format
- **responseBody** : optional. The response body that overwrites the original response body of an error response if the error response needs to be mapped based on the current mapping rule.

Take note of the following points when you configure an error code mapping plugin:

- The parameters that are used to write conditional expressions in **mappingCondition** and **mappings[].condition** must be defined in **parameters** . Otherwise, the plugin does not work and reports an error. For information about how to define parameters and write conditional expressions, see [Use parameters and conditional expressions](#).
- The value of the **errorCode** parameter must be the name of a parameter that is defined in **parameters** .
- When you configure a mapping rule specified by **mappings** , you must specify **code** or **condition** . When you specify **code** , the value of this parameter must be unique among all mapping rules. When you specify **condition** , you must write conditional expressions in the order that meets your requirements. This is because the order of conditions determines their priorities.
- You can specify **errorMessage** and **responseBody** in a format similar to **"\${Code}: \${Message}"** and obtain the parameter values from those specified in **parameters** .
- You can specify **responseHeaders** in the **\${Message}** format.

- If you do not specify `responseBody`, the body of an error response returned to the client after mapping is the same as that of the original error response.
- You can use the `responseHeaders` parameter to specify headers and their settings to replace corresponding headers in a backend error response. If you specify the value of a header as `"`, this header will be deleted after mapping. If you do not specify this parameter, the headers of the error response returned to the client after mapping are the same as those of the original error response.
- If you do not specify `defaultMapping`, the error code mapping does not take effect. The original error response from your backend service is returned to the client.

### 3.2 Parameters involved in mapping

As described in the following code, you must specify the parameters that are involved in mapping as key-value pairs in `parameters`. Each key is the name of a parameter. Each value is specified in the `Location:Name` format. This format indicates that the value of a parameter is obtained from a specific location in the response or system context.

```
---
# The parameters that are involved in mapping.
parameters:
  statusCode: "StatusCode"
  resultCode: "BodyJsonField:$.result_code"
  resultId: "BodyJsonField:$.req_msg_id"
```

An error code mapping plugin supports the parameters at specific locations in the following table.

Location	Included in	Description
StatusCode	Response	The HTTP status code in a backend error response, such as <code>200</code> or <code>400</code> .
ErrorCode	Response	The error code of a system error response in API Gateway.
ErrorMessage	Response	The system error message in API Gateway.
Header	Response	Use <code>Header:{Name}</code> to obtain the first value of the HTTP header that is specified by <code>{Name}</code> .

Location	Included in	Description
BodyJsonField	Response*	Use <code>BodyJsonField:{JPath}</code> to obtain the JSON string in the body of an API request or a backend response in <code>JSONPath</code> mode.
System	Response	Use <code>System:{Name}</code> to obtain the value of the system parameter that is specified by <code>{Name}</code> .
Token	Response	If <code>JWT</code> is used with <code>OAuth2</code> for authentication, use <code>Token:{Name}</code> to obtain the value of the parameter that is specified by <code>{Name}</code> in a token.

- `ErrorCode` and `ErrorMessage` are used to return system error codes and detailed system error information in API Gateway. For more information, see [Error codes](#).
- `BodyJsonField` can be used to obtain the JSON string in the body of a backend response. However, if the size of the response body exceeds 15,360 bytes, the string obtained is `null`.

### 3.3 Working mechanism

The following operations describe how an error code mapping plugin works:

- Step 1: Based on the list of parameters that are defined in `parameters`, the plugin obtains the values of the parameters from a backend error response and the system context.
- Step 2: The plugin uses the parameters and obtained values to execute the conditional expression that is written in `errorCondition`. If the result is `true`, go to the next step. If the result is `false`, the process ends.
- Step 3: If `errorCode` is specified, the plugin obtains the value of `errorCode`. Then the plugin checks whether there is a mapping rule that indicates that the `errorCode` setting is the same as the setting of `code`. The mapping rule is specified by `mappings`.
- Step 4: If no mapping rule meets requirements, the plugin executes in sequence the conditional expressions that are written in `condition` in mapping rules.
- Step 5: If a mapping rule is hit in Step 3 or Step 4, the original error response is mapped based on the mapping rule. Otherwise, the original error response is mapped based on the default mapping rule.

### 3.4 Mapping of system error codes and error logs

- In API Gateway, system errors may occur in processes such as check, verification, throttling, and plugin operations. For more information, see [Error codes](#). You can use `ErrorCode` as a location to obtain information in a system error response. For example, clients support only HTTP status code 200 and want to map HTTP status code 429 that is returned by API Gateway to HTTP status code 200.
- For a system error response, the values that are obtained from locations such as `StatusCode`, `Header`, and `BodyJsonField` are all `null`. When you define a mapping condition for an error code mapping plugin, note that for a backend error response, the value that is obtained from the `ErrorCode` location is `OK`.
- The error code of a system error response is specified by the `X-Ca-Error-Code` header and by the `errorCode` parameter in error logs. This value cannot be overwritten by an error code mapping plugin.
- The `statusCode` parameter in error logs records the value of the HTTP status code that is sent from API Gateway to the client. This value can be overwritten by an error code mapping plugin.

## 4. Configuration examples

### 4.1 Use the error codes in error responses for mapping

Mapping

```
---
# The parameters that are involved in mapping.
parameters:
  statusCode: "StatusCode"
  resultCode: "BodyJsonField:$.result_code"
  resultId: "BodyJsonField:$.req_msg_id"
# The mapping condition.
errorCondition: "$statusCode = 200 and $resultCode <> 'OK'"
# The parameter in an error response that is used to specify the error code and hit mapping rules.
errorCode: "resultCode"
# Mapping rules.
mappings:
  - code: "ROLE_NOT_EXISTS"
    statusCode: 404
    errorMessage: "Role Not Exists, RequestId=${resultId}"
  - code: "INVALID_PARAMETER"
    statusCode: 400
    errorMessage: "Invalid Parameter, RequestId=${resultId}"
# Optional. The default mapping rule.
defaultMapping:
  statusCode: 500
  errorMessage: "Unknown Error, ${resultCode}, RequestId=${resultId}"
```

## 5. Limits

- A maximum of **16** parameters can be specified in an error code mapping plugin.
- A single conditional expression can contain a maximum of **512** characters.
- If you use the **BodyJsonField** location to obtain the JSON string in the body of an error response, the size of the response body cannot exceed **16,380** bytes. If the size of the response body exceeds this limit, the obtained string is null.
- The metadata of an error code mapping plugin can contain a maximum of **16,380** characters.
- For an error code mapping plugin, you can configure a maximum of **20** mapping rules by using the **condition** parameter defined in **mappings**.

### 2.5.4.3. Bind a plugin to an API

After you create a plugin, you must bind the plugin to an API for the plugin to take effect.

#### Context

You can bind a plugin to multiple APIs. The plugin will individually take effect on each API. For each type of plugin, you can bind only one plugin of such type to an API. If you bind two plugins of the same type to an API, the new plugin will replace the previous one and take effect.

#### Procedure

1. [Log on to the API Gateway console](#)
2. In the left-side navigation pane, choose **Publish APIs -> Plugin**.
3. On the Plugins list page, find the target plugin and click **Bind API** in the Operation column.
4. Select the publish environment and the group of the APIs to which you want to bind a plugin.
5. To bind a plugin to one API, find the target API and click +Add in the Operation column to add the API to the right pane. To bind a plugin to multiple APIs, select the target APIs and click Add Selected in the lower-left corner to add these APIs to the right pane. Then, click **OK**.

**Bind API**

You will bind the API to the following plugins:

Plugin Name: testErrorMapping

**Please note: If the API has already been bound to a plugin of the same type, it will be overwritten by this plugin. Please choose carefully!**

Select the API to bind to:

testGroup Release Enter the API name to search Search

API Name	Operation
<input type="checkbox"/> testAPI3	+ Add
<input type="checkbox"/> testAPI	+ Add

Add Selected 2 entries in total 1

Selected API(s) (1)

testAPI3 X Remove

OK Cancel

## 2.5.4.4. Delete a plugin

You can delete existing plugins.

### Procedure

1. [Log on to the API Gateway console](#)
2. In the left-side navigation pane, choose **Publish APIs Plugin**.
3. On the Plugins list page, find the target plugin and click **Delete** in the Operation column.
4. In the Confirm Deletion message, click **OK**.

## 2.5.4.5. Unbind a plugin

You can unbind plugins from the APIs to which they are bound.

### Procedure

1. [Log on to the API Gateway console](#)
2. In the left-side navigation pane, choose **Publish APIs Plugin**.
3. On the Plugins list page, click the name of the target plugin to go to the **Create Plugin** page.
4. Click **Bound API List**. The bound APIs are displayed. Find the target APIs one at a time and click **Unbind** in the Operation column.
5. In the Confirm Unbind message, click **OK**.

## 2.6. Manage monitoring

## 2.6.1. View monitoring information and configure alerts

API Gateway works together with Cloud Monitor to provide visualized real-time monitoring and alerting features. You can use these features to obtain statistical data about your APIs in multiple dimensions, such as the number of API calls, traffic, backend response time, and error distribution. You can view data in different units of time.

