

# Cloud Service Engine

## User Guide

**Issue**            01  
**Date**             2026-01-06



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. All rights reserved.**

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

## **Trademarks and Permissions**



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

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

## **Notice**

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

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

## **Huawei Cloud Computing Technologies Co., Ltd.**

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

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

---

# Contents

---

<b>1 Service Overview</b>	<b>1</b>
1.1 What Is CSE?	1
1.2 Scenarios	4
1.3 Functions	6
1.4 Glossary	8
1.5 Version Support by ServiceComb Engines	11
1.6 Specifications	12
1.7 Restrictions	13
1.8 Permissions	20
1.9 Relationships Between CSE and Other Services	26
<b>2 Getting Started</b>	<b>27</b>
2.1 Connecting Spring Cloud Applications to ServiceComb Engines	27
2.2 Connecting Spring Cloud Applications to Nacos Engines	31
<b>3 Before You Start</b>	<b>35</b>
<b>4 Creating a User and Granting CSE Permissions</b>	<b>37</b>
<b>5 ServiceComb Engines</b>	<b>40</b>
5.1 ServiceComb Engine Overview	40
5.2 Creating a ServiceComb Engine	41
5.3 Managing ServiceComb Engines	44
5.3.1 Viewing ServiceComb Engine Information	44
5.3.2 Managing ServiceComb Engine Tags	47
5.3.3 Managing Public Network Access for a ServiceComb Engine	49
5.3.4 Managing Security Authentication for a ServiceComb Engine	50
5.3.5 Configuring Backup and Restoration of a ServiceComb Engine	52
5.3.6 Upgrading a ServiceComb Engine	54
5.3.7 Changing ServiceComb Engine Specifications	55
5.3.8 Viewing ServiceComb Engine Operation Logs	55
5.3.9 Deleting a ServiceComb Engine	56
5.4 Viewing Microservice Running Metrics Through the Microservice Dashboard	56
5.5 Managing Microservices	57
5.5.1 Viewing an Application	57
5.5.2 Microservice Management	58

5.5.3 Instance Management.....	66
5.6 Service Scenario Governance (Applicable to ServiceComb Engine 2.x).....	68
5.6.1 Service Scenario Governance Overview.....	68
5.6.2 Creating a Service Scenario.....	69
5.6.3 Creating a Governance Policy.....	70
5.7 Microservice Governance (Applicable to ServiceComb Engine 1.x and 2.4.0+).....	74
5.7.1 Microservice Governance Overview.....	74
5.7.2 Configuring a Load Balancing Policy.....	76
5.7.3 Configuring a Rate Limiting Policy.....	77
5.7.4 Configuring a Service Degradation Policy.....	78
5.7.5 Configuring a Fault Tolerance Policy.....	79
5.7.6 Configuring a Circuit Breaker Policy.....	81
5.7.7 Configuring a Fault Injection Policy.....	83
5.7.8 Configuring Blacklist and Whitelist.....	84
5.7.9 Configuring Public Key Authentication.....	85
5.8 Configuration Management (Applicable to ServiceComb Engine 2.x).....	85
5.8.1 Creating a Configuration for ServiceComb Engine 2.x.....	85
5.8.2 Managing Configurations for ServiceComb Engine 2.x.....	89
5.9 Configuration Management (Applicable to ServiceComb Engine 1.x).....	94
5.9.1 Creating a Configuration for ServiceComb Engine 1.x.....	94
5.9.2 Managing Configurations for ServiceComb Engine 1.x.....	95
5.10 System Management.....	97
5.10.1 Overview.....	97
5.10.2 Accounts.....	98
5.10.3 Roles.....	102
<b>6 Nacos Engines.....</b>	<b>108</b>
6.1 Nacos Engine Overview.....	108
6.2 Creating a Nacos Engine.....	110
6.3 Managing Nacos Engines.....	112
6.3.1 Viewing Nacos Engine Information.....	112
6.3.2 Managing Nacos Engine Tags.....	113
6.3.3 Managing the Nacos Engine Whitelist.....	114
6.3.4 Increasing Nacos Engines.....	115
6.3.5 Upgrading a Nacos Engine.....	116
6.3.6 Deleting a Nacos Engine.....	116
6.4 Managing Namespaces.....	117
6.5 Permission Control.....	118
6.5.1 Permission Control Overview.....	118
6.5.2 Enabling and Disabling Security Authentication.....	119
6.5.3 Accounts.....	119
6.5.4 Roles.....	121
6.5.5 Console Resource Management.....	122

6.6 Managing Nacos Engine Services.....	122
6.7 Managing Nacos Engine Configurations.....	124
6.7.1 Nacos Engine Configuration Overview.....	125
6.7.2 Creating a Nacos Engine Configuration.....	125
6.7.3 Managing Nacos Engine Configurations.....	127
6.7.4 Managing Dark Launch of Nacos Engine Configurations.....	130
6.7.5 Managing Historical Nacos Engine Versions.....	131
6.7.6 Using the Listening and Query Function of the Nacos Engine.....	132
6.8 Viewing Monitoring of a Nacos Engine.....	133
<b>7 Key Operations Recorded by CTS.....</b>	<b>135</b>
7.1 CSE Operations That Can Be Recorded by CTS.....	135
7.2 Viewing CTS Traces in the Trace List.....	138
<b>8 FAQs.....</b>	<b>143</b>
8.1 Precautions When Using CSE.....	143
8.1.1 Why Can't I See Information About Cloud Services?.....	143
8.1.2 What Should I Do If an Agency Creation Failed?.....	143
8.2 Nacos Engines.....	144
8.2.1 Why Do I Get an Error When I Request gRPC?.....	144
8.3 ServiceComb Engines.....	144
8.3.1 How Can I Handle a Certificate Loading Error?.....	144
8.3.2 What If the Header Name Is Invalid?.....	145
8.3.3 What Is the Performance Loss of Mesher?.....	145
8.3.4 Why Is "Version validate failed" Displayed When I Attempt to Connect to the Service Center?.....	146
8.3.5 Why Is "Not enough quota" Displayed When I Attempt to Connect to the Service Center?.....	146
8.3.6 What Is Service Name Duplication Check?.....	147
8.3.7 Why Do I Have to Define Service Contracts?.....	147
8.3.8 Why Are Microservice Development Framework and Netty Versions Unmatched?.....	147
8.3.9 Why Is "Duplicate cluster name" Displayed?.....	148
8.3.10 Error Message "the subnet could not be found" Is Displayed When the Access Address Fails to Be Processed During Engine Creation.....	148
8.3.11 What Should I Do If SpringCloud Applications Fail to Connect to the Configuration Center of ServiceComb Engine 2.x?.....	149
8.3.12 Why Could My the Global Configuration Not Be Modified?.....	149
8.3.13 Obtain Configurations Failed.....	150

# 1 Service Overview

---

## 1.1 What Is CSE?

Cloud Service Engine (CSE) is cloud middleware for microservice applications. It provides users with high-performance and high-resilience enterprise-class cloud service capabilities, such as registry and discovery, configuration management, and service governance. Furthermore, it is seamlessly compatible with open-source ecosystems such as Spring Cloud, ServiceComb, and Dubbo. Users can use other cloud services to quickly build a cloud-native microservice system to implement quick development and high O&M of microservice applications.

### Nacos Engine

CSE Nacos is compatible with open-source Nacos and Eureka clients. It provides registry and discovery, dynamic configuration management, access permission control, and observability. It can be used to build highly available and easy-to-manage microservice middleware.

### ServiceComb Engine

CSE ServiceComb engines are developed based on the Apache ServiceComb open-source ecosystem and provide a one-stop microservice platform. You can use Java-Chassis SDK, SpringCloudHuawei SDK, or non-intrusive Sermant Java Agent (standard Spring Cloud and Dubbo frameworks supported) to access the platform. After accessing the platform, you can easily use functions such as service contract, service governance, dark launch, service governance, service monitoring, configuration management, and access control, practice API-first development, and build secure, high-performance, and stable microservice applications. For details about Apache ServiceComb Service Center, see the following:

- <https://github.com/apache/servicecomb-service-center/>
- <https://service-center.readthedocs.io/en/latest/user-guides.html>

ServiceComb engine has two versions: 1.x and 2.x.

ServiceComb 2.x engines are commercial engines that manage large-scale microservice applications. You can select different engine specifications based on

service requirements. Exclusive engines are exclusively used; therefore, the performance is not affected by other tenants.

Compared with ServiceComb engine 1.x, the underlying architecture, functions, security, and performance of ServiceComb engine 2.x are upgraded, providing an independent service registry and discovery center and configuration center, and supports custom service scenarios and governance. [Table 1-1](#) lists features supported in CSE 1.0 and CSE 2.0.

**Table 1-1** Comparison between ServiceComb engine 2.x and ServiceComb engine 1.x

Feature	Sub-feature		2.x	1.x	Remarks
Engine management	Security	Security authentication	√	√	-
	Reliability	3-AZ high reliability	√	√	-
Microservice management	Basic capability	Registry and discovery	√	√	-
		Multi-frame access	√	√	Supports Spring Cloud and ServiceComb Java Chassis.
		Automatic clear of versions without instances	√	x	The latest three microservice versions are retained for version 2.3.7 and later, and the versions without instances are automatically cleared.
	Performance	Millisecond-level push of instance changes	√	√	-
Configuration management	Basic capability	Management and configuration	√	√	-
		Diversified configuration formats	√	Only text is supported.	2.x supports YAML, JSON, TEXT, Properties, INI and XML.
		Import and export	√	√	2.x supports configuration import of the same policy.

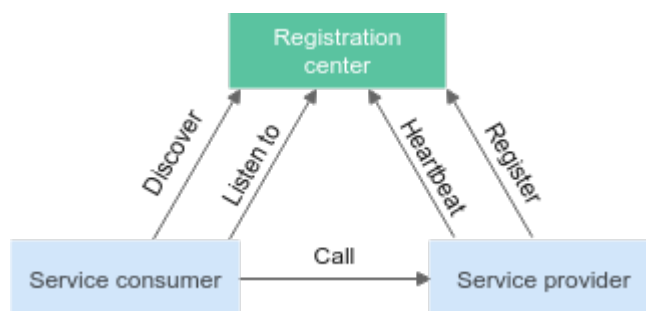
Feature	Sub-feature		2.x	1.x	Remarks
	Advanced features	History version management	√	x	-
		Version comparison	√	x	-
		Fast rollback	√	x	-
		Configuration labels	√	x	-
	Performance	Second-level delivery	√	x	-
Microservice governance	Scenario-based service governance	Custom service scenario	√	x	-
		Matching rule based on the request method	√	x	-
		Matching rule based on the request path	√	x	-
		Matching rule based on request headers	√	x	-
	Governance policy: rate limiting	Token bucket rate limiting on the server	√	√	-
	Governance policy: retry	The client performs retry to ensure the availability, fault tolerance, and consistency of user services.	√	√	-
	Governance policy: circuit breaker	The server breaks faulty services to prevent large-scale faults.	√	√	-

Feature	Sub-feature		2.x	1.x	Remarks
	Governance policy: repository isolation	The server controls the request concurrency capability based on the semaphore.	√	x	-
Development tool	Local lightweight engine	One-click local startup, facilitating offline microservice development	√	√	-

## 1.2 Scenarios

### Microservice Registry and Discovery

The microservice architecture should solve the communication problem between microservices. Compared with the traditional communication bus and LB solutions, the registry and discovery mechanism implements load balancing on clients and has advantages in communication efficiency and elasticity. CSE provides a highly available, stable, and O&M-free service registration center for development frameworks such as Spring Cloud, Dubbo, and ServiceComb.



The microservice registry and discovery mechanism is implemented through the service registration center.

- Service registration:** When a microservice instance starts, it sends a registration request to the service registration center to register its metadata (such as the service name, IP address, port number, and version number) with the registration center. The registration center stores the information in an internal data structure for subsequent query. The service provider sends heartbeat messages containing health information such as the service running status, workload, and resource consumption to the registration center at a fixed interval to notify the registration center that the service provider is active. The registration center receives and records heartbeat messages, and

determines the service availability based on the message status and content. If no heartbeat message is received within a specified period, the registration center marks the service instance as unhealthy. If no heartbeat message is received after the timeout period, the registration center removes the instance from the available service list. When the service instance resumes sending heartbeat messages, it can be re-registered.

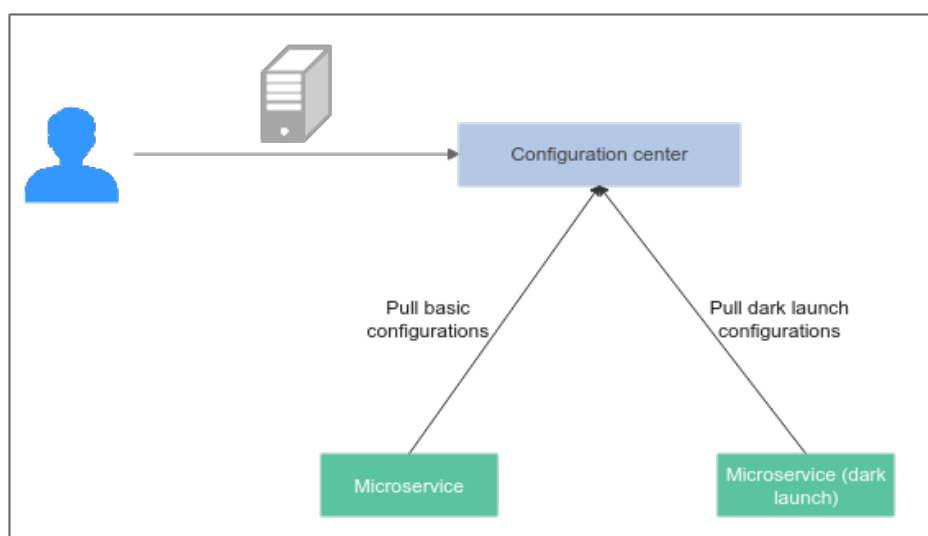
- **Service discovery:** Service consumers directly interact with the registration center to obtain service provider's instance information and implement load balancing locally. In the microservice architecture, service provider instances may change due to various reasons (such as scale-in, scale-out, and faults). Service consumers can listen to the registration center to detect these changes in real time and update the local service instance list in a timely manner to ensure that available service providers can be accurately found during subsequent service calling.

## Dynamic Configuration Management

The configuration center provides centralized configuration management to implement differentiated configuration for different environments, clusters, and instances. Configuration changes are dynamically made during running, which is more efficient and standard than those made through the traditional configuration file mode.

The configuration center can be used in the following ways based on its usage:

- As a deployed environment configuration, it is integrated with the deployment delivery service, for example, the application.yaml file of the common Spring Cloud service to carry configuration information such as the application data source and access private key.
- As an O&M parameter configuration, it is integrated with the O&M service, for example, to dynamically adjust the log level and the number of connection pools.
- As a service parameter configuration, for example, it dynamically changes the product price discount, bulletin, and winning rate.



During startup, a microservice sends a request to the configuration center to obtain its configuration information. The configuration center reads the

corresponding configuration data from the storage based on the microservice request and returns the data to the microservice. Then, the microservice parses the received configuration information and loads it to the memory for the application to use.

In the configuration center, administrators define dedicated dark launch configuration items based on dark launch requirements. These configuration items are distinguished from normal production configurations and can be associated with specific identifiers (such as dark launch version numbers and user group identifiers).

1. Upon startup, the microservice pulls basic configurations from the configuration center and carries its own identifier (such as the service name and version number) and dark launch identifier (if the dark launch group to which the microservice belongs has been determined).
2. After receiving the microservice request, the configuration center determines whether the microservice is in the dark launch range based on the identifier carried. If yes, the configuration center searches the dark launch configuration storage area for the corresponding dark launch configuration and returns the configuration. If no, the configuration center returns the normal production configuration.
3. During running, the microservice dynamically pulls the updated dark launch configurations based on the notification mechanism (such as persistent connection push and scheduled polling) of the configuration center. When the dark launch configuration changes, the configuration center notifies related microservices in a timely manner. The microservices pull and apply the new dark launch configurations.

## 1.3 Functions

Cloud Service Engine (CSE) is a cloud middleware service for microservice applications. It supports ServiceComb engines contributed to Apache, open-source enhanced Nacos engines. You can also use other cloud services to quickly build a cloud-native microservice system, implementing quick development and high-availability O&M of microservice applications.

### Nacos Engine

- Managing Nacos engines  
On the CSE console, you can view the Nacos engine information, increase Nacos engine instances, upgrade the Nacos engine, manage Nacos engine tags, and delete Nacos engines. For details, see [Managing Nacos Engines](#).
- Using Nacos engines
  - Namespaces isolate configurations in different environments. For example, resources (such as configurations and services) in the development and test environments are isolated from those in the production environment. Different namespaces can have the same group or data ID. For details, see [Managing Namespaces](#).
  - You can use the CSE console to manage services registered with Nacos. For details, see [Managing Nacos Engine Services](#).
  - The CSE console manages configurations of services registered with Nacos. For details, see [Managing Nacos Engine Configurations](#).

## ServiceComb Engine

- Managing ServiceComb engines

On the CSE console, you can view the ServiceComb engine information, set ServiceComb engine backup and restoration, manage public network access of ServiceComb engines, upgrade ServiceComb engine versions, change ServiceComb engine specifications, manage ServiceComb engine security authentication, manage ServiceComb engine tags, and delete ServiceComb engines. For details, see [Managing ServiceComb Engines](#).

- Using ServiceComb engines
  - The service center of the ServiceComb engine provides the service registry function that registers basic information, such as the application to which a microservice belongs, microservice name, microservice version, and listening address, with the service center when the microservice is started.
  - The configuration center is used to manage microservice application configurations. Microservices connect to the configuration center to obtain the information and changes of configurations. The configuration center is also the core component for managing microservices. For example, service governance rules are delivered through the configuration center.
  - Unified request marker-based governance is provided based on dynamic configurations for different microservice development frameworks, such as Spring Cloud, Java Chassis, and Dubbo. Microservice governance of ServiceComb engines can be used by introducing related governance components to the development frameworks.
  - In dark launch, a small number of users test the trial version, ensuring smooth rollout of new features. Once new features become mature, a formal version is released for all users. Dark launch ensures stability of the entire system. During initial dark launch, problems can be detected and fixed.
  - Dashboard provides basic capabilities for monitoring microservice running. Microservices report running status using SDKs. The reported data includes request statistics, such as the quantity, latency, and error rate, as well as governance-related status, such as the circuit breaker status.
  - A ServiceComb engine may be used by multiple users. Different users must have different engine access and operation permissions based on their responsibilities and permissions.

## 1.4 Glossary

### General

Concept	Description
Microservice	<p>Microservice is a service concept, that is, a process that provides a service.</p> <p>Each service has its own service functions. APIs that are not restricted by languages are open to other systems (HTTP frequently used).</p> <p>Multiple microservices form an application.</p>
Instance	<p>An instance is the minimum running and deployment unit of a microservice. Generally, it corresponds to an application process. A microservice can be deployed in multiple containers or VMs to enable multiple instances to run at the same time.</p>
Configuration	<p>The configuration in the microservice scenario is to control the values of some variables in the program code. For example, dynamic configuration is to dynamically change the values of some variables during microservice running.</p>

### Glossary (Nacos Engine)

Concept	Description
Namespace	<p>Used for tenant-level configuration isolation. Namespaces isolate configurations in different environments. For example, resources (such as configurations and services) in the development and test environments are isolated from those in the production environment.</p>
Configuration set	<p>A set of configuration items is called a configuration set. Generally, a configuration file is a configuration set that contains all system configurations.</p>

Concept	Description
Configuration set ID	ID of a configuration set in Nacos. A system or application can contain multiple configuration sets, and each one can be identified by a name.
Group	Group of configuration sets in Nacos. It is one of the dimensions according to which configurations are organized. The configuration sets are always grouped by a meaningful string to differentiate the configuration sets with the same Data ID. When you create a configuration on Nacos, the group name is replaced by DEFAULT_GROUP by default if not specified.
Protection threshold	Protect threshold is related to the proportion of healthy instances in a cluster. When the proportion of healthy instances to the total instance is smaller than this value, all instances are returned to the client regardless of the health of the instance. If the protect threshold is not triggered, Nacos returns only healthy instances to the client.
Dark launch	Before the configuration is officially released, a small part of the configuration is released and verified. If this part is correct, the whole configuration will be officially released, reducing the risk.
Weight	Instance-level configuration. Weight is a floating-point number. The greater the weight, the greater the traffic that the instance expects to be allocated.
Metadata	Description of Nacos data (such as configurations and services), such as the service version and weight. From the scope of action, it is divided into meta-information of service level, meta-information of virtual cluster, and meta-information of instance.

## Glossary (ServiceComb Engine)

Concept	Description
Version	In the microservice scenario, a version is used to mark the iteration record of a microservice to facilitate management of different iterations of a microservice.
Service contract	<p>A service contract in the microservice scenario is a microservice API constraint specification based on the OpenAPI definition and defines APIs on the server and consumer.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>By default, service contract is used in Java chassis.</li> <li>By default, service contract is not used in Spring Cloud. If it is used, the following dependencies need to be introduced:</li> </ul> <pre>&lt;dependency&gt;   &lt;groupId&gt;com.huaweicloud&lt;/groupId&gt;   &lt;artifactId&gt;spring-cloud-starter-huawei-swagger&lt;/artifactId&gt; &lt;/dependency&gt;</pre>
Application	A software system that completes a complete service scenario. An application consists of multiple microservices, which can discover and call each other.
Environment	A logical concept established by the service center, which can be development or production. Microservice instances in different environments are logically isolated and cannot be discovered or called by each other.
Governance policy	A concept in microservice governance. It refers to a method used for governance. Each governance policy can be bound to a service scenario. A policy cannot be bound to multiple service scenarios. Different governance policies can be bound to the same service scenario.
Service scenario	A condition for a governance policy to take effect. A service scenario can be bound to multiple governance policies.

## 1.5 Version Support by ServiceComb Engines

This section describes the versions supported by ServiceComb engines.

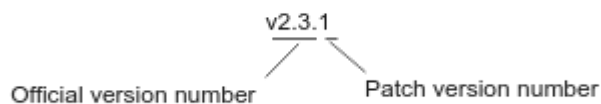
### Version Description

The version number format is {major}.{minor}.{patch},

where,

- {major}.{minor} indicates the official version number.
- {patch} indicates the patch version number.

For example, v2.3.1. 2.3 is the official version number, and 1 is the patch version number.



### Version Support

- Engine creation  
Only the ServiceComb engine of the latest version can be created. The ServiceComb engine of a specified version cannot be created.
- Engine maintenance  
The latest three official versions can be maintained. For other versions, Huawei will no longer provide technical support, including new functions, bug fixing, vulnerability fixing, and upgrades.
- Engine upgrade
  - Official version upgrade: Two earlier versions among the latest three official versions can be upgraded to the latest version. For example, if the latest three official versions are 2.3, 2.2, and 2.1, 2.1 and 2.2 can be upgraded to 2.3.

#### NOTE

If the engine upgrade is not supported, for example, from 2.0 to 2.3, the management function of ServiceComb engines may be unavailable. Exercise caution when performing this operation.

- Patch version upgrade: The CSE backend provides automatic patch version upgrade, for example, from 2.3.0 to 2.3.1.

### Version Constraints

Version rollback is not supported after the microservice engine version is upgraded.

## 1.6 Specifications

### Nacos engine instance specifications

You can select proper Nacos instance specifications as required.

**Table 1-2** Nacos engine instance specifications

Maximum Number of Microservice Instances	Capacity Unit	Consumer Connections	Number of configuration files
500	10	1,000	1,000
1,000	20	2,000	2,000
2,000	40	4,000	4,000

 **NOTE**

One capacity unit = 50 microservice instances

### ServiceComb engine instance specifications

You can select ServiceComb engine instance specifications based on the number of microservice instances to be hosted. For details, see [Table 1-3](#).

As shown in [Table 1-3](#), ServiceComb engine instances with different microservice instances are rewarded with corresponding configuration items and the maximum number of microservice versions supported.

**Table 1-3** ServiceComb Engine Instance Specifications

Microservice Instances	Configuration Items
100	600
200	600
500	3,000
2,000	12,000

## 1.7 Restrictions

### Relationship Between the Nacos Engine and Microservice Framework

Nacos Engine Version	Spring Cloud Alibaba Version	Spring Cloud Version	Spring Boot Version
2.1.0.x	2022.0.0.0-RC*	Spring Cloud 2022.0.0	3.0.0
	2021.0.4.0*	Spring Cloud 2021.0.4	2.6.11
	2021.0.1.0	Spring Cloud 2021.0.1	2.6.3
	2021.1	Spring Cloud 2020.0.1	2.4.2
	2.2.10-RC1*	Spring Cloud Hoxton.SR12	2.3.12.RELEASE
	2.2.9.RELEASE	Spring Cloud Hoxton.SR12	2.3.12.RELEASE
	2.2.8.RELEASE	Spring Cloud Hoxton.SR12	2.3.12.RELEASE
	2.2.7.RELEASE	Spring Cloud Hoxton.SR12	2.3.12.RELEASE
	2.2.6.RELEASE	Spring Cloud Hoxton.SR9	2.3.2.RELEASE
	2.2.1.RELEASE	Spring Cloud Hoxton.SR3	2.2.5.RELEASE
	2.2.0.RELEASE	Spring Cloud Hoxton.RELEASE	2.2.X.RELEASE
	2.1.4.RELEASE	Spring Cloud Greenwich.SR6	2.1.13.RELEASE
	2.1.2.RELEASE	Spring Cloud Greenwich	2.1.X.RELEASE
	2.0.4.RELEASE (Maintenance stopped. Upgrade is recommended.)	Spring Cloud Finchley	2.0.X.RELEASE

Nacos Engine Version	Spring Cloud Alibaba Version	Spring Cloud Version	Spring Boot Version
	1.5.1.RELEASE (Maintenance stopped. Upgrade is recommended.)	Spring Cloud Edgware	1.5.X.RELEASE

## Microservice Development Framework Versions

The following table lists the recommended versions of the microservice development framework.

- If you have used the microservice development framework of an earlier version to build applications, you are advised to upgrade it to the recommended version to obtain the stable and rich function experience.
- If an application has been developed using the Spring Cloud microservice development framework, you are advised to use **Spring Cloud Huawei** to access the application.
- If new microservice applications are developed based on open source and industry ecosystem components, you are advised to use the Spring Cloud framework.
- If you want to use the out-of-the-box governance capability and high-performance RPC framework provided by ServiceComb engines, you are advised to use the Java chassis framework.