### View monitoring information of API calls

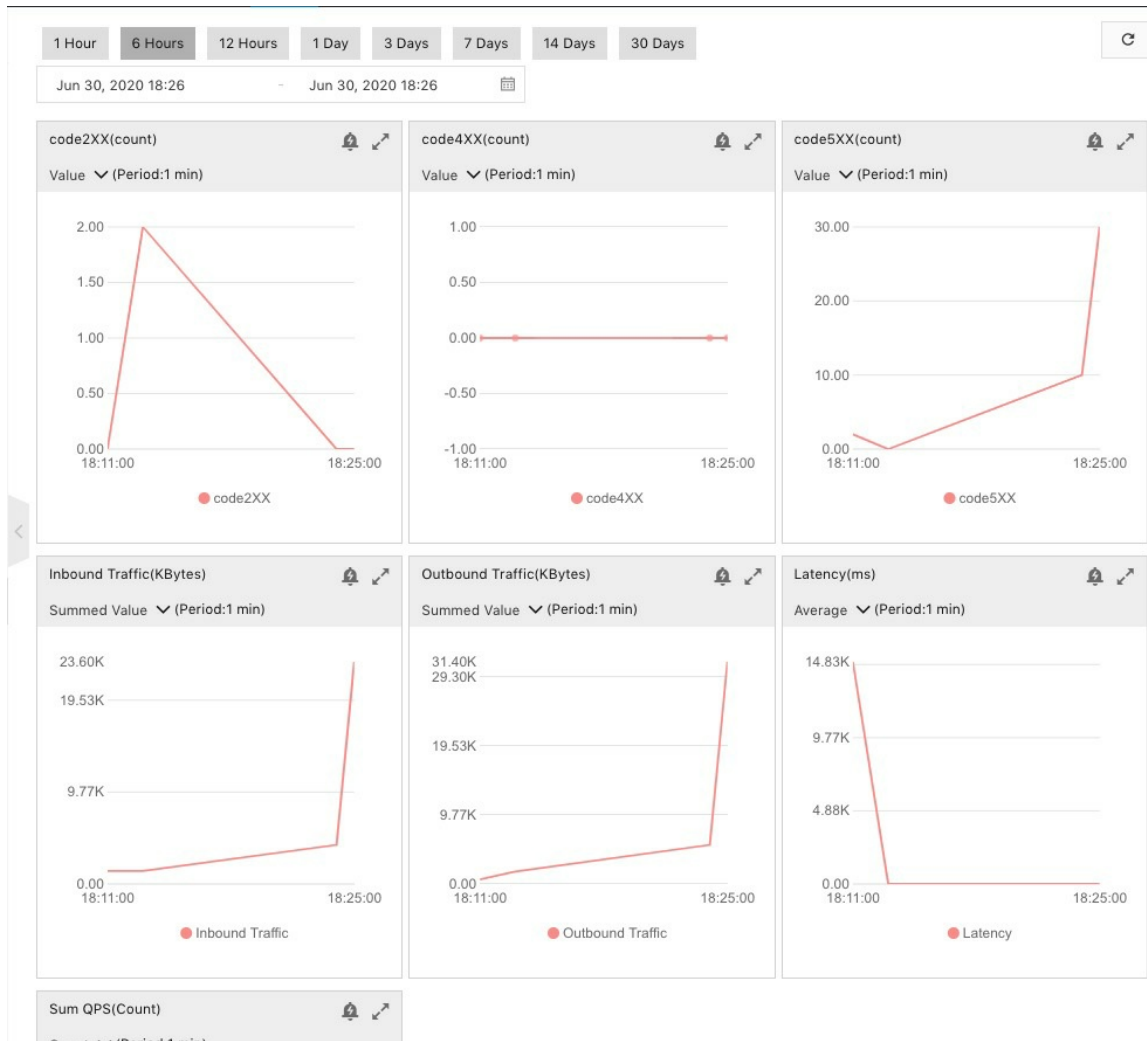
Perform the following steps to view data of API calls under your Apsara Stack tenant account in the Cloud Monitor console.

1. In the top navigation bar of the Apsara Uni-manager Management Console, choose **Products > Monitoring and O&M > Cloud Monitor**.
2. In the Cloud Monitor console, choose **Cloud Service Monitoring > API Gateway** in the left-side navigation pane.

The screenshot shows the CloudMonitor console interface. On the left is a navigation pane with a tree structure. The 'API Gateway' option is selected and highlighted. The main content area is titled 'API Gateway Monitoring List'. It features a search bar with a dropdown for 'API Name' and a text input for 'Enter an instance name', followed by a 'Search' button. Below the search bar is a table with columns: 'API Name', 'Group Name', 'Created At', 'Modified At', and 'Actions'. There are two rows of data. The first row shows an API named 'testAPI3' with a long ID, belonging to 'testGroup', created on Jun 30, 2020, and modified on Jun 30, 2020. The second row shows an API named 'testAPI' with a long ID, also in 'testGroup', created and modified on May 21, 2020. Each row has a checkbox in the 'API Name' column and a link in the 'Actions' column that says 'Monitoring Charts | Alarm Rules'. At the bottom of the table, there are two buttons: 'Create Alarm Rule' and 'Alarm Rules'.

API Name	Group Name	Created At	Modified At	Actions
<input type="checkbox"/> testAPI3 f63b308fc121472ca41eaf413c0470e1	testGroup	Jun 30, 2020, 18:05:52	Jun 30, 2020, 18:12:21	<a href="#">Monitoring Charts   Alarm Rules</a>
<input type="checkbox"/> testAPI fe8d45ddcd640988dac288bef8af530	testGroup	May 21, 2020, 21:29:32	May 21, 2020, 21:29:32	<a href="#">Monitoring Charts   Alarm Rules</a>

3. On the API Gateway Monitoring List page, find your API and click **Monitoring Charts** in the Actions column.



#### Descriptions of monitoring charts:

- (Old)code2XX Number(count)  
: shows the number of requests with a 2XX HTTP status code returned. A 2XX HTTP status code, such as 200, indicates that the request succeeded at the backend.
- code4XX Number(count)  
: shows the number of requests with a 4XX HTTP status code returned. A 4XX HTTP status code, such as 404, indicates a client error.
- (Old)code5XX Number(count)  
: shows the number of requests with a 5XX HTTP status code returned. A 5XX HTTP status code, such as 500, indicates a server error.
- (Old)TrafficRX(KBytes)  
: shows the size of API requests.
- (Old)TrafficTX(KBytes)  
: shows the size of API responses.
- (Old)Latency(ms)  
: shows the response time of your backend service. The latency in API Gateway ranges from 3 ms to 5 ms, which is excluded from the response time.
- (Old)SumQPS(Count)  
: shows the total number of requests initiated for the API.

## Configure API alert rules

You can configure API alert rules in the Cloud Monitor console to achieve real-time API alerting.

1. On the API Gateway Monitoring List page, find your API.
2. Click **Alert Rules** in the Actions column.
3. On the Alert Rules page, click **Create Alert Rule** in the upper-right corner. In the Modify Alert Rule panel, the Product parameter is set to API Gateway by default, and Resource Range is set to your API by default.
4. Click **Add Rule Description** and specify Rule Name, Metric Name, Comparison, and Threshold and Alert Level in the Add Rule Description panel. Then, click **OK**.

### Add Rule Description

\*Rule Name

APIservice

\*Metric Name

code5XX

Comparison

>=

Drop down to show more options

\*Threshold And Alarm Level(Unit:count)

Critical	10	Continuous 3 Count Peri... ▼
Warn	5	Continuous 3 Count Peri... ▼
Info		Continuous 3 Count Peri... ▼

OK

5. Specify Alert Contact Group and click OK.

Create Alarm Rule

Resource Range

☒ Instances

Resource Range

testAPI3

Rule Description

Rule Name	Rule Description	Resource Description	Actions
APIService	(Critical) code5XX continuous 3 times consecutivelyValue>=10 Send a notification. (Warn) code5XX continuous 3 times consecutivelyValue>=5 Send a notification.	-	<div></div> <div></div>

Add Rule Description

Effective Time

24 h

Effective From

00:00

To

23:59

HTTP CallBack

Alarm Contact Group

Default Contact Group

OK

Cancel

 **Note**


To monitor the service status of APIs, we recommend that you monitor the 5XX HTTP status code.

## 2.6.2. View statistical information on the dashboard of API Gateway

You can view monitoring information about API calls in the Cloud Monitor console. The API Gateway console provides an overview page for statistics of API calls. You can also view statistical information about API calls on the dashboard page.

1. Log on to the API Gateway console. In the left-side navigation pane, click **Instances**. On the Instance list page, click **View Dashboard** in the upper-right corner.
2. On the page that appears, select different time granularities to view specific information about the API groups and API calls under your account. By default, this page displays a summary of API calls by domain name. The summarized information includes the number of requests, return code, and the latency to call a backend service.

3. To view the API calls of a specific domain name, click **domain List** in the left-side navigation pane. On the page that appears, click the name of the required domain name to go to the domain Detail page. You can view all API calls under this domain name on this page.

 **Note** On the dashboard page, you can view only the information about API groups and API calls under your account. Even user root cannot view data on the dashboard page.

## 2.7. Advanced usage

### 2.7.1. Customize business parameters for logs

API Gateway provides the Hack mode, which allows you to record the request and response parameters of API calls in logs.

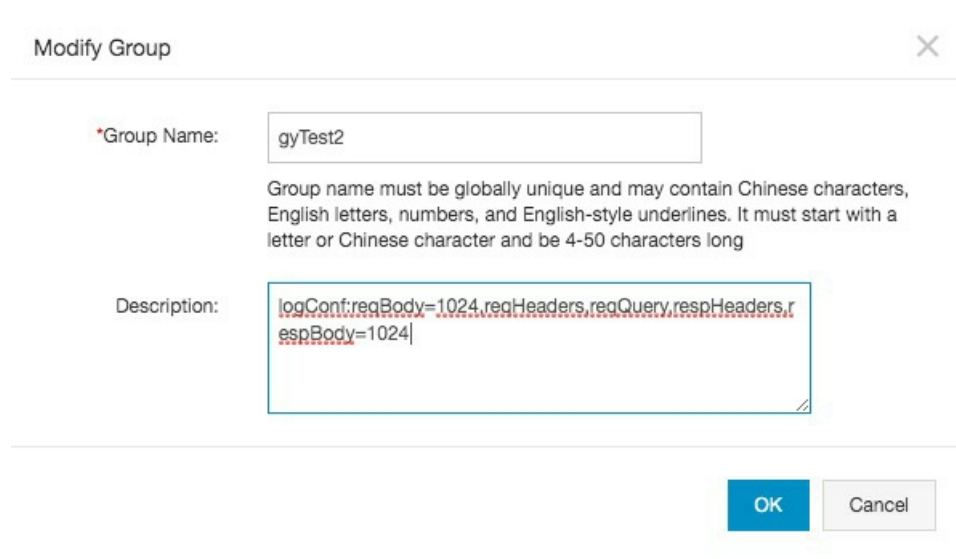
#### Procedure

1. [Log on to the API Gateway console](#)
2. In the left-side navigation pane, choose **Publish APIs > API Groups**.
3. On the Group List page, find your API group and click its name to go to the Group Details page.
4. Click **Modify Group Message** in the upper-right corner.
5. In the **Description** field of the Modify Group dialog box, add the following content:

```
logConf:reqBody=1024,reqHeaders,reqQuery,respHeaders,respBody=1024
```

 **Note**

- The content must be in a separate line. Otherwise, the configuration fails.
- You can also modify the content based on your requirements. For example, you can remove respHeaders to omit the response header and its content in the logs.
- reqBody=1024 indicates that the log records a maximum of 1,024 characters from the request body. Extra characters are discarded.



**Modify Group**

\*Group Name:

Group name must be globally unique and may contain Chinese characters, English letters, numbers, and English-style underlines. It must start with a letter or Chinese character and be 4-50 characters long

Description:

**OK** **Cancel**

## 6. Log on to the Log Service console and check whether the description change has been recorded.

```
apiUid:73c226d7169148489a71c5d33813b3a5
appId:157555089414951
appName:integration_root
clientIp:172.31.224.26
clientNonce:c3de6af9-17be-49a4-b516-12d7d7535101
domain:ad41bca1175433389b06fa1669e2472.apigateway.inter.env11b.shuguang.com
errorCode:
errorMessage:
exception:
httpMethod:GET
initialRequestId:
instanceId:
path:/testIpControl
providerApiUid:1663875445751523
region:cn-qingdao-env11-d01
requestBody:
requestHandleTime:2020-01-03T07:52:14Z
requestHeaders:{"X-Ca-Key":"157555089415157","X-Ca-Stage":"PRE","X-Forwarded-Proto":"http","Host":"ad41bca1175433389b06fa1669e2472.apigateway.inter.env11b.shuguang.com","Date":"Fri, 03 Jan 2020 07:52:14 GMT","X-Ca-Signature-Headers":"X-Ca-Key,X-Ca-Nonce,X-Ca-Timestamp","X-Ca-Nonce":"c3de6af9-17be-49a4-b516-12d7d7535101","X-Ca-Timestamp":"1578037934480","X-Ca-Signature-Method":"HmacSHA256","X-Forwarded-For":"172.31.224.26","X-Ca-Signature":"CIBQR3azw5GgGZQgZT8l5m3V4C2TncCKUVnn0se5c=","eagleeye-rpid":"0.1","X-Real-IP":"172.31.224.26","accept-encoding":"gzip","user-agent":"unirest-java/1.3.11"}
requestId:F88B9A2C-1191-4BC7-98AE-D5304BFC88DA
requestProtocol:http
requestQueryString:
requestSize:554
responseBody:
{"Body":"","Headers":{"date":"Fri, 03 Jan 2020 07:52:14 GMT","host":"172.31.224.26:8080","x-ca-request-id":"F88B9A2C-1191-4BC7-98AE-D5304BFC88DA","connection":"Keep-Alive","accept-encoding":"gzip","x-ca-stage":"PRE","user-agent":"unirest-java/1.3.11","via":"73c226d7169148489a71c5d33813b3a5"},"Method":"GET","Params":{},"Path":"/web/cloudapi","RequestURL":"http://172.31.224.26:8080/web/cloudapi"}
responseHeaders:{"Transfer-Encoding":"chunked","Date":"Fri, 03 Jan 2020 07:52:14 GMT","Content-Type":"application/json"}
responseSize:220
serviceLatency:1
statusCode:200
```

## 2.7.2. Configure Log Service logs for API Gateway

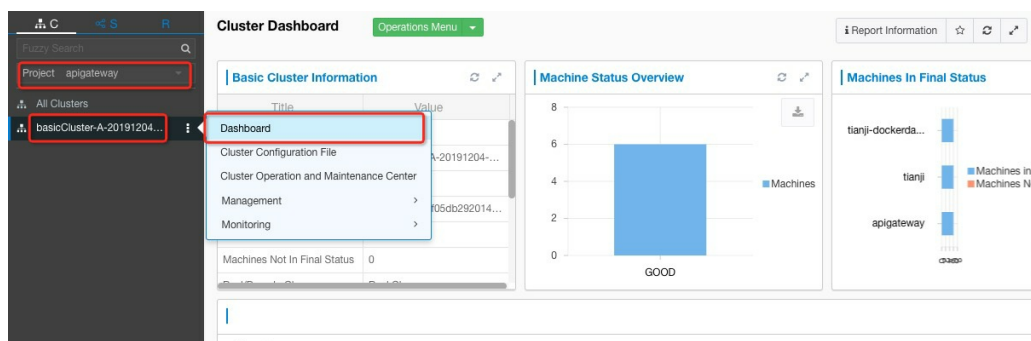
### 2.7.2.1. Initialize the default Log Service configuration of API Gateway


By default, API call logs in API Gateway are synchronized to Log Service in Apsara Infrastructure Management Framework. However, your account can be activated only after you log on to the Log Service console from Apsara Infrastructure Management Framework.

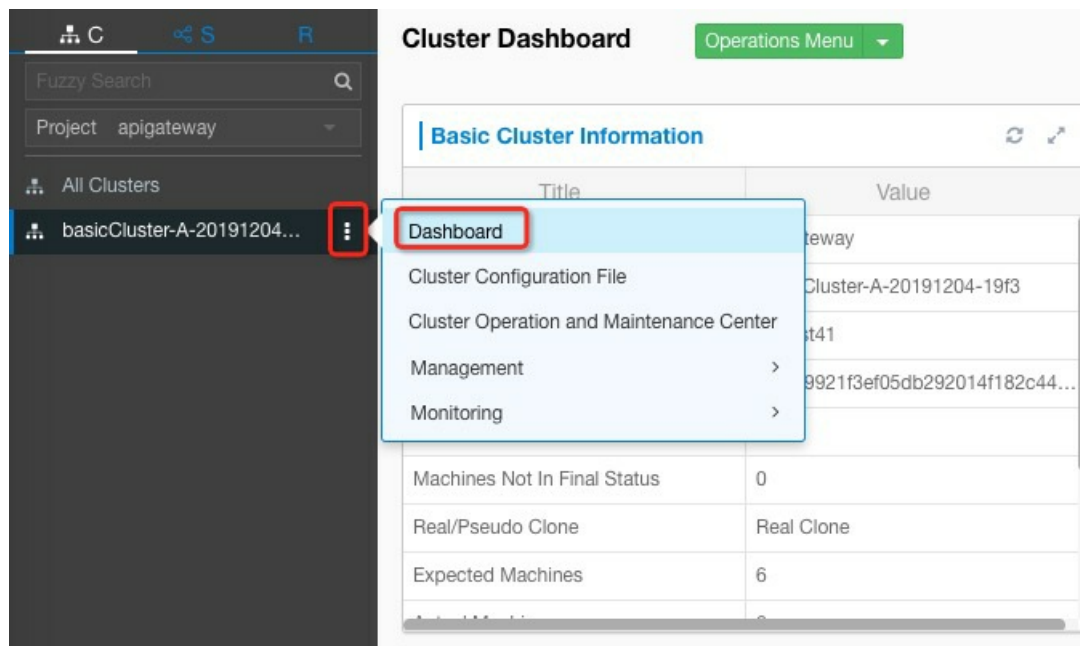
#### Procedure

1. Log on to the Apsara Infrastructure Management Framework console.
2. In the left-side navigation pane, click Tools, Operation Tools, and then Machine Tools. On the Machine Tools page, click Go. On the page that appears, click the C tab in the left-side navigation pane. Then select *apigateway* from the **Project** drop-down list.

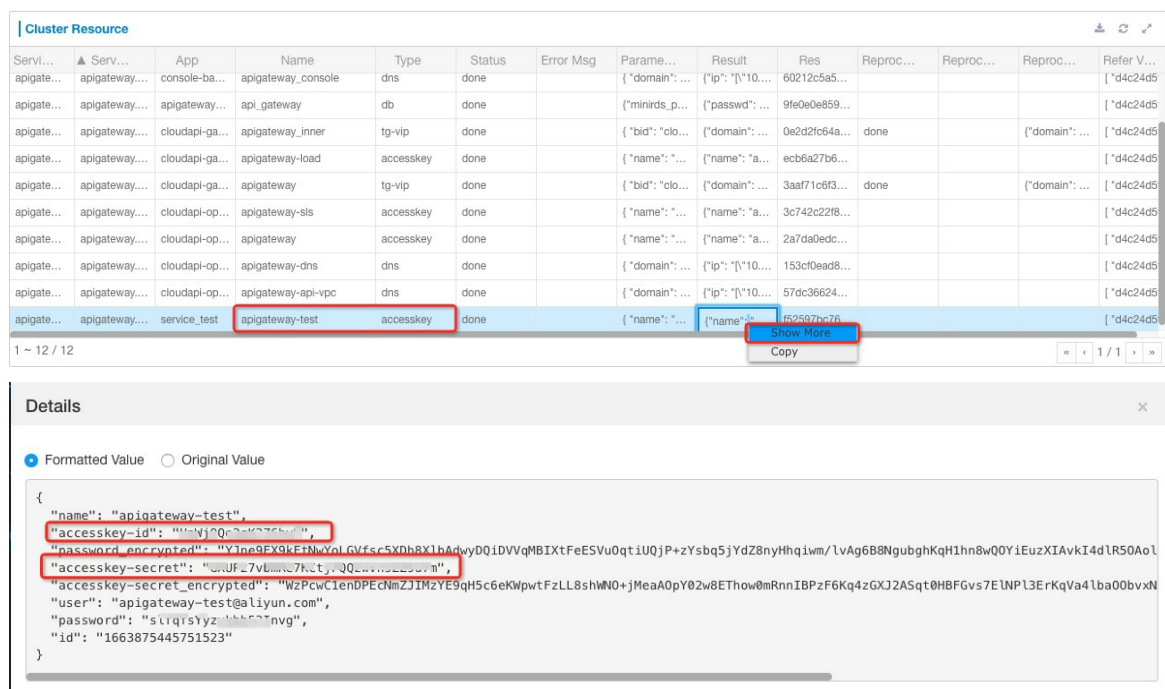
Cluster O&M page



3. Move the pointer over the  icon next to one of the filtered clusters, and select **Dashboard** from the shortcut menu.

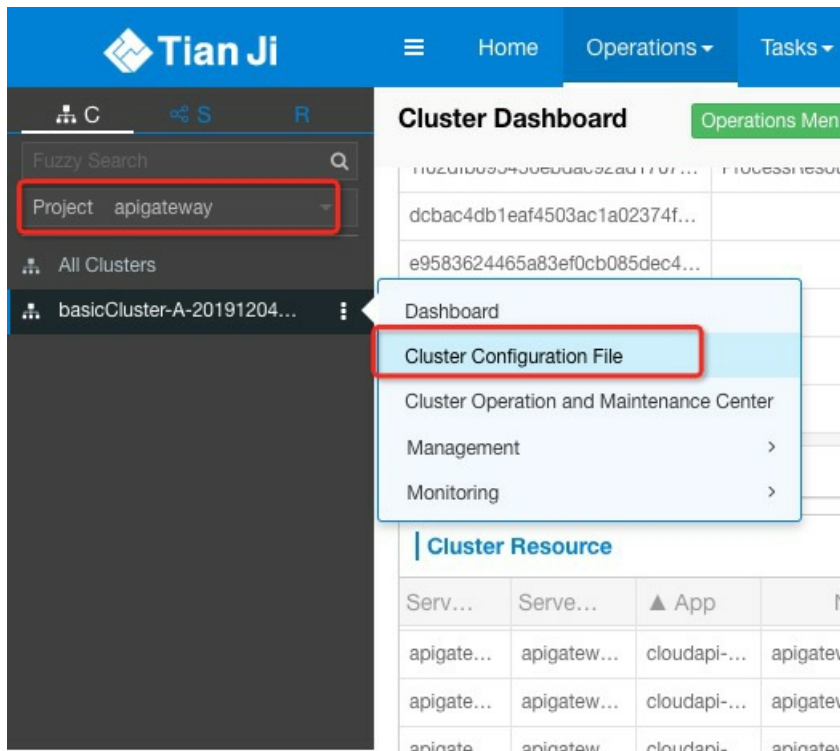


4. In the Cluster Resource section of the page that appears, find the service with Name set to apigateway-sls and Type set to accesskey. Right-click the value in the Result column and select **Show More** from the shortcut menu to view the values of accesskey-id and accesskey-secret.