Framework	Recommended Versions	Description
Spring Cloud Huawei	1.11.11-2021.0.x or later	<p>Uses <b>Spring Cloud Huawei</b> for connection.</p> <ul style="list-style-type: none"> <li>• Spring Cloud version 2021.0.9</li> <li>• Spring Boot 2.7.18</li> </ul> <p>Version description of the Spring Cloud microservice development framework: <a href="https://github.com/huaweicloud/spring-cloud-huawei/releases">https://github.com/huaweicloud/spring-cloud-huawei/releases</a></p>
Java Chassis	2.8.25 or later	<p>Uses the software package provided by the open-source project for connection without introducing third-party software packages.</p> <p>Version description of the Java chassis microservice development framework: <a href="https://github.com/apache/servicecomb-java-chassis/releases">https://github.com/apache/servicecomb-java-chassis/releases</a>.</p>

## Function Comparison Between Spring Cloud Huawei, ServiceComb, and Sermant

Level-1 Feature	Level-2 Feature	servicecomb-java-chassis	spring-cloud-huawei	sermant agent	Remarks
Microservice gateway	Rate limiting on the provider	√	√	√	-
	Server isolation warehouse	√	√	√	-
	Circuit breaker on the consumer	×	√	×	-
	Fault tolerance on the consumer	×	√	×	-
	Service downgrade on the consumer	×	×	×	-
	Fault injection on the consumer	×	×	×	-
	Load balancing	√	√	×	-
	Dark launch	×	√	√	-
	Graceful shutdown	√	√	×	-
Microservice governance	Graceful startup and shutdown	√	√	√	-
	Hitless upgrade	√	√	√	-
	Rate limiting on the provider	√	√	√	-

	Fault tolerance on the consumer	√	√	√	-
	Circuit breaker on the consumer	√	√	√	-
	Service downgrade on the consumer	√	√	√	-
	Server isolation bulkhead	√	√	√	-
	Isolation bulkhead on the consumer	√	√	√	-
	Load balancing	√	√	√	-
	Dark launch	√	√	√	-
	Full-link log tracing	√	√	×	-
	Service governance status upload	√	√	×	-
	Fail-fast	√	√	×	-
	Fault injection	√	×	√	-
	Blacklist and Whitelist	√	√	×	-
Registry and discovery	Local registry and discovery	√	√	×	-
	Single registry-CSE	√	√	√	-

	Single registry-service center	√	√	√	-
	Dual registry	×	×	√	Dual registry indicates that a service is registered with two registry centers at the same time. Currently, the sermant injector supports registry with both the CSE and the native registry center of the host.
Config uration center	ServiceComb engine	√	√	√	Configurations, such as service governance rules and service configurations, can be delivered based on the configuration center.
	Nacos engine	√	√	√	
	ServiceComb-Kie	√	√	√	
	ZooKeeper	×	×	√	
	Lightweight configuration center (zero-config)	√	×	×	
	apollo	×	×	×	
Security	Security authentication	√	√	×	Authentication between service instances and the registry center and between the consumer and provider.

Development	Multi-protocol	√	×	×	<p>Java chassis supports the following communication protocols for the consumer and provider:</p> <ul style="list-style-type: none"> <li>• Provider: JAX-RS, SpringMVC, and transparent RPC.</li> <li>• Consumer: transparent RPC, RestTemplate, and InvokerUtils.</li> </ul>
	Expansion	<ul style="list-style-type: none"> <li>• Allows users to process traffic by custom tracing.</li> <li>• Supports governance for expanded traffic.</li> </ul>	<ul style="list-style-type: none"> <li>• Supports native Spring Cloud expansion.</li> <li>• Supports governance for expanded traffic.</li> </ul>	New functions are added based on plugin development.	-

## Quotas

Quota is the maximum number of resources that can be created for engine instances. To increase the quota, click .

- [Table 1-4](#) lists the maximum number of resources that can be created in Nacos engine instances.

**Table 1-4** Nacos engine resource quota

Resource	Quota	Modifiable	Precaution
Number of namespaces in a single Nacos instance	50	No	-

Resource	Quota	Modifiable	Precaution
Size of a Nacos configuration file	100 KB	No	-
Size of a single Nacos namespace	10 MB	No	-
Bandwidth (sum of inbound and outbound bandwidths)	2 Mbit/s	No	-

- **Table 1-5** lists the maximum number of resources that can be created in ServiceComb engine instances.

**Table 1-5** Resource quota limits of ServiceComb engines

Function	Resource	Quota	Modifiable	Precaution
Microservice management	Microservice versions	10,000	No	-
	Data volume of a single instance (KB)	200	Yes	Increasing quotas prolongs the microservice discovery latency.
	Number of contracts of a single microservice	500	No	-
Configuration management	Data volume of a single configuration item (KB)	128	No	-
	Data volume of an application-level configuration	2,000	No	-
Microservice governance	Application-level governance policies	1,000	No	A maximum of 1000 governance policies are supported.

 NOTE

- A single governance policy contains governance rules and service scenarios. Governance rules and service scenarios occupy the same quota in the configuration center.
- Microservice version: In the microservice scenario, a version is used to mark the iteration record of a microservice to facilitate management of different iterations of a microservice.
- Microservice instance: An instance is the minimum running and deployment unit of a microservice. Generally, it corresponds to an application process. A microservice can be deployed in multiple containers or VMs to enable multiple instances to run at the same time.
- Configuration item: The configuration in the microservice scenario is to control the values of some variables in the program code. For example, dynamic configuration is to dynamically change the values of some variables during microservice running.

## 1.8 Permissions

If you need to assign different permissions to employees in your enterprise to access your CSE resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you securely access cloud resources.

With IAM, you can use your account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. For example, you want the software developers in your enterprise to have the permission to use CSE, but do not want them to have the permission to perform high-risk operations such as deletion. Then, you can use IAM to create users for developers and grant them only the permissions required for using CSE resources.

If your cloud account does not require individual IAM users for permissions management, you can skip this section

IAM is free of charge. You pay only for the resources in your account. For more information about IAM, see "Section Overview" in the *Identity and Access Management User Guide*.

### CSE Permissions

By default, new IAM users do not have any permissions assigned. You need to add a user to one or more user groups, and assign permission policies to the user groups. The user then inherits permissions from the user groups. This process is known as authorization. After authorization, the user can perform specified operations on CSE based on the permissions.

CSE is a project-level service deployed and accessed in specific physical regions. To assign CSE permissions to a user group, specify the scope as region-specific projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing CSE, the users need to switch to a region where they have been authorized to use this service.

You can grant users permissions by using roles and policies.

- Roles: A coarse-grained authorization mechanism provided by IAM to define permissions based on users' job responsibilities. There are only a limited

number of roles for granting permissions to users. When you grant permissions using roles, you also need to assign dependency roles. However, roles are not an ideal choice for fine-grained authorization and secure access control.

- Policies are a type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization and secure access control.

**Table 1-6** lists all the system policies supported by CSE.

**Table 1-6** CSE system permissions

Role/Policy Name	Description	Type	Dependency
CSE FullAccess	Administrator permissions for Cloud Service Engine.	System-defined policy	None
CSE ReadOnlyAccess	View permissions for Cloud Service Engine.	System-defined policy	None

If the listed permissions in **Table 1-6** do not meet actual requirements, you can create a custom a policy by referring to [Creating a Custom Policy for a Microservice Engine](#).

For details about the service permissions required by CSE functions, see **Table 1-7**.

**Table 1-7** Roles/Policies dependencies of the CSE console

Console Function	Dependency	Role/Policy Required
Engine deletion and creation	Virtual Private Cloud (VPC)	To create or delete an engine, an IAM user must be granted the VPC ReadOnlyAccess permission.
Dashboard	Application Operations Management (AOM)	To view chart data such as dashboard data, an IAM user must be granted the AOM ReadOnlyAccess permission.
Tag management	Tag Management Service (TMS)	To use TMS to add tags to the ServiceComb engine or Nacos engine, an IAM user must be granted the TMS ReadOnlyAccess permission to identify and manage ServiceComb engines or Nacos engines.
IAM user import	Identity and Access Management (IAM)	To import IAM users, the IAM ReadOnlyAccess permission is required.

**Table 1-8** lists the common operations for each system-defined policy or role of CSE. Select policies or roles as needed.

**Table 1-8** Common operations supported by each system policy

Operation	CSE ReadOnlyAccess	CSE FullAccess
Create a microservice engine	x	√
Maintain a microservice engine	x	√
Query a microservice engine	√	√
Delete a microservice engine	x	√
Create a microservice	x	√
Query a microservice	√	√
Maintain a microservice	x	√
Delete a microservice	x	√
Create microservice configurations	x	√
Query microservice configurations	√	√
Edit microservice configurations	x	√
Delete microservice configurations	x	√
Create a microservice governance policy	x	√
Query a microservice governance policy	√	√
Edit a microservice governance policy	x	√
Delete a microservice governance policy	x	√

To use a custom fine-grained policy, log in to IAM as the administrator and select fine-grained permissions of CSE as required. [Table 1-9](#) describes fine-grained permission dependencies of CSE.

**Table 1-9** Fine-grained permission dependencies of CSE

Permission	Description	Permission Dependency	Scenario
cse:engine:list	List all microservice engines	None	<ul style="list-style-type: none"> <li>View ServiceComb engine list</li> <li>View Nacos engine list</li> </ul>
cse:engine:get	View engine information	<ul style="list-style-type: none"> <li>cse:engine:list</li> </ul>	<ul style="list-style-type: none"> <li>View ServiceComb engine details</li> <li>View Nacos engine details</li> </ul>
cse:engine:modify	Modify an engine	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> </ul>	<ul style="list-style-type: none"> <li>Enable/Disable public network access to ServiceComb engines</li> <li>Enable/Manage security authentication for ServiceComb engines</li> <li>Retry a ServiceComb engine task</li> <li>Enable/Disable security authentication for Nacos engines</li> <li>Manage a Nacos user and role</li> <li>Manage relationship between namespaces and enterprise projects</li> </ul>

Permission	Description	Permission Dependency	Scenario
cse:engine:upgrade	Upgrade an engine	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> </ul>	<ul style="list-style-type: none"> <li>Upgrade a ServiceComb engine</li> <li>Upgrade a Nacos engine</li> </ul> <p>Upgrades include version upgrade and specification change.</p>
cse:engine:delete	Delete an engine	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> <li>vpc:ports:get</li> <li>vpc:ports:delete</li> </ul>	<ul style="list-style-type: none"> <li>Delete a ServiceComb engine</li> <li>Delete a Nacos engine</li> </ul>
cse:engine:create	Create an engine	<ul style="list-style-type: none"> <li>cse:engine:get</li> <li>cse:engine:list</li> <li>ecs:cloudServerFlavors:get</li> <li>vpc:vpcs:get</li> <li>vpc:vpcs:list</li> <li>vpc:subnets:get</li> <li>vpc:ports:get</li> <li>vpc:ports:create</li> </ul>	<ul style="list-style-type: none"> <li>Create a ServiceComb engine</li> <li>Back up a ServiceComb engine/Restore task creation</li> <li>Create a Nacos engine</li> </ul>
cse:config:modify	Modify ServiceComb engine configuration management	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> <li>cse:config:get</li> </ul>	Modify global and governance configurations of a ServiceComb engine
cse:config:get	View ServiceComb engine configuration management	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> </ul>	View service configurations for a ServiceComb engine
cse:dashboard:get	View the ServiceComb engine dashboard	<ul style="list-style-type: none"> <li>cse:engine:list</li> <li>cse:engine:get</li> </ul>	View service dashboard information of a ServiceComb engine

Permission	Description	Permission Dependency	Scenario
cse:dashboard:modify	Modify the ServiceComb engine dashboard	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> <li>● cse:dashboard:get</li> </ul>	Modify the dashboard configuration of a ServiceComb engine
cse:governance:modify	Modify the governance center of a ServiceComb engine	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> <li>● cse:config:get</li> <li>● cse:config:modify</li> <li>● cse:registry:get</li> <li>● cse:registry:modify</li> <li>● cse:governance:get</li> </ul>	Create and modify service governance of a ServiceComb engine
cse:governance:get	View the governance center of a ServiceComb engine	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> <li>● cse:config:get</li> <li>● cse:registry:get</li> </ul>	View service governance on a ServiceComb engine
cse:registry:modify	Modify service registry and management of a ServiceComb engine	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> <li>● cse:registry:get</li> </ul>	Modify a ServiceComb engine
cse:registry:get	View service registry and management of a ServiceComb engine	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> </ul>	View service catalog on a ServiceComb engine
cse:namespace:read	View namespace resources of a Nacos engine	<ul style="list-style-type: none"> <li>● cse:engine:list</li> <li>● cse:engine:get</li> </ul>	View Nacos service list and configuration list
cse:namespace:write	Modify namespace resources of a Nacos engine	<ul style="list-style-type: none"> <li>● cse:engine:get</li> <li>● cse:namespace:read</li> </ul>	Modify Nacos service list and configuration list resources

## Helpful Links

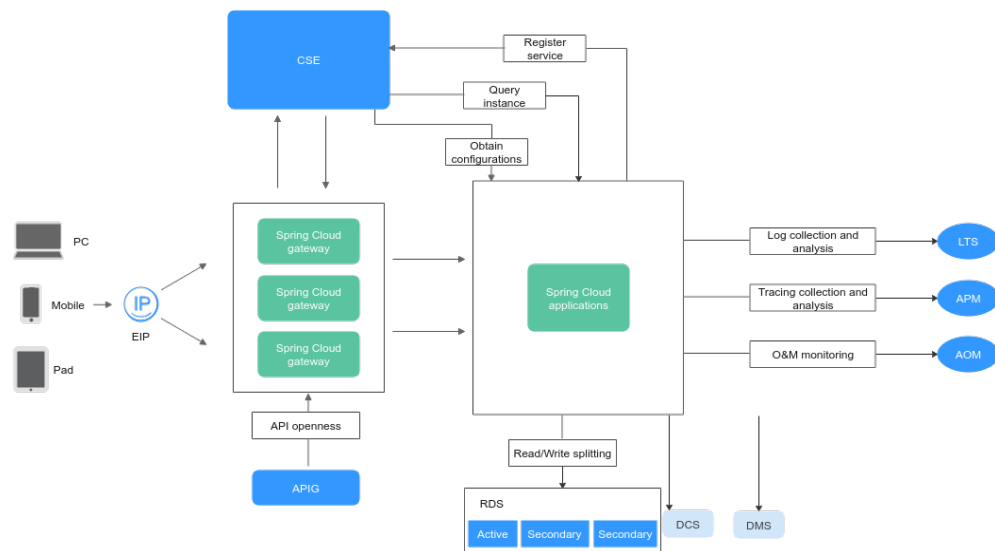
- "Service Overview" in the *Identity and Access Management User Guide*
- [Creating a User and Granting CSE Permissions](#)

## 1.9 Relationships Between CSE and Other Services

In the cloud-native architecture, many services need to cooperate with each other to implement service functions.