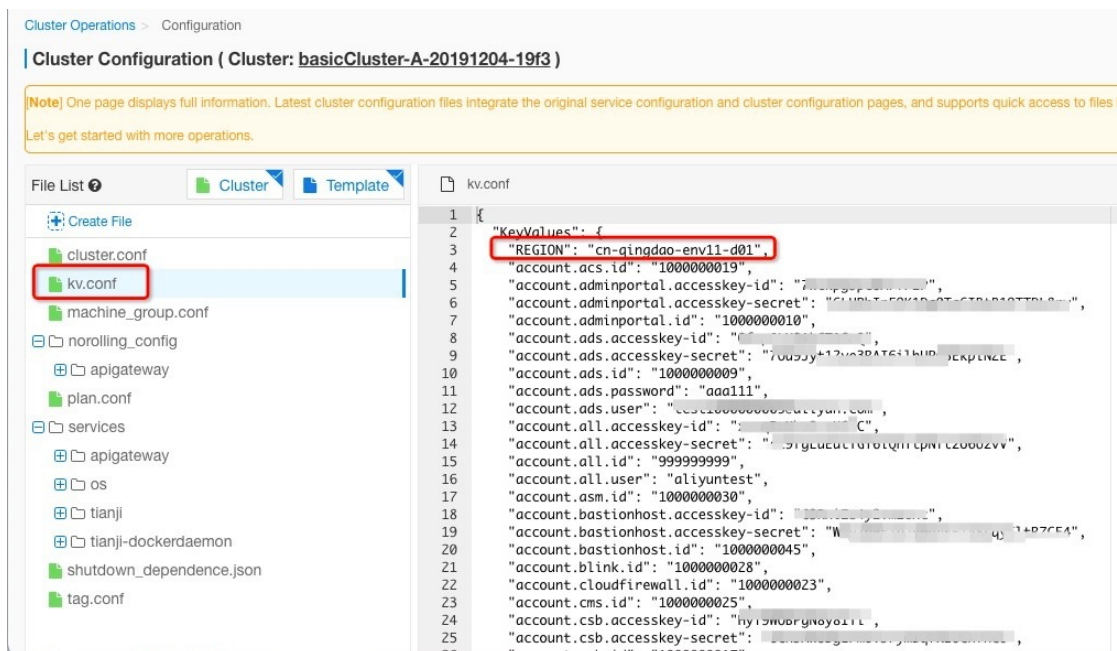


5. Use the obtained accesskey-id and accesskey-secret values to log on to the Log Service console from Apsara Infrastructure Management Framework. The URL for the Log Service console in Apsara Infrastructure Management Framework is `http://portal.${region}.sls.${internet-domain}`. You can obtain the values of region and internet-domain from the kv.conf file in the Apsara Infrastructure Management Framework console.

- i. Move the pointer over the More icon next to the apigateway cluster and select **Cluster Configuration File** from the shortcut menu.



- ii. Click the **kv.conf** file to view the values of region and internet-domain.



**Note** After you log on to the Log Service console, Log Service is automatically configured for API Gateway. This operation takes several minutes.

## 2.7.2.2. Configure API Gateway to ship logs to your Log Service project

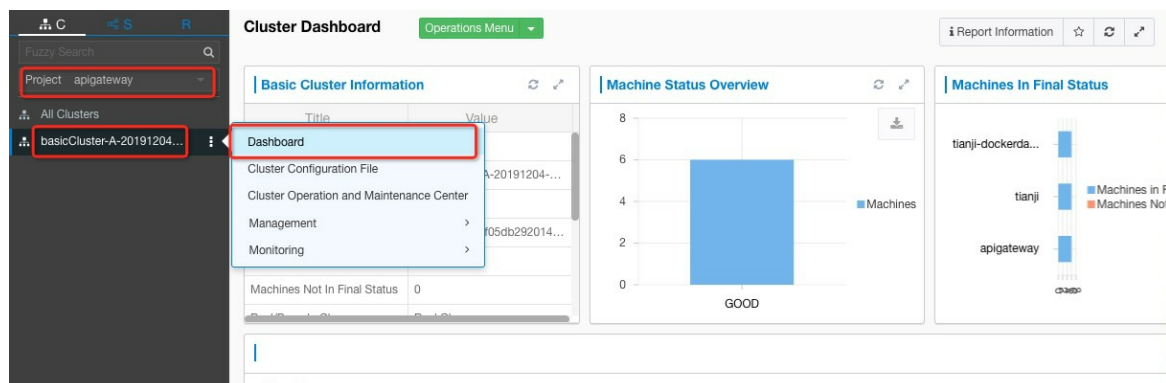
If you do not want to use Log Service and a fixed account in Apsara Infrastructure Management Framework, you can create a Log Service project. Then, you can configure API Gateway to ship logs to your Log Service project.

### Context

You must perform the following steps to create a Log Service project and configure Logstores and machine groups:

#### 1. Log on to the machine where API Gateway resides

1. Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, choose Tools > Operation Tools > Machine Tools. The Operation Tools page appears. Click the C tab in the left-side navigation pane. Then, select apigateway from the Project drop-down list. Move the pointer over the More icon next to one of the filtered clusters, and select Dashboard from the shortcut menu.



2. In the **Service Instances** section of the page that appears, find the row where the value of Service Instance is apigateway.

Service Instances					
Service Instance	Final Status	Expected Server Roles	Server Roles In Final Status	Server Roles Going Offline	Actions
apigateway	True	5	5	0	Actions ▾ Details
os	True	--	--	--	Actions ▾ Details
tianji	True	1	1	0	Actions ▾ Details
tianji-dockerdaemon	True	1	1	0	Actions ▾ Details

3. Click **Details** in the Actions column. In the **Server Role List** section of the page that appears, find the row where the value of Server Role is ApigatewayLite#.

Server Role List							
Server Role	Current Status	Expected Machines	Machines In Final ...	Machines Going ...	Rolling Task Status	Time Used	Actions
ApigatewayConsole#	In Final Status	2	2	0	no rolling		Details
ApigatewayDB#	In Final Status	1	1	0	no rolling		Details
ApigatewayLite#	In Final Status	3	3	0	no rolling		Details
ApigatewayOpenAPI#	In Final Status	2	2	0	no rolling		Details
ServiceTest#	In Final Status	1	1	0	no rolling		Details

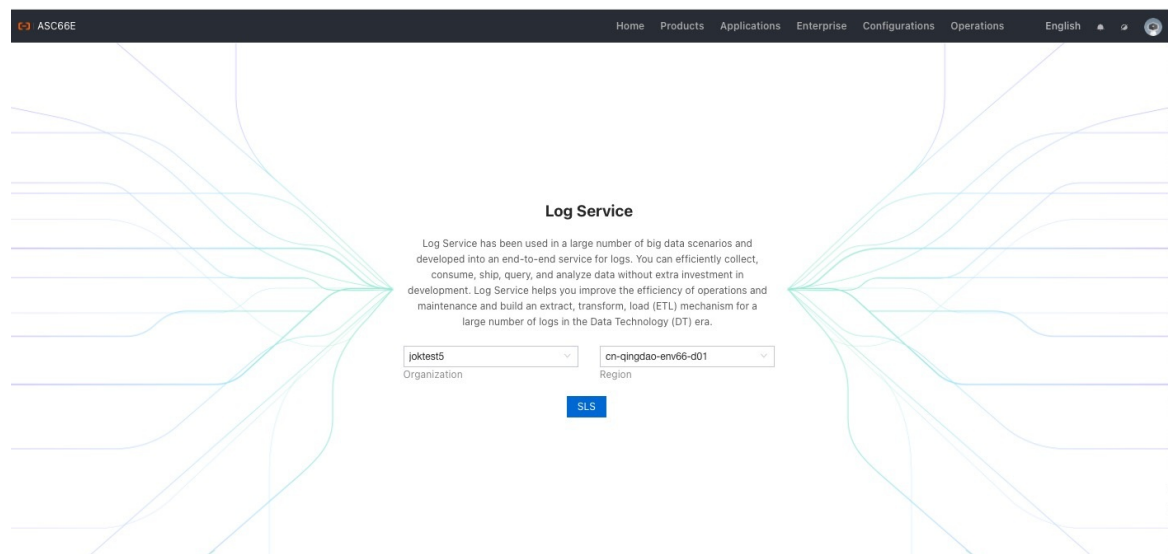
4. Click **Details** in the Actions column. Find the required machine in the **Machine Information** section

and click Terminal in the Actions column to log on to the machine.