- Generally, CSE is used together with the database, cache, and message middleware to develop service functions.
- Tools such as AOM, APM, and LTS provide O&M capabilities for services, helping detect service faults and analyze fault causes.

The following takes Spring Cloud as an example. The typical cloud-native architecture and technology selection are as follows:



The cloud-native architecture and DevOps are inseparable. ServiceStage can be used together to implement cloud-native environment management and pipeline deployment, simplifying the process of deploying microservice applications to CCE.

# 2 Getting Started

---

## 2.1 Connecting Spring Cloud Applications to ServiceComb Engines

This section uses a demo to demonstrate how to connect a Spring Cloud application to a ServiceComb engine Using Spring Cloud Huawei SDK.

1. **Preparations**

You need to grant permissions, create a VPC and subnet, obtain the demo package, and prepare the local host for compilation, building, and packaging.

2. **Creating a ServiceComb Engine**

You can select engine specifications, AZs, and networks when creating an engine.

3. **Connecting Spring Cloud Applications to ServiceComb Engines**

This section describes how to connect a provider service and a consumer service to a ServiceComb engine.

### Preparations

1. You have obtained an account and its password for logging in to the console.
2. Grant permissions.  
You must have the required permissions to create dependent resources and ServiceComb engines. For details, see [Creating a User and Granting CSE Permissions](#).
3. Create a VPC and subnet.  
ServiceComb engines run in a VPC and need to be bound to a subnet. You must have a VPC and subnet to create ServiceComb engines. For details, see "Creating a VPC with a Subnet" in *Virtual Private Cloud User Guide*. Skip this step if you already have a VPC and subnet.
4. The Java JDK and Maven have been installed on the local host for compilation, building, and packaging, and the Maven central library can be accessed.

- Download the [demo source code](#) from GitHub to the local host and decompress it.


## Creating a ServiceComb Engine

**Step 1** Log in to CSE.

**Step 2** Choose **Exclusive ServiceComb Engines > Buy ServiceComb Engine**.

**Step 3** Set parameters according to the following table. Parameters marked with an asterisk (\*) are mandatory. For details, see [Creating a ServiceComb Engine](#).

**Table 2-1** Creating a ServiceComb engine

Parameter	Description
*Billing Mode	Billing mode. Currently, <b>Pay-per-use</b> is supported. Pay-per-use is postpaid. You use ServiceComb engines and then pay as billed for your usage duration.
*Enterprise Project	Select the enterprise project where the ServiceComb engine locates. Enterprise projects let you manage cloud resources and users by project. <b>default</b> is selected by default.
*Instances	Select the engine specifications. In this example, select 100 microservice instances.
*Engine Type	If the engine type is cluster, the engine is deployed in cluster mode and supports host-level DR.
*Name	Enter a ServiceComb engine name. The name contains 3 to 24 characters, including letters, digits, and hyphens (-), and starts with a letter but cannot end with a hyphen (-). For example, <b>cse-test</b> .
*AZ	Select one or three AZs. In this example, select one AZ to provide host-level DR.
*Network	Select the created VPC and subnet. You can search for and select a VPC and subnet from the drop-down list.
Description	Click  and enter the engine description, for example, create a ServiceComb engine.
*Security Authentication	The exclusive ServiceComb engine with security authentication enabled provides the system management function using the role-based access control (RBAC) through the microservice engine console. In this example, disable security authentication. You can enable it after the instance is created.

**Step 4** Click **Buy**. The page for confirming the engine information is displayed.

**Step 5** Click **Submit** and wait until the engine is created.

 NOTE

- It takes about 31 minutes to create a microservice engine.
- After the microservice engine is created, its status is **Available**. For details about how to view the microservice engine status, see [Viewing ServiceComb Engine Information](#).
- If the microservice engine fails to be created, view the failure cause on the **Operation** page and rectify the fault. Then, you can perform the following operations:
  - In the **Microservice Engine Information** area, click **Retry** to create a microservice engine again.
  - If the retry fails, delete the microservice engine that fails to be created. For details, see [Deleting a ServiceComb Engine](#).

----End

## Connecting Spring Cloud Applications to ServiceComb Engines

**Step 1** Log in to CSE.

**Step 2** Choose **Exclusive ServiceComb Engine**.

**Step 3** Click the ServiceComb engine created in [Creating a ServiceComb Engine](#).

**Step 4** Obtain the service center address and configuration center address of the ServiceComb engine.

In the **Service Discovery and Configuration** area, you can view the service center address and configuration center address of the microservice engine.

**Step 5** Change the addresses of the registry center and configuration center in the demo.

1. In the directory of the demo source code downloaded to the local host, find the `\basic\consumer\src\main\resources\bootstrap.yml` and `\basic\provider\src\main\resources\bootstrap.yml` files.
2. Add the service center address and configuration center address of the ServiceComb engine to the project configuration file (`\basic\consumer\src\main\resources\bootstrap.yml` is used as an example).

```
spring:
  application:
    name: basic-consumer
  cloud:
    servicecomb:
      discovery:
        enabled: true
        watch: false
        # Registry center address
        address: https://192.168.0.210:30100,https://192.168.0.246:30100
        appName: basic-application
        serviceName: ${spring.application.name}
        version: 0.0.1
        healthCheckInterval: 30
      config:
        # Configuration center address
        serverType: kie
        serverAddr: https://192.168.0.210:30110,https://192.168.0.246:30110
```

 NOTE

In the ServiceStage deployment scenario, the service registry center and configuration center addresses are automatically injected during the deployment. You do not need to manually add them.

**Step 6** Pack the demo source code into a JAR package.

1. In the root directory of the demo source code, open the Command Prompt and run the **mvn clean package** command to package and compile the project.
2. After the compilation is successful, two JAR packages are generated, as shown in [Table 2-2](#).

**Table 2-2** Software packages

Directory Where the Software Package Is Located	Package Name	Description
basic\consumer\target	basic-consumer-1.0-SNAPSHOT.jar	Service consumer
basic\provider\target	basic-provider-1.0-SNAPSHOT.jar	Service provider

**Step 7** Deploy the application.

Deploy provider and consumer on ServiceStage.

1. Upload the JAR package generated in [Step 6](#) to OBS.
2. Create a CCE cluster in the same VPC as the ServiceComb engine instance. For details, see "Creating a Kubernetes Cluster" in *Cloud Container Engine User Guide*.
3. Create a ServiceStage environment in the VPC where the engine instance is located, and manage the ServiceComb engine and CCE resource. For details, see "Creating an Environment" in *ServiceStage User Guide*.
4. Deploy provider and consumer. For details, see "Creating and Deploying a Component" in *ServiceStage User Guide*.

**Step 8** Confirm the deployment results.

1. **Optional:** On the CSE console, choose **Exclusive ServiceComb Engine** and click the ServiceComb engine created in [Creating a ServiceComb Engine](#).
2. Choose **Microservice Catalog > Microservice List** to view the number of **basic-consumer** and **basic-provider** microservice instances.
  - If **Instances** is not **0**, the demo has been connected to the ServiceComb engine.
  - If **Instances** is **0** or the **basic-consumer** and **basic-provider** services cannot be found, the demo fails to be connected to the microservice engine.

----End

## 2.2 Connecting Spring Cloud Applications to Nacos Engines

This section uses a demo to demonstrate how to connect microservice applications to Nacos engines.

### 1. Preparations

You need to grant permissions, create a VPC and subnet, obtain the demo package, and prepare the local host for compilation, building, and packaging.

### 2. Creating a Registry/Configuration Center

You can select engine specifications, AZs, and networks when creating an engine.

### 3. Connecting Spring Cloud Applications to Nacos Engines

This section describes how to connect a provider service and a consumer service to a Nacos engine.

## Preparations

1. You have obtained an account and its password for logging in to the console.
2. Grant permissions.

You must have the required permissions to create dependent resources and Nacos engines. For details, see [Creating a User and Granting CSE Permissions](#).

#### NOTE

To create a registry/configuration center, you must have the CSE FullAccess and DNS FullAccess permissions.

3. Create a VPC and subnet.  
Nacos engines run in a VPC and need to be bound to a subnet. You must have a VPC and subnet to create Nacos engines. For details, see "Creating a VPC with a Subnet" in *Virtual Private Cloud User Guide*. Skip this step if you already have a VPC and subnet.
4. The Java JDK and Maven have been installed on the local host for compilation, building, and packaging, and the Maven central library can be accessed.
5. Download the [demo source code](#) from GitHub to the local host and decompress it.

## Creating a Registry/Configuration Center

**Step 1** Log in to CSE.

**Step 2** In the left navigation pane, choose **Registry/Configuration Center** and click **Buy Registry/Configuration Center Instance**.

**Step 3** Set parameters according to the following table. Parameters marked with an asterisk (\*) are mandatory. For details, see [Creating a Nacos Engine](#).

**Table 2-3** Creating a registry/configuration center

Parameter	Description
*Billing Mode	Billing mode. Currently, <b>Pay-per-use</b> is supported. Pay-per-use is postpaid. You use registry/configuration centers and then pay as billed for your usage duration.
*Enterprise Project	Select the enterprise project where the Nacos engine is located. Enterprise projects let you manage cloud resources and users by project. <b>default</b> is selected by default.
*Name	Enter a Nacos engine name. The name contains 3 to 24 characters, including letters, digits, and hyphens (-), and starts with a letter but cannot end with a hyphen (-). For example, <b>nacos-test</b> .
*Registry/Configuration Center Instance	The registry/configuration center supports Nacos engines. <b>NOTE</b> Cluster nodes in the registry/configuration center are evenly distributed to different AZs. A failure of a single node does not affect external services. The registry/configuration center does not support AZ-level DR but provides host-level DR.
*Instances	Select the required capacity specifications. In this example, the number of instances is 500, and the number of capacity units is 10.
Version	Only the latest version can be created.
*Network	Select the created VPC and subnet. You can search for and select a VPC and subnet from the drop-down list. A VPC enables you to provision logically isolated, configurable, and manageable virtual networks for your engine.

**Step 4** Click **Buy Now**. The registry/configuration center starts to be created. When the status is **Available**, the registry/configuration center is created.

----End

## Connecting Spring Cloud Applications to Nacos Engines

**Step 1** Log in to CSE.

**Step 2** Obtain the registry and discovery address of a Nacos engine.

1. In the left navigation pane, choose **Registry/Configuration Center** and click the Nacos engine instance.
2. In the **Connection Information** area on the **Basic Information** page, obtain the service center address.

**Step 3** Change the configuration center address, service center address, and microservice name in the demo.

1. Configure the Nacos configuration center in **bootstrap.properties**.

Find the `nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-consumer-example\src\main\resources` file in the demo source code directory downloaded to the local host, add the `bootstrap.properties` file, and configure the Nacos configuration center.

```
spring.cloud.nacos.config.server-addr= XXX.nacos.cse.com:8848 //Address of the Nacos configuration center
spring.cloud.nacos.config.prefix= example //Prefix of the configuration file name
spring.cloud.nacos.config.file-extension= properties //Extension of the configuration file name
spring.cloud.nacos.config.group= XXX //Group to which the configuration file belongs. If this parameter is left blank, the default value DEFAULT_GROUP is used.
spring.cloud.nacos.config.namespace= XXX //ID of the namespace to which the configuration file belongs. If this parameter is left blank, the default value public is used.
```

2. Configure the Nacos service center address and microservice name in the `application.properties` file.

- Find the `nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-consumer-example\src\main\resources\application.properties` file in the demo source code directory downloaded to the local host and configure the consumer service.

```
server.port=8080
spring.application.name= service-consumer //Microservice name
spring.cloud.nacos.discovery.server-addr= XXX.nacos.cse.com:8848 //Nacos service center address
spring.cloud.nacos.discovery.group= XXX //Group to which the microservice belongs. If this parameter is left blank, the default value DEFAULT_GROUP is used.
spring.cloud.nacos.discovery.namespace= XXX //ID of the namespace to which the microservice belongs. If this parameter is left blank, the default value public is used.
spring.cloud.nacos.discovery.cluster-name= XXX //Name of the cluster to which the microservice belongs. If this parameter is left blank, the default value DEFAULT is used.
```

- Find the `nacos-examples-master\nacos-spring-cloud-example\nacos-spring-cloud-discovery-example\nacos-spring-cloud-provider-example\src\main\resources\application.properties` file in the demo source code directory downloaded to the local host and configure the provider service.

```
server.port=8070
spring.application.name= service-provider //Microservice name
spring.cloud.nacos.discovery.server-addr= XXX.nacos.cse.com:8848 //Nacos service center address
spring.cloud.nacos.discovery.group= XXX //Group to which the microservice belongs. If this parameter is left blank, the default value DEFAULT_GROUP is used.
spring.cloud.nacos.discovery.namespace= XXX //ID of the namespace to which the microservice belongs. If this parameter is left blank, the default value public is used.
spring.cloud.nacos.discovery.cluster-name= XXX //Name of the cluster to which the microservice belongs. If this parameter is left blank, the default value DEFAULT is used.
```

**Step 4** Pack the demo source code into a JAR package.

1. In the root directory of the demo source code, open the Command Prompt and run the `mvn clean package` command to package and compile the project.
2. After the compilation is successful, two JAR packages are generated, as shown in [Table 2-4](#).