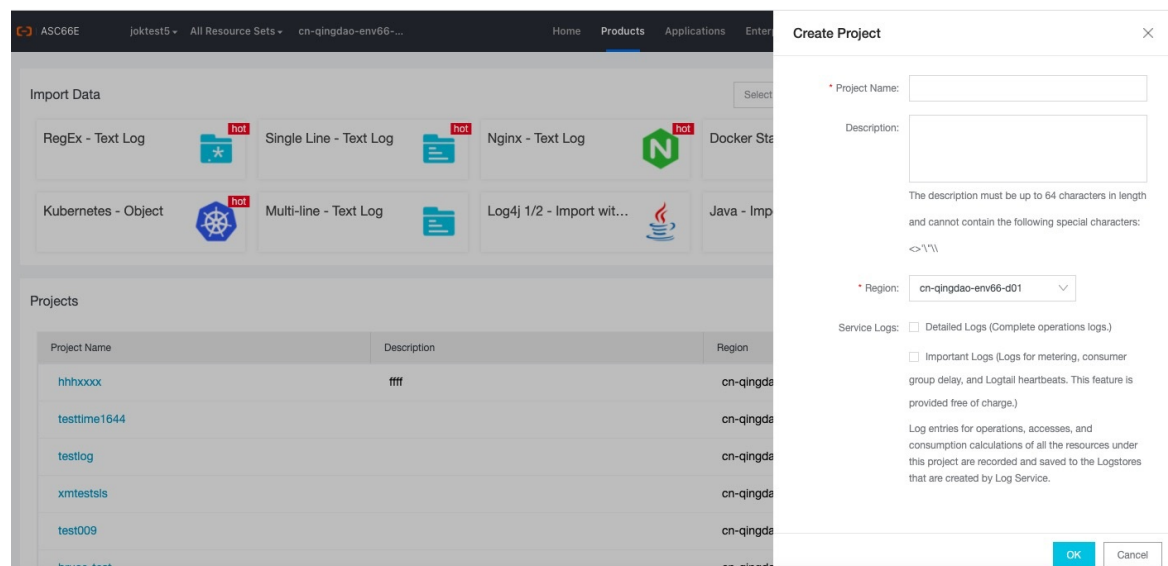
Machine Information								
Machi...	IP	Machi...	Machi...	Server...	Server...	Curren...	Target ...	Error ...
ecsapigate...	172.31.228.9	good		good   PR...		f52a09921...	f52a09921...	
ecsapigate...	172.31.22...	good		good   PR...		f52a09921...	f52a09921...	
ecsapigate...	172.31.22...	good		good   PR...		f52a09921...	f52a09921...	

## 2. Configure logs in the Log Service console

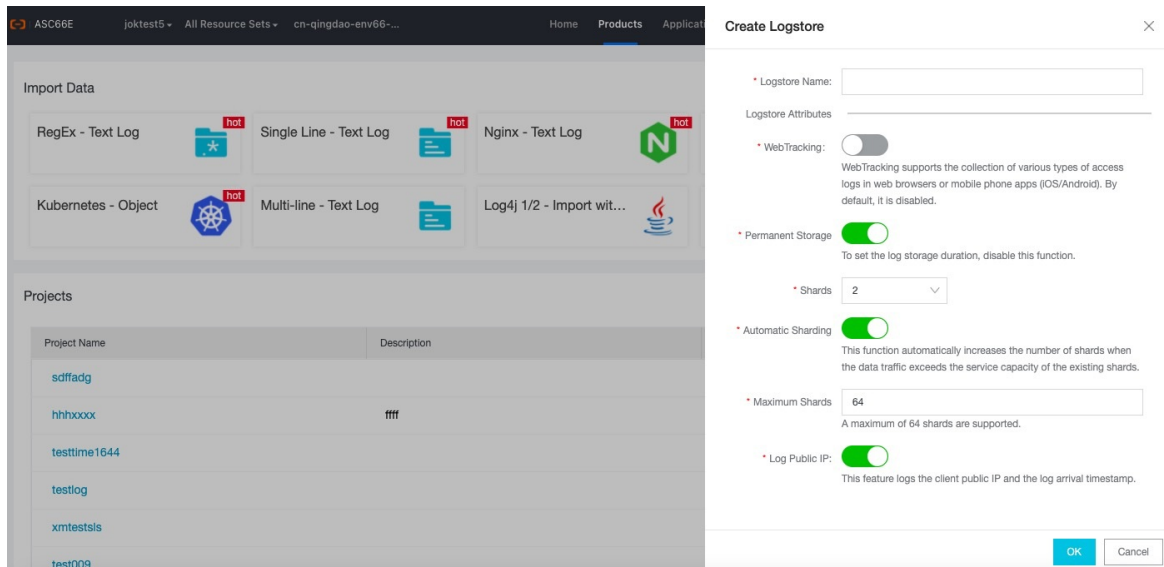
1. Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose Products — Application Services — Log Service. On the page that appears, specify **Region** and **Organization** and click SLS.



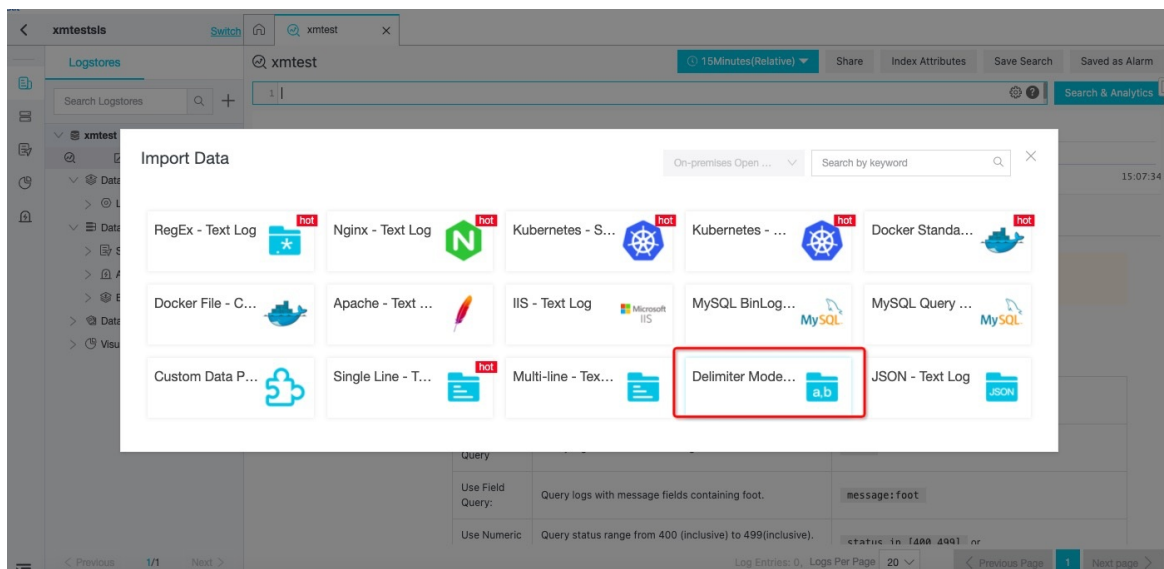
2. Click Create Project. In the Create Project panel, configure parameters and click OK.



3. In the Created message, click OK. In the Create Logstore panel, configure parameters and click OK to create a Logstore.



4. In the Created message, click OK. On the Import Data page, click **Delimiter Mode - Text Log**.



5. Access the container whose service role starts with ApigatewayLite in the Apsara Infrastructure Management Framework console. Then, run the installation command that is displayed in the Create Machine step on the TerminalService page. The operations are similar to those in the first section. If the "install logtail success" message appears, the installation succeeds.

Project: xmtests1s Logstores: xmtest Region: cn-qingdao-env66-d01

1 Logtail is a log collection agent provided by Log Service. Install the Logtail agent on the servers. [Use Existing Machine Groups](#)

**On-premises Servers**

**Linux**

Network Type	Installation Command
Alibaba Cloud Intranet (Classic Network, VPC)	wget http://data.cn-qingdao-env66-d01.sls-pub.inter.env66.shuguang.com/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh install

**Windows**

- Download the installation package. [Click here to download the Logtail installation package.](#)
- Decompress logtail\_installer.zip to the current directory.
- Run the installation command corresponding to your network type.

Network Type	Installation Command
Alibaba Cloud Intranet (Classic Network, VPC)	./logtail_installer.exe -install cn-qingdao-env66-d01

Log on to the server, and configure AliUid to identify your Alibaba Cloud account.

**Linux-based server:**

Create a file named after the AliUid in the directory /etc/logtail/users. If the directory does not exist, you need to create one. You can configure multiple AliUids for a single server by running the following commands:

```
touch /etc/logtail/users/1560108780048968
```

6. After the installation is complete, click Complete Installation and configure parameters in the Create Machine step. Specify name. Set IP Addresses to the IP addresses of all the Docker containers whose service role starts with ApigatewayLite.

Project: xmtests1s Logstores: xmtest Region: cn-qingdao-env66-d01

1 Specify Logstore 2 **Create Machine Group** 3 Machine Group Settings 4 Logtail Config 5 Configure Query and Analysis 6 End

**Name:**

**Identifier:** ☒ IP Addresses ☐ Custom ID

**Topic:**

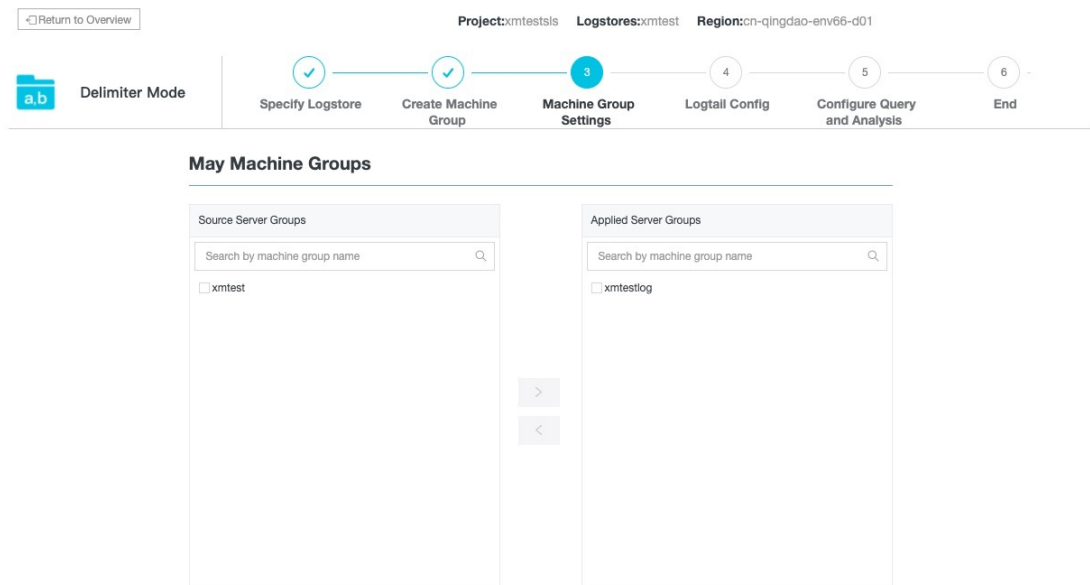
**\* IP Addresses:**

Enter an IP address. Separate multiple IP addresses with line breaks.

1. Enter the value of the ip field in app\_info.json.  
2. Separate two IP addresses with a line break.  
3. Windows and Linux instances are not allowed in the same machine group.

[Previous](#) [Skip](#) [Return to Installation](#) [Next](#)

7. Click Next to go to the Machine Group step. Configure machine groups and click Next. A heart beat check automatically starts. If the check succeeds, you are redirected to the Logtail Config step.

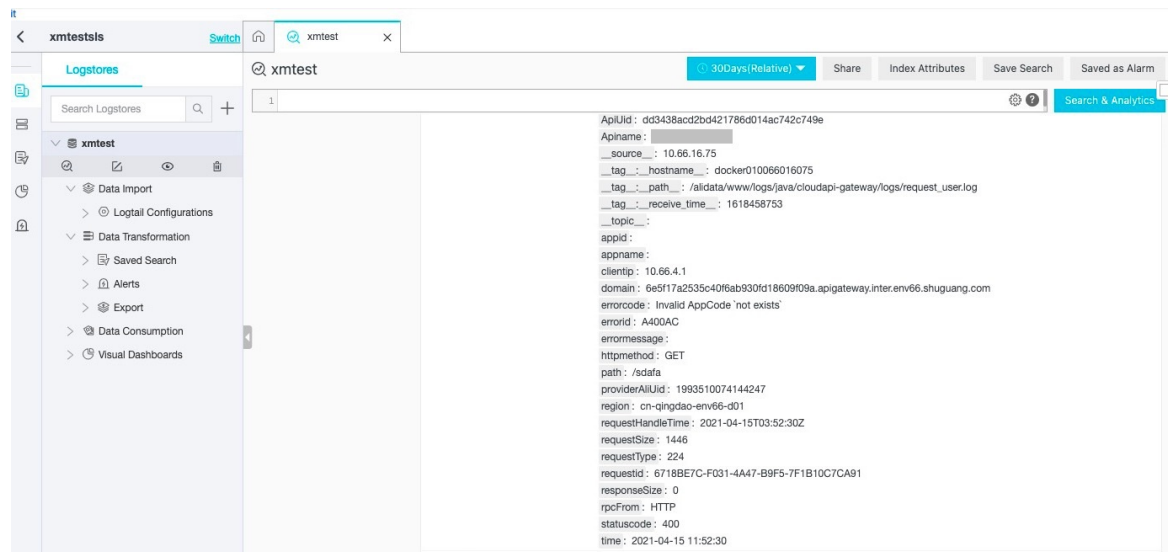


8. In the Logtail Config step, configure the parameters that are described in the following table and click **Next**.

Parameter	Value
Config Name	gateway_request_log
Log Path	/alidata/www/logs/java/cloudapi-gateway/logs/request_user.log
Mode	Delimiter Mode
Log Sample	<div>2019-08-15 14:25:05 CC7C526B-C915-44F1-93A2-F44DBBE35177 b2909c9fa66146f19baf2bd8f4709ab8 integration_root e4032f87ace14cc6965cf5352a9637a6 RELEASE 0619f7763b004fc3a16a41dd712ce8d7 biz1_anonymous 10.4.21.241   b2909c9fa66146f19baf2bd8f4709ab8.apigateway.env4b.shuguang.com POST /biz1/anonymous 403 A403JT:Invalid JWT: deserialized JWT failed  1453964555641148 cn-qingdao-env4b-d01 2019-08-15T06:25:05Z 614 0 0 A403JT http   cf802da1-54d0-49b8-b77c-2b20b172d90a  {"X-JWT-Token":"bad jwt token"} a=%21111     </div>
Delimiter	Vertical Line

9. Use the default settings until the entire configuration process is complete.
10. After the configuration is complete, make some API calls in the API Gateway console and check

whether the configuration takes effect in the Log Service console.



## 2.7.3. Cross-user VPC authorization

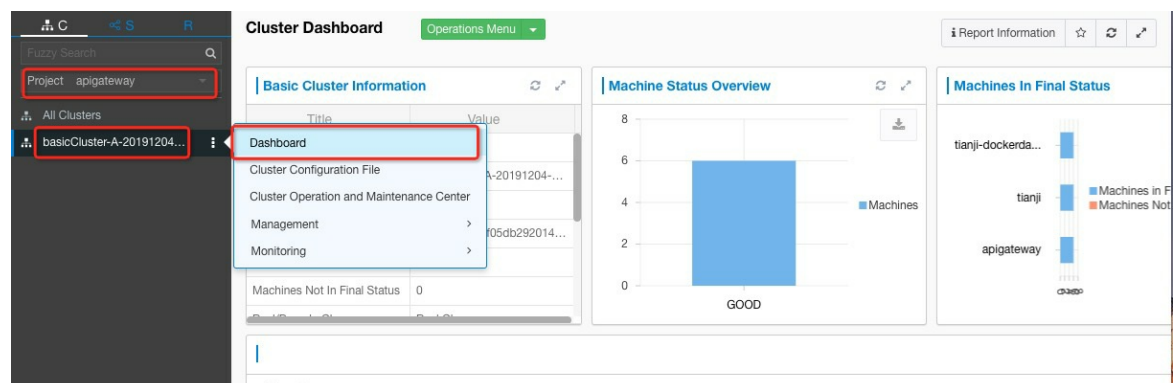
If your backend service resides in a VPC, you must configure the backend service address through VPC authorization. By default, a VPC owner must be the same user as an API owner. Starting from Apsara Stack V3.8.1, API Gateway provides two internal APIs that allow VPC owners to authorize their VPCs to other users.

### 2.7.3.1. User authorization across VPCs

APIs can be used across multiple VPCs. For security reasons, VPC owners must call APIs to explicitly authorize access to VPCs before API providers can use the VPCs. The OpenAPI component of API Gateway has a built-in Aliyun CLI tool. You can use this tool to authorize access to VPCs. The following steps describe how to call the API:

#### Procedure

1. Log on to the Apsara Infrastructure Management Framework console. In the left-side navigation pane, click the C tab and select **apigateway** from the Project drop-down list. Place the pointer over the More icon next to one of the filtered clusters and choose **Dashboard** from the shortcut menu.



2. On the dashboard page, go to **Service Instance List**, and find **apigateway** in the Service Instance column.

Service Instances					
Service Instance	Final Status	Expected Server Roles	Server Roles In Final Status	Server Roles Going Offline	Actions
apigateway	True	5	5	0	Actions ▾ Details
os	True	--	--	--	Actions ▾ Details
tianji	True	1	1	0	Actions ▾ Details
tianji-dockerdaemon	True	1	1	0	Actions ▾ Details

- Click **Details** in the Actions column corresponding to the apigateway service instance. In the **Server Role List** section, find ApigatewayLite# in the Server Role column.

Server Role List							
Server Role	Current Status	Expected Machines	Machines In Final ...	Machines Going ...	Rolling Task Status	Time Used	Actions
ApigatewayConsole#	In Final Status	2	2	0	no rolling		Details
ApigatewayDB#	In Final Status	1	1	0	no rolling		Details
ApigatewayLite#	In Final Status	3	3	0	no rolling		Details
ApigatewayOpenAPI#	In Final Status	2	2	0	no rolling		Details
ServiceTest#	In Final Status	1	1	0	no rolling		Details

- Click **Details** in the Actions column corresponding to the ApigatewayLite# server role. Then, find the **Machine Information** section.

Machine Information									
Machi...	IP	Machi...	Machi...	Server...	Server...	Curren...	Target ...	Error ...	Actions
ecsapigate...	172.31.228.9	good		good   PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation
ecsapigate...	172.31.22...	good		good   PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation
ecsapigate...	172.31.22...	good		good   PR...		f52a09921...	f52a09921...		Terminal Restart Details Machine System View Machine Operation

- Click **Terminal** in the Actions column corresponding to a machine in the server role to access the container.
- Configure the AccessKey pair used to call the CLI tool. Run the following commands:

```
aliyun configure --profile vpctest //Add the AccessKey pair configuration. vpctest is the profile name, which can be customized. After you press Enter, configure AccessKeyId and AccessSecret as prompted.
aliyun configure list //View the config configuration to check whether the preceding profile has been added.
```

- Run the following command to perform authorization:

```
aliyun cloudapi AuthorizeVpc --VpcId vpc-tb5mfcwx3s4zqctzw**** --TargetUserId 147546214349****
--endpoint apigateway.cn-qingdao-env11-d01.inter.env11b.shuguang.com --force
--profile vpctest
```

**Note**

- Log on to the Apsara Infrastructure Management Framework console. In the top navigation bar, choose **Reports > System Reports**. On the System Reports page, click Registration Vars of Services. On the Registration Vars of Services report, right-click the value in the Service Registration column corresponding to the apigateway service and choose Show More from the shortcut menu. The value of the apigateway.openapi.endpoint variable must be used as the endpoint in the preceding command. The profile value also needs to be replaced based on actual needs.
- This command authorizes the user whose ID is 1475462143497330 to use the VPC with the ID vpc-tb5mfcwx3s4zqctzw19w2. You can replace the parameter values as needed.

**Success Operation Sample**

```
[root@docker010011102023 ~]#
#aliyun cloudapi AuthorizeVpc --VpcId vpc-tb5mfcwx3s4zqctzw19w2 --TargetUserId 1475462143497330 --endpoint apigateway.cn-qingdao-env
11-d01.inter.envlib.shuguang.com --force --profile vptest
{"RequestId": "8E64BC51-60D7-4D85-B5F0-3E3F12C4B6D2"}
```

## 2.7.3.2. Configure APIs

After an app is authorized to call an API, the API owner must configure the API and define the API backend service. This is because you cannot select the VPC ID of other users in the Apsara Uni-manager Management Console.

### Context

When you configure an API, take note of the following points:

- Backend Service Type cannot be set to VPC.
- The backend service address must be in the http(s)://{Backend service IP address}. {vpcId}.gateway.vpc:{port} format. The content in {} can be substituted as required. Example:

```
http://192.168.XX.XX.vpc-tb5mfcwx3s4zqctzw****.gateway.vpc:8080
```

### Procedure

- Log on to the [API Gateway console](#)
- In the left-side navigation pane, choose **Publish APIs > APIs**.
- On the API List page, find your API and perform the following operations:
  - Click the name of the API to go to the **API Definition** page. You can view information about the API.
  - Click **Edit** in the upper-right corner, modify configurations as required, and then click **Save**. The procedure of creating an API is similar to that of modifying an API. For more information about how to create an API, see [Create an API](#). If you want to cancel the modifications before the modifications are submitted, click Cancel Edit in the upper-right corner of the edit page.

## 2.7.4. Call an API over HTTPS

### Context

API Gateway locates a unique API group by domain name, and locates a unique API in the API group by using Path and HTTPMethod. API Gateway assigns a second-level domain for each API group. You can use the domain name to call APIs that belong to the API group. The second-level domain supports only access over HTTP. You can also use a custom domain to call APIs. This topic describes how to call APIs by using a second-level domain or by using a custom domain.


## Use a second-level domain to call APIs over HTTPS

To use a second-level domain to call APIs over HTTPS, you must perform the following steps to configure a wildcard domain name certificate in Apsara Stack. You must prepare the certificate yourself.