**Table 2-4** Software packages

Directory Where the Software Package Is Located	Package Name	Description
\nacos-spring-cloud-consumer-example\target	nacos-spring-cloud-consumer-example-0.2.0-SNAPSHOT.jar	Service consumer
\nacos-spring-cloud-provider-example\target	nacos-spring-cloud-provider-example-0.2.0-SNAPSHOT.jar	Service provider

**Step 5** Deploy Spring Cloud applications.

Deploy provider and consumer on the ECS node in the VPC where the Nacos engine is located.

1. Create an ECS node in the VPC where the engine instance is located and log in to the ECS node.
2. Install JRE to provide a running environment for services.
3. Upload the JAR package generated in [Step 4](#) to the ECS node.
4. Run the `java -jar {JAR package}` command to run the generated JAR package.

**Step 6** Confirm the deployment results.

1. **Optional:** On the CSE console, choose **Registry/Configuration Center** and click the Nacos engine created in [Creating a Registry/Configuration Center](#).
2. Choose **Service Management** and check the number of instances of microservices **service-consumer** and **service-provider**.
  - If **Instances** is not **0**, the demo has been connected to the Nacos engine.
  - If **Instances** is **0** or the **service-consumer** and **service-provider** services cannot be found, the demo fails to be connected to the Nacos engine.

----End

# 3 Before You Start

---

To use CSE, ensure that the following conditions are met:

1. You have registered a cloud account.
2. The current login account has been authorized to use CSE. For details, see [Creating a User and Granting CSE Permissions](#).

## Restrictions

CSE needs the Virtual Private Cloud (VPC) and Domain Name Service (DNS) services to run. If you have not granted any permissions, create a cloud service agency to grant permissions to CSE. That is, click **OK** in the **Grant Permissions to CSE** dialog box. CSE will create an agency named `cse_admin_trust` on IAM. Go to the agency list to view the details. You must have the Security Administrator role. Without permissions, some CSE functions will be affected, including engine creation and upgrade and security authentication enabling/disabling.

**Figure 3-1** Creating an agency

### Grant Permissions to CSE ×

You have not granted any permissions. CSE needs the Virtual Private Cloud (VPC) and Domain Name Service (DNS) services to run, so create a cloud service agency to grant permissions to CSE.

- **Cloud Trace Service (CTS)**  
Pushes operation logs to CTS.
- **Domain Name Service (DNS)**  
Creates a service access domain name.
- **Virtual Private Cloud (VPC)**  
Creates a component access mode.
- **Elastic Load Balance (ELB)**  
Creates load balancers.
- **Enterprise Project Management Service (EPS)**  
Classifies enterprise project resources.
- **Elastic Volume Service (EVS)**  
Obtains EVS disk types.
- **Log Tank Service (LTS)**  
Reports service logs.

Once authorized, CSE will create an `cse_admin_trust` agency on Identity and Access Management (IAM). Go to the agency list to view the details. To grant permissions, you must have the Security Administrator role permissions. Confirm the permissions in the IAM service.

Without permissions, some CSE functions will be affected, including engine creation and upgrade and security authentication enabling/disabling.



# 4 Creating a User and Granting CSE Permissions

---

This section describes how to use IAM to implement fine-grained permissions control for your CSE resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials for access to CSE resources.
- Grant only the permissions required for users to perform a task.
- Entrust an account or cloud service to perform professional and efficient O&M on your CSE resources.

If your account does not require individual IAM users, skip this section.

This section describes the procedure for granting permissions (see [Figure 4-1](#)).

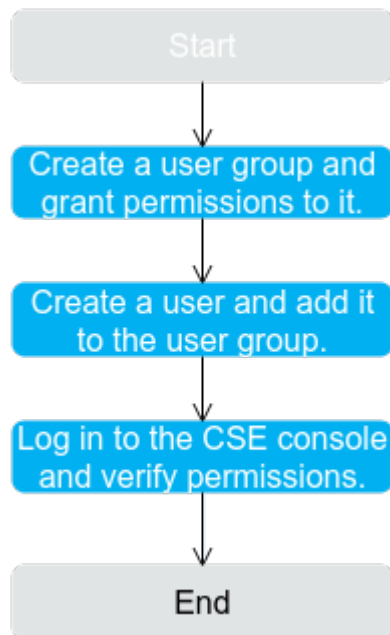
## Prerequisites

Before assigning permissions to user groups, you should learn about CSE policies and select the policies based on service requirements.

- For details about system permissions supported by CSE, see [Permissions](#).
- To grant permissions for other services, learn about all permissions supported by IAM by referring to System-defined Permissions.

## Process Flow

**Figure 4-1** Process of granting CSE permissions



1. Create a user group and grant permissions to it.  
Create a user group on the IAM console, and grant the **CSE ReadOnlyAccess** policy to the group.
2. Create a user and add it to the user group.  
Create a user on the IAM console and add the user to the group created in **1**.
3. Log in to CSE and verify permissions.  
Log in to the CSE console as the created user, and verify that it has the read-only permission for CSE.
  - In **Service List**, choose **Cloud Service Engine**. On the console, choose **ServiceComb > Buy Exclusive Microservice Engine**. If a message is displayed indicating insufficient permissions, the **ReadOnlyAccess** policy has taken effect.
  - Choose any other service in **Service List**. If a message is displayed indicating insufficient permissions to access the service, the **ReadOnlyAccess** policy has taken effect.

## CSE Custom Policies

Custom policies can be created as a supplement to the system-defined policies of CSE.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a JSON policy or edit an existing one.

For details, see "Creating a Custom Policy" in *Identity and Access Management User Guide*. The following section contains examples of common CSE custom policies.