1. Prepare configuration files for a second-level domain. Modify configurations in the following code: Replace \*.wildcard.com with your wildcard domain name \*.apigateway.\${internet-domain}. You can obtain the value of the \${internet-domain} variable in the kv.conf configuration file for Apsara Infrastructure Management Framework. For example, if the domain name that you use to provide external services is abc.alibaba.com, replace \*.wildcard.com with \*.alibaba.com. Save the modified code to a wildcard.conf file. Set the name of the public key file in the certificate to wildcard.crt. Set the name of the private key file in the certificate to wildcard.key.

```
server {
    listen      443 http2 ssl;
    server_name *.wildcard.com;
    limit_req   zone=perserver_req burst=100;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:ECDHE-RSA-AES128-SHA256:! aNULL:! eNULL:! RC4:! EXPORT:! DES:! 3DES:! MD5:! DSS:! PKS;
    ssl_certificate /home/admin/cai/certs/wildcard.crt;
    ssl_certificate_key /home/admin/cai/certs/wildcard.key;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    location / {
        proxy_pass http://gateway_upstream;
    }
}
```

2. Place the files at specified locations. Copy and paste the wildcard.conf file to the /alidata/sites/conf directory of the host for apigateway.ApiGatewayLite#. Copy and paste the wildcard.crt and wildcard.key files to the /alidata/sites/certs directory of the host for apigateway.ApiGatewayLite#.
3. Activate the certificate. Run `/home/admin/cai/bin/nginxctl reload` in the apigateway.ApiGatewayLite# SR container.

 **Note** You must perform Step 2 and Step 3 on all machines under the apigateway.ApiGatewayLite# SR container.

## Use a custom domain to call APIs over HTTPS

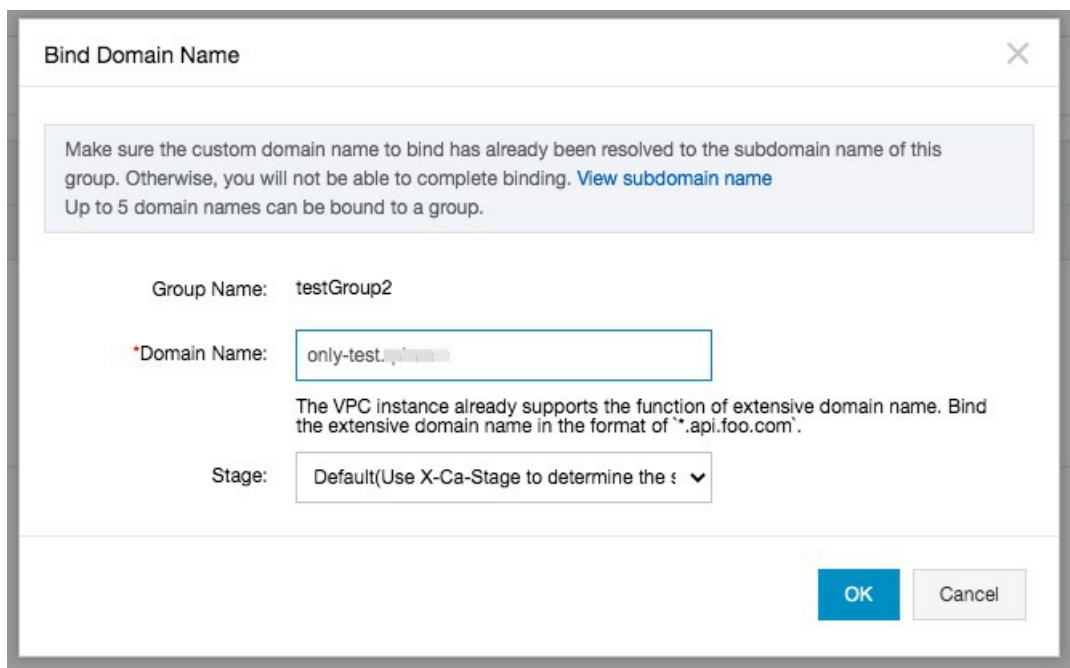
Perform the following steps to use a custom domain to call APIs over HTTPS:

1. Bind a custom domain to a specific API group.

- i. Log on to the API Gateway console. In the left-side navigation pane, choose Publish APIs > API Groups. On the Group List page, click the name of the target group to go to the Group Details page.
- ii. In the Custom Domain Name section, click **Bind Domain**.



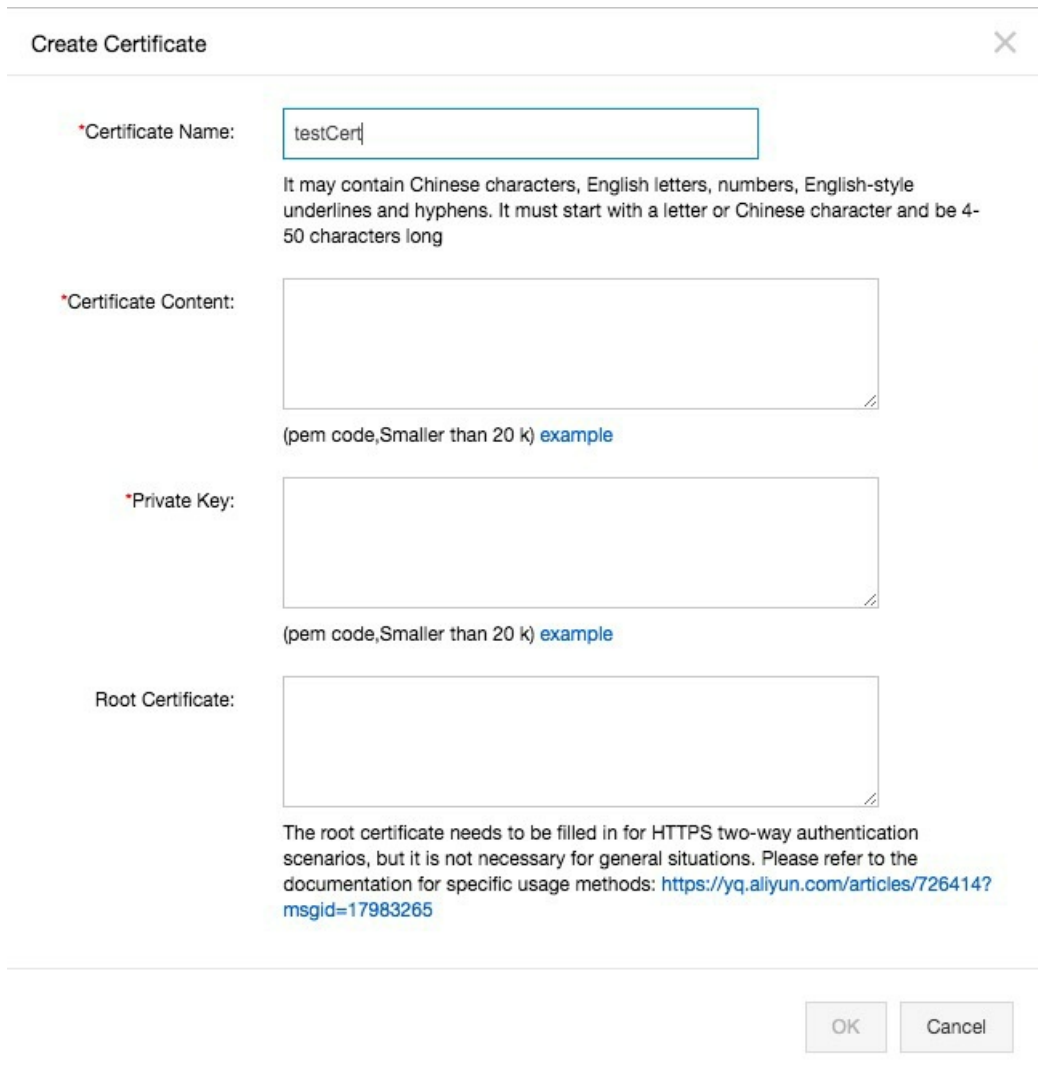
- iii. In the Bind Domain Name dialog box, specify Domain Name and click **OK**.



2. Resolve the domain name. To access API Gateway by using this domain name, you must resolve the domain name in Apsara Stack to the virtual IP address of API Gateway. Run the ping [Second-level domain] command to obtain the virtual IP address of API Gateway.
3. Upload a domain name certificate. After a domain name is bound to an API group, you can use the domain name to call all the APIs that belong to this API group over HTTP. If you want to call APIs over HTTPS, you must upload an SSL certificate for the domain name. You must prepare the certificate yourself and upload the certificate to API Gateway.
  - i. In the **Custom Domain Name** section of the **Group Details** page, find the target domain name and click **Create Certificate** in the SSL Certificate column.



- ii. In the Create Certificate dialog box, specify **Certificate Name**, **Certificate Content**, and **Private Key**. Then click OK.



The image shows a 'Create Certificate' dialog box with a title bar and a close button. It contains four input fields: 'Certificate Name' (with a value 'testCert'), 'Certificate Content', 'Private Key', and 'Root Certificate'. Each field has a text area and a small icon in the bottom right corner. Below the 'Certificate Content' and 'Private Key' fields, there is a note: '(pem code, Smaller than 20 k) example'. Below the 'Root Certificate' field, there is a note: 'The root certificate needs to be filled in for HTTPS two-way authentication scenarios, but it is not necessary for general situations. Please refer to the documentation for specific usage methods: <https://yq.aliyun.com/articles/726414?msgid=17983265>'. At the bottom right, there are 'OK' and 'Cancel' buttons.

**Note** If the certificate is a self-signed certificate, ignore certificate verification when you call APIs.

## 2.8. FAQ

### 2.8.1. How do I obtain error information?

API Gateway returns a response to the client after it receives a request.

You must check the response headers that start with X-Ca. Take note of the following points:

// The unique ID of the request. When API Gateway receives a request, it generates a request ID and returns the request ID to the client in the X-Ca-Request-Id header. We recommend that you record the request ID in both the client and your backend service for troubleshooting and tracking.

X-Ca-Request-Id: 7AD052CB-EE8B-4DFD-BBAF-EFB340E0A5AF

// The error message returned by API Gateway. If a request fails, API Gateway returns the error message to the client in the X-Ca-Error-Message header.

X-Ca-Error-Message: Invalid Parameter Required `field1`

// The error code of a system error in API Gateway. If a request is blocked by API Gateway due to an error, API Gateway returns the corresponding error code in the X-Ca-Error-Code header. Instances of the classic network type do not have this header.