#### Example Custom Policies

This procedure creates a policy that an IAM user is prohibited to create and delete a microservice engine.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "cse:*:*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "cse:engine:create",
        "cse:engine:delete"
      ],
      "Effect": "Deny"
    }
  ]
}
```

A policy with only "Deny" permissions must be used together with other policies. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

You can verify your granted permissions using the console or REST APIs.

The following uses the custom policy as an example to describe how to verify that a user is not allowed to create microservice engines on the console.

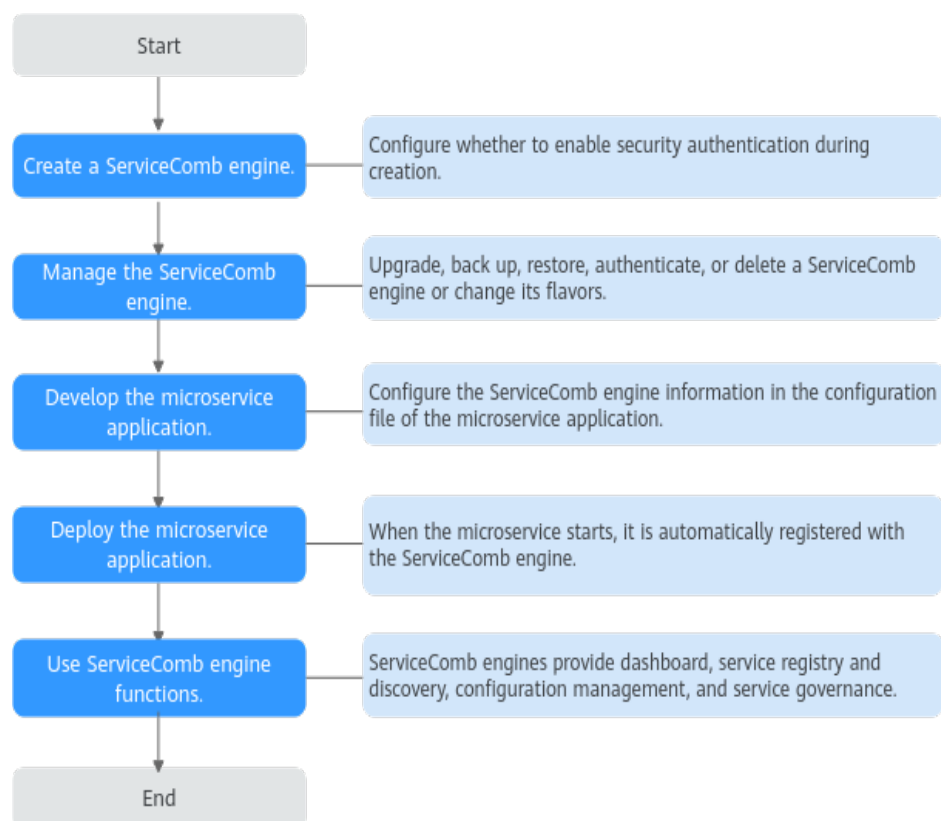
1. Log in to as an **IAM user**.
  - **Tenant name**: Name of the account used to create the IAM user
  - **IAM username** and **IAM user password**: Username and password specified during IAM user creation using the **Tenant name**
2. Create a microservice engine on the CSE console. If error 403 is returned, the permissions are correct and are in effect.

# 5 ServiceComb Engines

## 5.1 ServiceComb Engine Overview

The ServiceComb engine provides service registry, service governance, and configuration management. It allows you to quickly develop microservice applications and implement high-availability O&M, and supports multiple languages, multiple runtime systems, and Spring Cloud and Apache ServiceComb Java Chassis (Java chassis) frameworks.

The process of using a ServiceComb engine is as follows:



1. [Create a ServiceComb engine.](#)
2. Upgrade, back up, restore, authenticate, or delete a ServiceComb engine or change its flavors by referring to [Managing ServiceComb Engines.](#)
3. Configure the ServiceComb engine information in the configuration file of the developed microservice application, including the configuration center, registry center, dashboard, and governance policy.
4. Deploy the microservice. When the microservice starts, it is automatically registered with the ServiceComb engine.
5. Manage microservices using the functions provided by the ServiceComb engine. For details, see [Viewing Microservice Running Metrics Through the Microservice Dashboard](#) to [System Management.](#)

## 5.2 Creating a ServiceComb Engine

This section describes how to create a ServiceComb engine.

### Restrictions

- When a ServiceComb engine is in use, do not disable the enterprise project. Otherwise, the engine will not be displayed in the engine list, affecting normal use.
- The VPC cannot be changed once the engine is created.
- By default, a maximum of five ServiceComb engines can be created for each project. ServiceComb engines share quotas (five engines) with Nacos engines.

### Prerequisites

- The login account has the permission to create a microservice engine. For details, see [Creating a User and Granting CSE Permissions.](#)
- A ServiceComb engine runs on a VPC. Before creating a ServiceComb engine, ensure that VPCs and subnets are available. For details, see "Creating a VPC with a Subnet" in *Virtual Private Cloud User Guide.*
- If the engine is created using an account with the minimum permission for creating engines, for example, **cse:engine:create** in the [CSE Permissions](#), the default VPC security group **cse-engine-default-sg** needs to be preset by the primary account and the rules listed in [Table 5-1](#) need to be added. For details, see "Adding a Security Group Rule" in *Virtual Private Cloud User Guide.*

**Table 5-1** cse-engine-default-sg rules

Direction	Priority	Policy	Protocol and Port	Type	Source Address
Inbound	1	Allow	ICMPv6: All	IPv6	::/0
	1	Allow	TCP: 30100–30130	IPv6	::/0

Direction	Priority	Policy	Protocol and Port	Type	Source Address
	1	Allow	All	IPv6	cse-engine-default-sg
	1	Allow	All	IPv4	cse-engine-default-sg
	1	Allow	TCP: 30100–30130	IPv4	0.0.0.0/0
	1	Allow	ICMP: all	IPv4	0.0.0.0/0
Outbound	100	Allow	All	IPv4	0.0.0.0/0
	100	Allow	All	IPv6	::/0





## Creating a ServiceComb Engine

**Step 1** Log in to CSE.

**Step 2** Choose **Exclusive ServiceComb Engines > Create Exclusive ServiceComb Engine**.

**Step 3** Set parameters according to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Enterprise Project	<p>Select the project where the ServiceComb engine is located. You can search for and select the required enterprise project from the drop-down list.</p> <p>Enterprise projects let you manage cloud resources and users by project.</p> <p>An enterprise project can be used after it is created and enabled. For details, see "Creating an Enterprise Project" in <i>Enterprise Management User Guide</i>. By default, <b>default</b> is selected.</p> <p>After a ServiceComb engine is created, you can remove ServiceComb engine resources out of the current enterprise project and into a new enterprise project. For details, see "Removing Resources from an Enterprise Project" in <i>Enterprise Management User Guide</i> and "Adding Resources to an Enterprise Project" in <i>Enterprise Management User Guide</i>.</p>
*Instances	<p>Select the microservice instance quota. You can select the number of instances based on the number of microservice instances to be hosted. ServiceComb engine instances with different microservice instances are rewarded with corresponding configuration items and the maximum number of microservice versions supported.</p>
*Engine Type	<p>Select a ServiceComb engine type.</p> <p>If the engine type is cluster, the engine is deployed in cluster mode and supports host-level DR.</p>

Parameter	Description
*Name	Enter a ServiceComb engine name. The name contains 3 to 24 characters, including letters, digits, and hyphens (-), and starts with a letter but cannot end with a hyphen (-). The name cannot be <b>default</b> .
*AZ	<p>Availability zone.</p> <p>Select one or three AZs for the engine based on the number of AZs in the environment.</p> <ul style="list-style-type: none"> <li>• Select one AZ to provide host-level DR.</li> <li>• Select three AZs to provide AZ-level DR.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• The AZs in one region can communicate with each other over an intranet.</li> <li>• Multiple AZs enhance DR capabilities.</li> </ul>
*Network	<p>Select a VPC and subnet to provision logically isolated, configurable, and manageable virtual networks for your engine.</p> <ul style="list-style-type: none"> <li>• To use a created VPC, search for and select a VPC created under the current account from the drop-down list.</li> <li>• To use a new VPC, click <b>Create VPC</b> in the drop-down list. For details, see "Creating a VPC with a Subnet" in <i>Virtual Private Cloud User Guide</i>.</li> </ul>
Description	Click  and enter the engine description. The description can contain 0 to 255 characters.
Tag	<p>Tags are used to identify cloud resources. When you have multiple cloud resources of the same type, you can use tags to classify them based on usage, owner, or environment.</p> <p>Click  <b>Add Tag</b>. In the <b>Add Tag</b> dialog box, enter a tag key and value. For details about tag naming rules, see <a href="#">Managing Tags</a>. In the <b>Add Tag</b> dialog box, you can click  <b>Add Tag</b> to add multiple tags at a time, or click  next to a tag to delete the tag.</p>

Parameter	Description
Authentication Mode	<p>The exclusive ServiceComb engine with security authentication enabled provides the system management function using the role-based access control (RBAC) through the microservice engine console.</p> <ul style="list-style-type: none"> <li>• Select <b>Enable security authentication</b>:               <ol style="list-style-type: none"> <li>1. Determine whether to enable <b>Authenticate Programming Interface</b>. After it is enabled, you need to add the corresponding account and password to the microservice configuration file. Otherwise, the service cannot be registered with the engine.  After it is disabled, you can register the service with the engine without configuring the account and password in the microservice configuration file, which improves the efficiency. You are advised to disable this function when accessing the service in a VPC.</li> <li>2. Enter and confirm the password of user <b>root</b>. Keep the password secure.</li> </ol> </li> <li>• Select <b>Disable security authentication</b>: Disable security authentication. You can enable it after the instance is created.</li> </ul>

**Step 4** Click **Create**. The page for confirming the engine information is displayed.

**Step 5** Click **Submit**. When the engine status changes to **Available**, the creation is successful.

 **NOTE**

- It takes about 10–30 minutes to create a ServiceComb engine.
- After the ServiceComb engine is created, its status is **Available**. For details about how to check the engine status, see [Viewing ServiceComb Engine Information](#).
- If the ServiceComb engine fails to be created, view the failure cause on the **Operation** page and rectify the fault. Then, you can perform the following operations:
  - In the **ServiceComb Engine Information** area, click **Retry** to create an engine again.
  - If the retry fails, delete the ServiceComb engine that fails to be created. For details, see [Deleting a ServiceComb Engine](#).

----End

## 5.3 Managing ServiceComb Engines

### 5.3.1 Viewing ServiceComb Engine Information

You can click an engine to go to the engine details page and obtain the basic information, service registry and discovery address, configuration center address, instance quota, and configuration item quota.

## Viewing Basic ServiceComb Engine Information

In the **ServiceComb Engine Information** area, you can view the microservice engine information as shown in [Table 5-2](#).



**Step 1** Log in to CSE.

**Step 2** Choose **Exclusive ServiceComb Engines**.

**Step 3** Click the target engine.

**Step 4** In the **ServiceComb Engine Information** area, view the engine information shown in [Table 5-2](#).

**Table 5-2** Engine details

Item	Description
Name	Engine name entered when <a href="#">creating the ServiceComb engine</a> . Click  to copy it.
Engine ID	Engine ID. Click  to copy it.
Status	Engine status. <ul style="list-style-type: none"> <li>• Creating</li> <li>• Available</li> <li>• Unavailable</li> <li>• Configuring</li> <li>• Deleting</li> <li>• Upgrading</li> <li>• Resizing</li> <li>• Creation failed</li> <li>• Deletion failed</li> <li>• Upgrade failed</li> <li>• Resizing failed</li> <li>• Unknown</li> </ul>
Version	Engine version.
Engine Type	Engine specification selected when <a href="#">creating the ServiceComb engine</a> .
AZ	AZ selected when <a href="#">creating the ServiceComb engine</a> .
Tag	Tags added to the ServiceComb engine. You can also click <b>Tag Management</b> and perform operations on tags as required. For details, see <a href="#">Managing ServiceComb Engine Tags</a> .
Description	Engine description entered when <a href="#">creating the ServiceComb engine</a> .

----End

## Obtaining the Service Center Address of a ServiceComb Engine

The registry and discovery address is the core of service registry and discovery to implement dynamic service management. When a microservice is started, the registry center address is used to report its metadata (such as the IP address, port, service contract, and version) to the ServiceComb service center. This implements automatic service registry, avoids hard-coded addresses, and simplifies O&M. Service consumers use this address to query the list of available service instances from the registry center, select instances based on load balancing (such as weighted round robin), and dynamically detect service startup and shutdown (through heartbeats) to ensure that faulty instances are automatically removed, improving system fault tolerance.

The service center address cannot be changed after the engine is created.

**Step 1** Click the target engine.

**Step 2** In the **Service Discovery and Configuration** area, view the service center address of the microservice engine.

Service Discovery and Configuration		<a href="#">Microservice Catalog</a>   <a href="#">Configuration Management</a>
Connection Address of Service Center	<input type="checkbox"/> https://192.168.0.32:30100,https://192.168.0.103:30100	
Instances	<input type="text" value="0/100 (used/total) (0%)"/>	
Address of Config Center	<input type="checkbox"/> https://192.168.0.32:30110,https://192.168.0.103:30110	
Configuration Items	<input type="text" value="0/600 (used/total) (0%)"/>	

----End

## Obtaining the Configuration Center Address of a ServiceComb Engine

The configuration center manages microservice configurations. When a microservice is connected to the ServiceComb engine, you need to configure the configuration center address of the engine in the configuration file. ServiceComb establishes a persistent connection with the configuration center through this address. When detecting configuration changes, the configuration center pushes the changes to the microservice instance, and the configuration can be updated without restart, implementing dynamic configuration management.

**Step 1** Click the target engine.

**Step 2** In the **Service Discovery and Configuration** area, view the configuration center address of the microservice engine.

Service Discovery and Configuration		<a href="#">Microservice Catalog</a>   <a href="#">Configuration Management</a>
Connection Address of Service Center	<input type="checkbox"/> https://192.168.0.32:30100,https://192.168.0.103:30100	
Instances	<input type="text" value="0/100 (used/total) (0%)"/>	
Address of Config Center	<input type="checkbox"/> https://192.168.0.32:30110,https://192.168.0.103:30110	
Configuration Items	<input type="text" value="0/600 (used/total) (0%)"/>	

 NOTE

- For ServiceComb engine 1.x, the port number of the configuration center address is 30103.

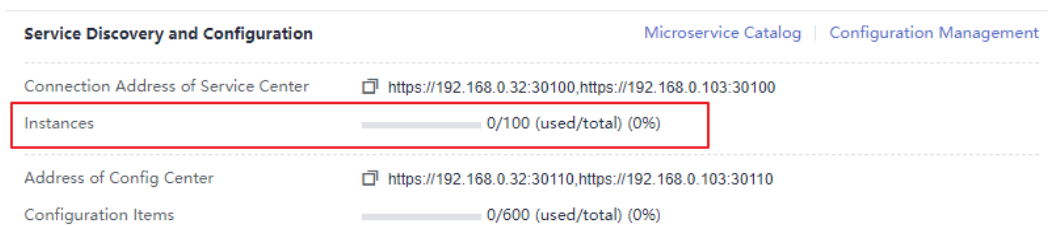
----End

## Viewing the Instance Quota of a ServiceComb Engine

This section describes how to view the instance quota and quota usage of a ServiceComb engine.

**Step 1** Click the target engine.

**Step 2** In the **Service Discovery and Configuration** area, view the instance quota and quota usage of the microservice engine.



----End

## Viewing the Configuration Item Quota of a ServiceComb Engine

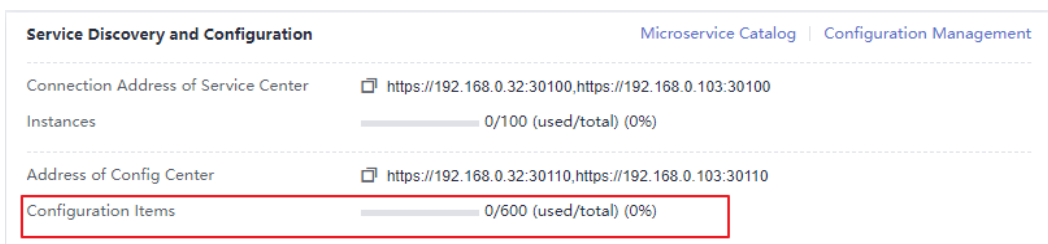
This section describes how to view the configuration item quota and quota usage of a ServiceComb engine.

 NOTE

This section applies only to ServiceComb engine 2.x.

**Step 1** Click the target engine.

**Step 2** In the **Service Discovery and Configuration** area, view the configuration item quota and quota usage of the microservice engine.



----End

## 5.3.2 Managing ServiceComb Engine Tags

Tags facilitate ServiceComb engine identification and management.

If your organization has configured tag policies for ServiceComb engines, add tags to engines based on the policies. If a tag does not comply with the tag policies,

engine creation may fail. Contact your administrator to learn more about tag policies.

You can add tags to a ServiceComb engine when creating the engine or add tags on the details page of the created engine. Up to 20 tags can be added to an engine. Tags can be modified and deleted.

A tag consists of a tag key and a tag value. [Table 5-3](#) lists the tag key and value requirements.

**Table 5-3** Tag naming rules

Tag	Rule
Key	<ul style="list-style-type: none"> <li>• Cannot be left blank.</li> <li>• Must be unique for the same instance.</li> <li>• Contain a maximum of 128 characters.</li> <li>• Contain only letters, digits, spaces, and special characters ( _ . : = + - @ ).</li> <li>• Cannot start with a space or <b>_sys_</b> or end with a space.</li> </ul>
Value	<ul style="list-style-type: none"> <li>• Contain a maximum of 255 characters.</li> <li>• Contain only letters, digits, spaces, and special characters ( _ . : = + - @ ).</li> </ul>


## Managing Tags

Adding or modifying tags will affect engine services for about 10 seconds. Add or modify tags during off-peak hours.

**Step 1** Click the target engine. The details page is displayed.

**Step 2** In the **ServiceComb Engine Information** area, perform the following operations in the **Tag** field as required:

- Add a tag
  - Click **Tag Management**. The **Edit Tag** dialog box is displayed.
  - Click **⊕ Add Tag** and enter a tag key and value in the text boxes.
  - Click **OK**.
- Modify a tag
  - Click **Tag Management**. The **Edit Tag** dialog box is displayed.
  - You can modify the tag key and value in the original text boxes.
  - Click **OK**.

- Delete a tag  
Click  in the row that contains the tag to be deleted. In the dialog box that is displayed, click **OK** to delete the tag.

----End

### 5.3.3 Managing Public Network Access for a ServiceComb Engine

The ServiceComb engine supports public network access, which helps expand the microservice architecture from a closed environment on the intranet to an open ecosystem on the public network. This not only meets the requirements of Internet services and cross-cloud collaboration, but also improves the system elasticity, scalability, and DR capabilities through the openness of the technical architecture. In addition, based on the security and governance mechanisms, public network access is balanced between openness and controllability, providing key technical support for the business model of internal and external linkage in enterprise digital transformation.

#### Restrictions

- ServiceComb engines with security authentication disabled do not have the authentication and authorization capabilities. Opening those engines to the public network may cause security risks and increases the system vulnerability. For example, data assets such as configurations and service information may be stolen.
- Do not use this function in a production environment or a network environment with high security requirements.

#### Prerequisites


An EIP has been created. For details, see "Assigning an EIP" in *Elastic IP User Guide*.

#### Binding an EIP

ServiceComb engines that are bound with EIPs can be accessed from the public network.

**Step 1** Click the target engine.

**Step 2** In the **Network Configuration and Security** area, click **Bind EIP**.

Network Configuration and Security				Enable security authentication	Bind EIP
Virtual Priv...	vpc-172-test	Subnet	subnet-c3a2(172.16.0.0/24)		
Public Net...		Security Au...	Disabled		

**Step 3** Read the security risk prompt in the displayed dialog box and select **I understand the security risks**.

**Step 4** In the **EIP** drop-down list, select the EIP to be bound. You can only select an EIP in the same enterprise project as the ServiceComb engine.

**Step 5** Click **OK**.

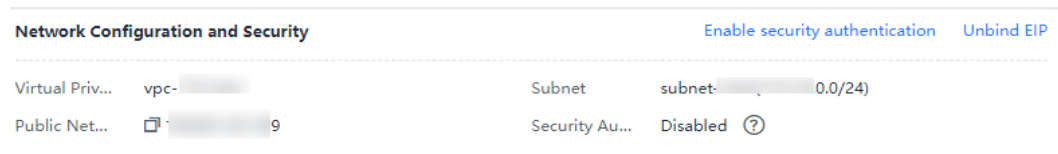
----End

## Unbinding an EIP

If an EIP has been bound to a ServiceComb engine, you can unbind the EIP from the engine to disable the public network access to the engine.

**Step 1** Click the target engine.

**Step 2** In the **Network Configuration and Security** area, click **Unbind EIP**.



**Step 3** In the displayed dialog box, click **OK**.

----End

## 5.3.4 Managing Security Authentication for a ServiceComb Engine

A ServiceComb engine may be used by multiple users. Different users must have different engine access and operation permissions based on their responsibilities and permissions. If security authentication is enabled for an exclusive ServiceComb engine, grant different access and operation permissions to users based on the roles associated with the accounts used by the users to access the engine.

For details about security authentication, see [System Management](#).

You can enable or disable security authentication for the exclusive ServiceComb engine based on service requirements.

- **Enabling Security Authentication**

If a ServiceComb engine is available with security authentication disabled, you can enable security authentication based on service requirements.

After security authentication is enabled and programming interface authentication is also enabled, if security authentication parameters are not configured for the microservice components connected to the engine, or the security authentication account and password configured for the microservice components are incorrect, the heartbeat of the microservice components fails and the service is forced to go offline.

- **Disabling Security Authentication**

If a ServiceComb engine is available with security authentication enabled, you can disable security authentication based on service requirements.

After security authentication is disabled for a microservice component, service functions of the microservice component are not affected no matter whether security authentication parameters are configured for the microservice component.

## Restrictions

Currently, Java chassis and Spring Cloud support security authentication for microservices. The Java chassis version must be 2.3.5 or later, and Spring Cloud must integrate Spring Cloud Huawei 1.6.1 or later.

## Enabling Security Authentication

**Step 1** Click the target engine.

**Step 2** In the **Network Configuration and Security** area, click **Enable security authentication**.

- If the engine version is earlier than 1.2.0, go to [Step 3](#).
- If the engine version is 1.2.0 or later, go to [Step 4](#).

**Step 3** Upgrade the engine to 1.2.0 or later.

1. Click **Upgrade**.
2. Select **Target Version** and view the version description. Determine whether to upgrade the software to this version. Then, click **OK**.
3. Select the upgraded ServiceComb engine. In the **Network Configuration and Security** area, click **Enable security authentication**.

**Step 4** On the **System Management** page, enable security authentication.

- To enable security authentication for the first time, click **Enable security authentication**.  
You need to create user **root** first. Enter and confirm the password of user **root**. Then, click **Create Now**.
- Enable security authentication again and enter the name and password of the account associated with the **admin** role in the engine.

**Step 5** (Optional) Create a role based on service requirements. For details, see [Roles](#).

**Step 6** (Optional) Create an account based on service requirements. For details, see [Accounts](#).

**Step 7** On the **System Management** page, click **Enable security authentication** and configure the security settings.

- If you enable **Authenticate Console**, go to [Step 9](#).  
To log in to CSE after **Authenticate Console** is enabled, determine whether to use an account and password based on the permissions of the login user. The login user can only view and configure services on which the user has permission.
- If you enable **Authenticate Programming Interface**, go to [Step 8](#).  
After **Authenticate Programming Interface** is enabled, **Authenticate Console** is automatically enabled.  
After it is enabled, you need to add the corresponding account and password to the microservice configuration file. Otherwise, the service cannot be registered with the engine.  
After it is disabled, you can register the service with the engine without configuring the account and password in the microservice configuration file,

which improves the efficiency. You are advised to disable this function when accessing the service in a VPC.

**Step 8** Configure the SDK. For microservice components that have been deployed but not configured with security authentication parameters, configure the account name and password for security authentication and then upgrade the component.

**Step 9** Click **OK**.

After the ServiceComb engine is updated and the engine status changes from **Configuring** to **Available**, security authentication is enabled successfully.

----End

## Disabling Security Authentication

After security authentication is disabled, accounts created on the engine will not be deleted.

**Step 1** Click the target engine.

**Step 2** In the **Network Configuration and Security** area, click **Disable security authentication**.

**Step 3** Click **OK**. After the ServiceComb engine is updated and the engine status changes from **Configuring** to **Available**, security authentication is disabled successfully.

----End

## 5.3.5 Configuring Backup and Restoration of a ServiceComb Engine

CSE provides the backup and restoration functions. You can back up and restore ServiceComb engine data, including microservices, contracts, configurations, and account role.

You can customize backup policies to periodically back up ServiceComb engines or manually back up ServiceComb engines.

### Restrictions

- Each ServiceComb engine supports a maximum of 15 successful backups, including a maximum of 10 manual backups and a maximum of 5 automatic backups.
- The backup data will be stored for 10 days. Expired backup data will be deleted.
- The backup data will overwrite the current data of the ServiceComb engine. As a result, the microservice and service instances may be messed, and dynamic configurations may be lost. Exercise caution when performing this operation.
- If security authentication is enabled, the backup data contains the account information. You are advised to disable security authentication before restoring the backup data. Otherwise, the authentication for accessing the ServiceComb engine may fail after the restoration.

## Automatic Backup

**Step 1** Click the target engine.

**Step 2** In the **Backup and Restoration** area, click **Automatic backup settings** and set backup parameters.

**Table 5-4** Automatic backup parameters

Parameter	Description
Automatic Backup	After automatic backup is disabled, the previously set backup policy will be deleted. In this case, you need to set the backup policy again.
Backup Interval	Backup period. Interval at which the system automatically backs up data. You can select one or more days from Monday to Sunday as the backup execution date. The selected dates will trigger automatic backup. This parameter takes effect after <b>Automatic Backup</b> is enabled.
Trigger Time	Time when a backup task starts. The backup task is triggered at the specified time on all the dates of the backup interval. Only the hour is supported. This parameter takes effect after <b>Automatic Backup</b> is enabled.

**Step 3** Click **OK**.

Once the backup policy is set, the backup task is triggered within one hour after the preset time.

----End

## Manual Backup

**Step 1** Click the engine whose data you want to manually back up.

**Step 2** In the **Backup and Restoration** area, click **Create Manual Backup** and set backup parameters.

**Table 5-5** Manual backup parameters

Parameter	Description
Name	Enter a backup task name. The name contains 3 to 24 characters, including letters, digits, and hyphens (-), and starts with a letter but cannot end with a hyphen (-). The name cannot be changed once the backup task is created.
Remarks	(Optional) Enter a description. The description can contain 0 to 255 characters.

**Step 3** Click **OK** to execute the backup task immediately. A backup task is generated in the backup task list. If the execution result changes from **Processing** to **Successful**, the manual backup is successful.

----End

## Restoring Backup Data

The backup data will overwrite the current data of the ServiceComb engine. As a result, the microservice and service instances may be messed, and dynamic configurations may be lost. Exercise caution when performing this operation.

**Step 1** Click the target engine.

**Step 2** In the **Backup and Restoration** area, click **Restore** in the **Operation** column of the row that contains the specified backup data.

1. Select **I have read and fully understand the risks**.
2. Click **OK**. To view the restoration status, click **Restoration History** in the **Backup and Restoration** area.
3. After successful restoration, choose **Configuration Management** to check whether the configurations are restored

----End

## Deleting Backup Data

**Step 1** Click the engine whose backup data you want to delete.

**Step 2** In the **Backup and Restoration** area, click **Delete** in the **Operation** column of the target backup data. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

## 5.3.6 Upgrading a ServiceComb Engine

ServiceComb engines are created using the latest engine version. When a later version is released, you can upgrade your engine.

During upgrade, two instances are upgraded in rolling mode without service interruptions. However, one of the two access addresses may be unavailable. In this case, you need to quickly switch to the other instance. Currently, ServiceComb SDK and Mesher support instance switching. If you call the APIs of the service center and configuration center for service registry and discovery, instance switching is required.

### Restrictions

- During the ServiceComb engine upgrade, the microservice and engine are intermittently disconnected, but services of running microservices are not affected. You are advised not to upgrade, restart, or change microservices when upgrading a ServiceComb engine.
- Version rollback is not supported after the upgrade.

## Upgrading a ServiceComb Engine

- Step 1** Click **Upgrade** in the **Operation** column of the target engine. Alternatively, click the target engine and click **Upgrade** in the **ServiceComb Engine Information** area.
- Step 2** Select **Target Version** and view the version description. Determine whether to upgrade the software to this version.
- Step 3** Click **OK**. If the engine status changes from **Upgrading** to **Available**, the upgrade is complete.

If the upgrade fails, click **Retry** to perform the upgrade again.

----End

### 5.3.7 Changing ServiceComb Engine Specifications

Specifications of exclusive ServiceComb engines can be automatically changed online. Currently, only scale-out is supported. The service will be disconnected intermittently during the change, which does not affect services. New services cannot be registered during the change.

#### Restrictions

The service will be disconnected intermittently during the change, which does not affect services. New services cannot be registered during the change.

### Changing ServiceComb Engine Specifications

- Step 1** Click **More > Change Specifications** in the **Operation** column of the target engine. Alternatively, click the target engine and click **Change Specifications** in the **ServiceComb Engine Information** area.
- Step 2** On the displayed page, select the target specifications.
- Step 3** Click **Change Now**, confirm the information, and click **Submit**. If the engine status changes from **Resizing** to **Available**, the change is complete.

----End


### 5.3.8 Viewing ServiceComb Engine Operation Logs

In the **Operation** area, view the operation logs of a ServiceComb engine. The operation logs record user operations on the engine, which are useful for security compliance and audit.

#### Viewing ServiceComb Engine Operation Logs

- Step 1** Click the target engine.
- Step 2** In the **Operation** area, view the operation logs of a ServiceComb engine.

Operation							2022-03-15 - 2022-03-23
No.	Operation Type	Status	Username	Start Time	End Time	Details	
1	Create	Successful		2022/03/22 16:23:18 GMT+08:00	2022/03/22 16:32:24 GMT+08:00	Create a new engine. More	

- Click  in the upper right corner to view operation logs in a specified period.
- Click **More** in the **Details** column of a specified operation log to view details about the operation log.

----End

### 5.3.9 Deleting a ServiceComb Engine

You can delete a ServiceComb engine if it is no longer used. You can delete ServiceComb engines in the following states:

- Available
- Unavailable
- Creation failed
- Resizing failed
- Upgrade failed
- Unknown

#### Restrictions

- Deleted engines cannot be restored. Exercise caution when performing this operation.
- For engine 1.x, if the `cse_admin_trust` agency is missing, deleting the engine will cause residual DNS, VPC, and security group resources on the tenant side. You need to delete them by yourself.

#### Deleting a ServiceComb Engine

**Step 1** Click **Delete** in the **Operation** column of the target engine. Alternatively, click the target engine and click **Delete** in the **ServiceComb Engine Information** area.

**Step 2** In the displayed dialog box, enter **DELETE** and click **OK**.

#### NOTE

If the deletion fails, click **Force Delete**.

----End

## 5.4 Viewing Microservice Running Metrics Through the Microservice Dashboard

You can view metrics related to microservices through the dashboard in real time. Based on abundant and real-time dashboard data, you can take corresponding governance actions for microservices.

#### Restrictions

- This function is supported by ServiceComb engine 1.x and 2.4.0 and later versions.

- If a microservice application is deployed on ServiceStage, you need to configure the microservice engine during application deployment. The application automatically obtains the service registry and discovery address, configuration center address, and dashboard address. You do not need to configure the monitor address.
- If the microservice application is locally started and registered with the ServiceComb engine, manually configure the monitor address before using the dashboard.
- When the Spring Cloud Huawei framework is used for access, the median latency, 90th latency, and 99th latency cannot be viewed on the dashboard.

## Viewing Microservice Running Metrics

**Step 1** Click the target engine.

**Step 2** Choose **Dashboard**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

### NOTE

- If you connect to the ServiceComb engine for the first time, enter the account name **root** and the password entered when [Creating a ServiceComb Engine](#).
- For details about how to create an account, see [Adding an Account](#).

**Step 4** On the **Dashboard** page, select the target application. Enter a microservice name in the search box to search for the microservice. The running metrics of the microservice are displayed.

Click **View Diagram** to view the description of operating metrics.

**Step 5** Select a sorting order to sort the filtered microservices.

----End

## 5.5 Managing Microservices

### 5.5.1 Viewing an Application

The **Application List** tab displays all applications of the current ServiceComb engine. You can search for the target application by application name, or filter applications by environment.

#### Viewing the Application List

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Application List** to view details about all applications of the current account under the engine.

You can search for the target application by application name, or filter applications by environment.

----End

## 5.5.2 Microservice Management

On the **Microservice List** tab, you can create, view, delete, dynamically configure, and dark launch microservices, and clean microservices without instances.

### Creating a Microservice

Create a microservice for testing or restoring the microservice that is deleted by mistake. If a microservice that is automatically registered with ServiceComb is deleted by mistake, you can manually create a microservice to restore the deleted microservice. The microservice name, application, version, and environment must be the same as those of the original microservice.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Choose **Microservice List > Create a Microservice** and set microservice parameters by referring to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Microservice	Microservice name, for example, <b>myServiceName</b> . Enter 1 to 128 characters. Start and end with a digit or letter. Only use digits, letters, and special characters (_-.). The special characters cannot be used consecutively.

Parameter	Description
*Application	Name of the application to which the microservice belongs. Microservices are isolated by applications. Enter 1 to 160 characters. Start and end with a digit or letter. Only use digits, letters, and special characters (_-). The special characters cannot be used consecutively.
*Version	Microservice version. The default value is <b>1.0.0</b> . The microservice version is in the format of X.Y.Z or X.Y.Z.B, where X, Y, Z, and B are digits and range from 0 to 32767. The value contains 3 to 46 characters.
*Environment	Environment where the microservice is located to isolate microservice data, including the version and instance.
Description	Microservice description.

**Step 5** Click **OK**.

Once the microservice is created, it will be displayed in **Microservice List**.

----End

## Viewing the Microservice List

The microservice list displays all microservices of the current ServiceComb engine. You can search for the target microservice by microservice name, or filter microservices by environment and application.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Microservice List** to view all microservices of the current account under the engine.

You can search for the target microservice by microservice name, or filter microservices by environment and application.

----End

## Viewing Microservice Details

On the microservice details page, you can view the instance list, called services, calling services, dynamic configuration, and service contract. You can also perform [Dynamic Configuration](#) and [Dark Launch](#) on microservice-level configurations.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the microservice to be viewed in **Microservice List**. On the displayed page, view the instance list, called services, calling services, dynamic configuration, dark launch, and service contract.

----End

## Cleaning Versions Without Instances

Delete microservice versions with no running instances to optimize resource utilization, improve governance efficiency, and ensure system stability.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Choose **Microservice List > Clean No Instance Services**. Select the microservice version without instances to be cleaned.

You can search for the target microservice by microservice name, or filter microservices by environment and application.

**Step 5** Click **OK**.

----End

## Dynamic Configuration

Create, edit, disable, and delete microservice-level configurations, and view historical versions of microservice-level configurations.

Configuration items are stored in plaintext. Do not include sensitive data.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Microservice List** and click a microservice.

**Step 5** Choose **Dynamic Configuration**. On the **Dynamic Configuration** tab, perform the following operations.

Operation	Procedure
Create a configuration item	See <a href="#">Creating a Configuration for ServiceComb Engine 2.x</a> . <b>Microservice-level</b> is selected for <b>Configuration Range</b> and <b>Microservices</b> is set to the current microservice.
View historical versions	Click <b>View Historical Version</b> in the <b>Operation</b> column of the target configuration item.
Disable a configuration item	<ol style="list-style-type: none"> <li>In the <b>Operation</b> column of the target configuration item, choose <b>More &gt; Disable</b>.</li> <li>Click <b>OK</b>.</li> </ol>
Modify a configuration item	<ol style="list-style-type: none"> <li>Click <b>Edit</b> in the <b>Operation</b> column corresponding to the target configuration item.</li> <li>On the configuration details page, click <b>Edit</b>.</li> <li>On the <b>Configuration Details</b> tab, enter the new configuration.</li> <li>Click <b>Save</b>.</li> </ol>
Delete a configuration item	<ol style="list-style-type: none"> <li>In the <b>Operation</b> column of the target configuration item, choose <b>More &gt; Delete</b>.</li> <li>Click <b>OK</b>.</li> </ol>

----End

## Dark Launch

In dark launch, new features are tested in a selected group of users. When the features become mature and stable, they are released to all users. This ensures the smooth feature rollout.

- For microservices developed based on the ServiceComb Java Chassis framework, add dependency **darklaunch** or **handler-router** to POM and add **servicecomb.router.type=router** to the **microservice.yaml** configuration file.
- For microservices developed based on the Spring Cloud Huawei framework, add dependency **spring-cloud-starter-huawei-router** to POM.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).

- For engines with security authentication enabled, if the login VDC user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** In the microservice list, click a microservice. On the displayed page, choose **Dark Launch**.

**Step 5** Click **Add Launch Rule**.

- To add a launch rule by **Weight**:
  - a. Click **Weight**.
  - b. Set the following parameters.

Parameter	Description
Rule Name	Name of the rule. Enter 3 to 24 characters. Start with a letter. Only use digits, letters, and special characters (.@_-).
Scope	<ul style="list-style-type: none"> <li>▪ Microservice version to which the rule applies.</li> <li>▪ Select <b>Do you want to add a customized version?</b> and add a new version as prompted.</li> </ul>
Rule Configuration	Traffic allocation rate for the selected version. Traffic is evenly allocated to the selected service versions based on the configured value.

- c. Click **OK** to complete the weight rule configuration and dark launch.
- To add a launch rule by **Customization**:
    - a. Click **Custom**.
    - b. Set the following parameters.

Parameter	Description
Rule Name	Name of the rule. Enter 3 to 24 characters. Start with a letter. Only use digits, letters, and special characters (.@_-).
Scope	<ul style="list-style-type: none"> <li>▪ Microservice version to which the rule applies.</li> <li>▪ Select <b>Do you want to add a customized version?</b> and add a new version as prompted.</li> </ul>

Parameter	Description
Rule Configuration	<p>Configure the matching rule. When <b>darklaunch</b> is used to implement dark launch, this parameter configures <b>policyCondition</b>.</p> <p>When <b>handler-router</b> is used to implement dark launch, this parameter configures <b>Headers</b>.</p> <ul style="list-style-type: none"> <li>▪ <b>Parameter Name</b> Set this parameter based on the parameter name of contract or the customized key of the header.</li> <li>▪ <b>Rules</b> By selecting the matching character and the value corresponding to the key of contract or the key of the header, requests that meet the rules are allocated to the microservice version.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>○ If ~ is selected from the drop-down list next to <b>Rules</b>, the asterisk (*) and question mark (?) in the <b>Rules</b> value can be used for fuzzy matching. The asterisk (*) indicates a character of any length, and the question mark (?) indicates one character. For example, if the rule value of <b>Name</b> is set to <b>*1000</b>, all <b>Name</b> fields ending with 1000 can be matched.</li> <li>○ If ~ is not selected from the drop-down list next to <b>Rules</b>, the asterisk (*) and question mark (?) in the <b>Rules</b> value cannot be used for fuzzy matching.</li> </ul>

- c. Click **OK** to complete the customization rule configuration and dark launch.

----End

Examples of delivering dark launch rules:

- For microservices developed based on the ServiceComb Java Chassis framework, rules are delivered based on dependency **darklaunch** on the ServiceComb engine page. You can add dark launch rules in customized mode.

**Create a Rule** ×

★ Launch Rule Weight Customization

★ Rule Name

**Scope**

★ Version  1.0.0

Do you want to add a customized version?

**Rule Configuration**

★ Parameter Name

★ Rules

\* Requests meeting the [name=11111] condition will be allocated to the 1.0.0 microservice version.

OK Cancel

This key must exist in the contract. It is possible that the server API is **String paramA**, but **paramB** is actually generated after the annotation is added. Therefore, **paramB** should be set here.

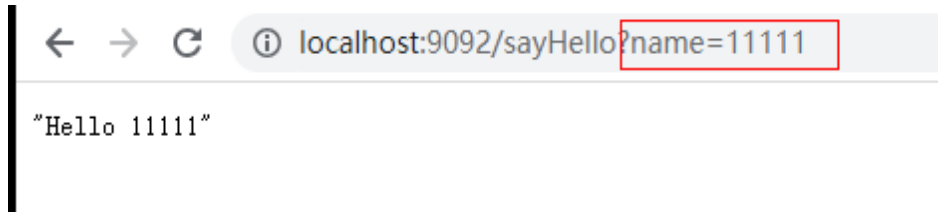
forecast

Swagger Yaml

⚠ Exercise caution when inputting sensitive information in Service Contract items and values, or encrypt sensitive information to avoid information leakage. For example, user privacy and database password.