X-Ca-Error-Code: I400MP

The `X-Ca-Error-Code` and `X-Ca-Error-Message` headers help you identify the error cause. The

`X-Ca-Request-Id` header helps you query request logs in Log Service. You can also provide the request ID included in the X-Ca-Request-Id header for technical support personnel to check log information and resolve issues.

For more information about `X-Ca-Error-Code`, see [Error codes](#).

## 2.8.2. Error codes

- If the client receives a response in which the `X-Ca-Error-Code` header is not empty, the header is returned by API Gateway. An error code is six characters in length. For more information, see the following table. `X-Ca-Error-Message` indicates detailed information about an error message.
- If the `X-Ca-Error-Code` header is empty, the HTTP error code is generated by your backend service. API Gateway transparently transmits the error message from your backend service.

Error code	HTTP status code	Error message	Description
I400HD	400	Invalid Header `\${HeaderName}` \${Reason}	The error message returned because the HTTP request header is invalid.
I400MH	400	Header `\${HeaderName}` is Required	The error message returned because the HTTP request header is missing.
I400BD	400	Invalid Body: \${Reason}	The error message returned because the HTTP request body is invalid.

Error code	HTTP status code	Error message	Description
I400PA	400	Invalid Request Path `\${Reason}`	The error message returned because the HTTP request path is invalid.
I405UM	405	Unsupported Method `\${Reason}`	The error message returned because the HTTP request method is not supported.
I400RU	400	Invalid Request Uri `\${Reason}`	The error message returned because the HTTP request URL is invalid.
I403PT	403	Invalid protocol `\${Protocol}` unsupported	The error message returned because a protocol that is not supported in API configurations is used. Check the API configurations.
I413RL	413	Request body too Large	The error message returned because the request body is too large.
I413UL	413	Request URL too Large	The error message returned because the request URL is too long.
I400CT	400	Invalid Content-Type: `\${Reason}`	The error message returned because the Content-Type setting is invalid.
I404DO	404	Invalid Domain `\${DomainName}`	The error message returned because the domain name is unknown.
I410GG	410	Group's instance invalid	The error message returned because an invalid instance is requested. The group may not belong to the current instance.

Error code	HTTP status code	Error message	Description
I400SG	400	Invalid Stage	The error message returned because an unknown environment is requested.
I404NF	404	API not found \${Reason}	The error message returned because the corresponding API is not found based on the Path and Method settings of the request.
X400PM	400	Invalid plugin meta \${PluginName} \${Reason}	The error message returned because the metadata of the plugin is invalid. Submit a ticket to contact customer service.
X500ED	500	Expired api definition	The error message returned because the specified API metadata definition is invalid. Submit a ticket to contact customer service.
X500AM	500	Invalid Api Meta, try deploy again or contact us via ticket	The error message returned because the specified API metadata definition is invalid. Submit a ticket to contact customer service.
X403DG	403	Bad Domain or Group: \${Reason}	The error message returned because the grouped data is invalid. Submit a ticket to contact customer service.

Error code	HTTP status code	Error message	Description
B451DO	451	Unavailable Domain for Legal Reasons	The error message returned because the domain name does not comply with the requirements of relevant laws and regulations.
B451GO	451	Unavailable Group for Legal Reasons	The error message returned because the group does not comply with the requirements of relevant laws and regulations.
B403OD	403	Provider Account Overdue	The error message returned because the API provider has overdue payments.
A400AC	400	Invalid AppCode \${Reason}	The error message returned because the corresponding AppCode is not found when you perform an authorization in AppCode mode.
A400IK	400	Invalid AppKey	The error message returned because the corresponding AppKey setting is not found when you perform an authorization by using a key-secret pair.
A403IS	403	Invalid Signature, Server StringToSign: `\${StringToSign}`	The error message returned because the signature is invalid. For more information, see Request signatures.
A403EP	403	App authorization expired	The error message returned because the authorization expired.
A403PR	403	Plugin Authorization Needed	The error message returned because plugin authorization is not performed.

Error code	HTTP status code	Error message	Description
A400MA	400	Need authorization, `X-Ca-Key` or `Authorization: APPCODE ...` is required	The error message returned because authorization is not performed in AppCode mode or by using a key-secret pair.
I400I5	400	Invalid Content-MD5 \${Reason}	The error message returned because Content-MD5 is invalid.
I400NC	400	X-Ca-Nonce is required	The error message returned because the X-Ca-Nonce header is not provided after you select Force Nonce Check (Anti Replay by X-Ca-Nonce).
S403NU	403	Nonce Used	The error message returned because a replay attack is detected. The X-Ca-Nonce header in the request is repeated.
S403TE	403	X-Ca-Timestamp is expired	The error message returned because the timestamp specified by the X-Ca-Timestamp header expired.
I400MP	400	Parameter `\${ParameterName}` is required	The error message returned because the required parameter is not specified in the API configuration.
I400IP	400	Invalid parameter `\${ParameterName}` \${Reason}	The error message returned because the value of the parameter that is specified in the API configuration is invalid.
I400JR	400	JWT required	The error message returned because no JWT-related parameters are found.

Error code	HTTP status code	Error message	Description
S403JI	403	Claim `jti` is required when `preventJtiReplay:true`	The error message returned because no valid jti claim is included in the request when preventJtiReplay is set to true in a JWT authentication plugin.
S403JU	403	Claim `jti` in JWT is used	The error message returned because the jti claim that is included in the request is used when preventJtiReplay is set to true in a JWT authentication plugin.
I400JD	400	JWT Deserialize Failed: `\${Token}`	The error message returned because the JWT that is read from the request failed to be parsed.
A403JT	403	Invalid JWT: \${Reason}	The error message returned because the JWT that is included in the request is invalid.
A403JK	403	No matching JWK, `\${kid}` not found	The error message returned because no JWK matches kid configured in the JWT included in the request.
A403JE	403	JWT is expired at `\${Date}`	The error message returned because the JWT that is read from the request expired.
I400JP	400	Invalid JWT plugin config: \${JWT}	The error message returned because the JWT authentication plugin is incorrectly configured.
A403OL	403	OAuth2 Login failed: \${Reason}	
A403OU	403	OAuth2 Get User Info failed: \${Reason}	
A401OT	401	Invalid OAuth2 Access Token	

Error code	HTTP status code	Error message	Description
A401OM	401	OAuth2 Access Token is required	
T429ID	429	Throttled by INNER DOMAIN Flow Control, \${Domain} is a test domain, only 1000 requests per day	The error message returned because the number of requests initiated has exceeded the upper limit allowed for a default second-level domain. To increase the quota, use your own domain name.
T429IN	429	Throttled by INSTANCE Flow Control	The error message returned because throttling is performed for the current instance.
T429GR	429	Throttled by GROUP Flow Control	The error message returned because throttling is performed for the current group.
T429PA	429	Throttled by API Flow Control	The error message returned because the default API-level throttling policy defined in the throttling plugin is used.
T429PR	429	Throttled by PLUGIN Flow Control	The error message returned because the special throttling policy defined in the throttling plugin is used.
T429UP	429	Throttled by Usage Plan Flow Control	The error message returned because throttling is performed for the usage plan.
T429SR	429	Throttled by SERVER Flow Control	
T429MR	429	Too Many Requests, throttle by `\${Description}`	
A403IP	403	Access denied by IP Control Policy	The error message returned because access is denied by the IP address-based access control plugin.

Error code	HTTP status code	Error message	Description
A403IN	403	Access from internet is disabled \${Reason}	The error message returned because you are not allowed to call APIs or access API groups over the Internet.
A403VN	403	Access from invalid VPC is disabled	The error message returned because access over a VPC is denied.
A403AC	403	Access Control Forbidden by \${RuleName}	The error message returned because access is denied by the access control plugin.
A403CO	403	Cross origin resource forbidden \${Domain}	The error message returned because access is denied by the CORS plugin.
I404CO	404	Cross origin resource not found \${Method} - \${Path}	The error message returned because the API definition is not found based on the Path and Method settings of the request that is pre-checked by the CORS plugin.
I404CH	404	Content not cached, with `Cache-Control: only-if-cached`	
I404NR	404	\${Resource} not found	
I404SR	404	Stage route missing: \${Reason}	
B403MO	403	Api Market Subscription overdue	The error message returned because the API provider has overdue payments.

Error code	HTTP status code	Error message	Description
B403MQ	403	Api Market Subscription quota exhausted	The error message returned because the API quota you purchased in Alibaba Cloud Marketplace has been exhausted.
B403ME	403	Api Market Subscription expired	The error message returned because the API subscription relationship expired.
B403MI	403	Api Market Subscription invalid	The error message returned because the API marketplace subscription relationship is invalid.
D504RE	504	Backend domain `\${Domain}` resolve failed	The error message returned because the domain name failed to be resolved at the backend.
D504IL	504	Backend domain `\${Domain}` resolve to illegal address `\${Address}`	The error message returned because the domain name resolution results are invalid at the backend.
D504CO	504	Backend service connect failed `\${Reason}`	The error message returned because the backend connection failed. Check the security group configurations, the startup status of the backend server, and firewall configurations.
D504CS	504	Backend http ssl connect failed `\${Reason}`	The error message returned because the backend connection over HTTPS failed. Check whether the backend protocol matches the port.

Error code	HTTP status code	Error message	Description
D504TO	504	Backend service request timeout	The error message returned because the backend request timed out.
X504VE	504	Backend service vpc mapped failed	The error message returned because the VPC mapping at the backend is invalid. Submit a ticket to contact customer service.
D503BB	503	Backend circuit breaker busy	The error message returned because the API is protected by its circuit breaker.
D503CB	503	Backend circuit breaker open, \${Reason}	The error message returned because the circuit breaker is open for the API. Check the backend performance of the API.
I508LD	508	Loop Detected	The error message returned because loopback call is detected.
I404DD	404	Device id \${DeviceId} not found	The error message returned because the device ID is not found when you call APIs over WebSocket.
A403FC	403	Function Compute AssumeRole failed \${RequestId}: \${Reason}	The error message returned because an authorization error occurs when Function Compute serves as the backend service.

Error code	HTTP status code	Error message	Description
D502FC	502	Function Compute response invalid: \${Reason}	The error message returned because responses from the backend service of the Function Compute type are invalid.
X500ER	500	Service Internal Error	The error message returned because an internal server error occurred. Submit a ticket to contact customer service.
X503BZ	503	Service Busy	The error message returned because the service is busy in API Gateway. Try again later or submit a ticket to contact customer service.
X504TO	504	Service timeout	The error message returned because the service processing timed out in API Gateway.

*Some error codes may change with version updates or the addition of new features.*