```
11 application/json
12 produces:
13 - "application/json"
14 paths:
15 /sayHello:
16 get:
17   operationId: "sayHello"
18   produces:
19     - "application/json"
20   parameters:
21     - name: "name"
22       in: "query"
23       required: true
24       type: "string"
25   responses:
26     "200":
27       description: "response of 200"
28       schema:
29         type: "string"
```

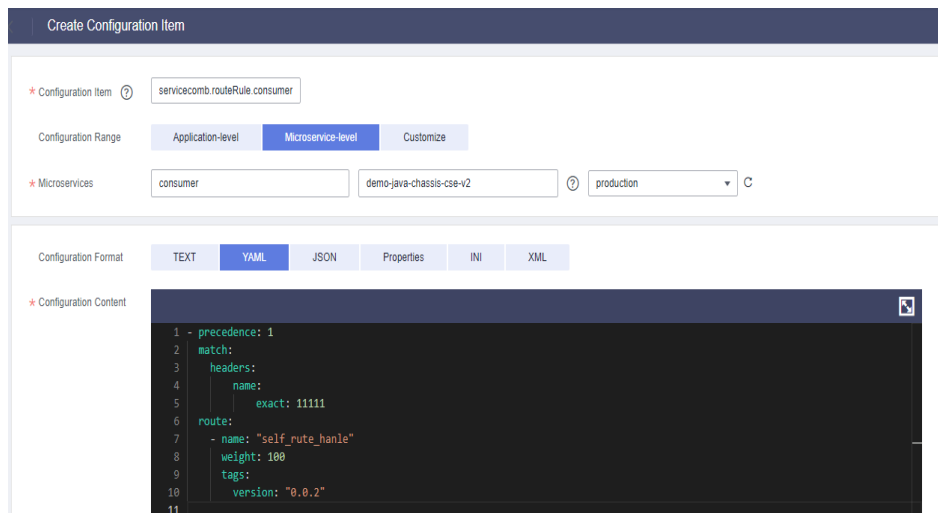
By selecting the matching character and the value corresponding to the key of contract, requests that meet the rules are allocated to the microservice version.



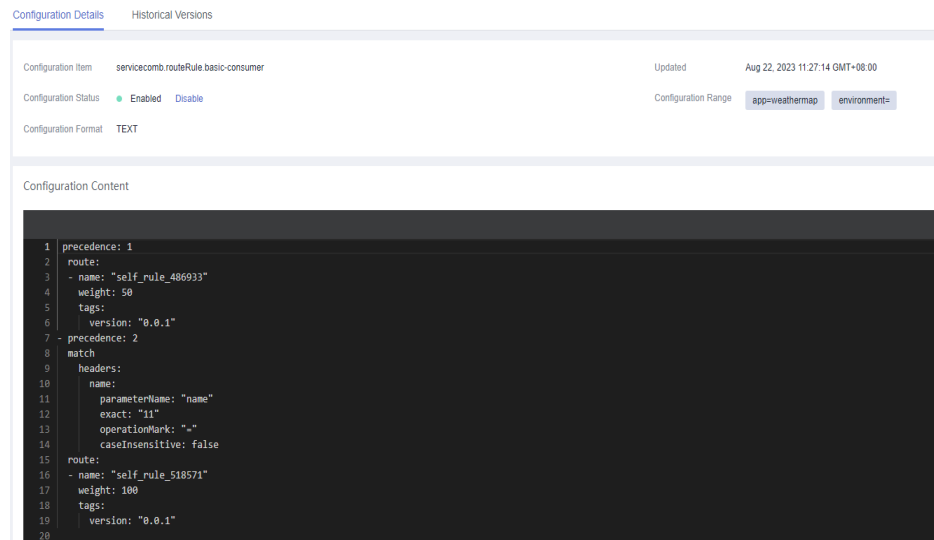
A delivered rule is as follows. The configuration item is `cse.darklaunch.policy.${serviceName}`.



- For microservices developed based on the ServiceComb Java Chassis framework, dark launch rules that depend on **handler-router** need to be manually delivered in the configuration center. The configuration item is `servicecomb.routeRule.${serviceName}`. The content is as follows:



- For microservices developed based on the Spring Cloud Huawei framework, dark launch rules delivered on the ServiceComb engine page are as follows:



## Deleting a Microservice

Delete a microservice that is no longer used.

- After a microservice is deleted, you can restore it by referring to [Restoring Backup Data](#).
- If the service to be deleted has instances, delete the instances first. Otherwise, the service will be registered again.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Microservice List**.

- To delete microservices in batches, select the microservices to be deleted and click **Delete** above the microservices.
- To delete one microservice, locate the row that contains the microservice to be deleted and click **Delete** in the **Operation** column.

**Step 5** In the displayed dialog box, enter **DELETE** to confirm the deletion and click **OK**.

----End

### 5.5.3 Instance Management

CSE allows you to view and change the status of all microservice instances registered with the ServiceComb engine.

## Viewing the Instance List

The instance list displays all instances of the current ServiceComb engine. You can search for the target instance by microservice name, or filter instances by environment and application.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Instance List** to view all instances of the engine.

You can search for the target instance by microservice name, or filter instances by environment and application.

----End

## Changing the Instance Status

**Status** indicates the status of a microservice instance.

The status of microservice instances synchronized by binding ServiceComb engines cannot be changed during component creation and deployment by referring to "Creating and Deploying a Component" in *ServiceStage User Guide*.

The following table describes the microservice instance statuses.

Statu s	Description
Onlin e	The instance is running and can provide services.
Offlin e	Before the instance process ends, the instance is marked as not providing services externally.
Out of Servic e	The instance has been registered with the ServiceComb engine and does not provide services.
Testin g	The instance is in the internal joint commissioning state and does not provide services.

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Catalog**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Instance List**, select the target instance, and change the instance status.

- Offline  
In the **Operation** column, click **Offline**.
- Online  
In the **Operation** column, choose **More > Online**.
- Out of Service  
In the **Operation** column, choose **More > Out of Service**.
- Testing  
In the **Operation** column, choose **More > Testing**.

----End

## 5.6 Service Scenario Governance (Applicable to ServiceComb Engine 2.x)

### 5.6.1 Service Scenario Governance Overview

ServiceComb engines provide unified traffic feature governance based on dynamic configurations for different microservice development frameworks, such as Spring Cloud and Java chassis. You can use the microservice governance function of CSE by introducing related governance components to the development frameworks.

ServiceComb engine governance consists of two steps: creating a service scenario and creating a governance policy. The two steps can be performed before microservice deployment for independent governance planning.

#### Restrictions

- Service scenario governance is applicable to ServiceComb engine 2.x.
- If you want to delete a governance policy in use, pay attention to the following information:
  - Risks: The governance policy may become invalid, which reduces the capability of the microservice system to resist abnormal traffic. When there is abnormal traffic, problems such as unbalanced call distribution and microservice avalanche may occur.
  - Precautions: Perform the operations during off-peak hours. Before the operations, ensure that there is no abnormal traffic, such as many calls and call timeout.

## Governance Policies

You can configure the following policies: rate limiting, circuit breaker, retry, and bulkhead. For details, see the following table.

Policy	Description
Rate limiting	In the case of a traffic storm or predictable traffic spikes, rate limiting is performed on non-key service scenarios to prevent service and data breakdown caused by instantaneous heavy traffic.
Retry	When a service encounters a non-fatal error (such as occasional timeout), retries can be performed to prevent service failures.
Bulkhead	In the case of a large-scale concurrent traffic storm or predictable traffic impact, the concurrent traffic is controlled to prevent service and data breakdown caused by instantaneously large concurrent traffic.
Circuit breaker	When the error rate of a service scenario exceeds the threshold, all requests in the service scenario will be rejected within one minute to ensure the availability of the entire service system. Then 50% of the service requests will be accepted and statistics on the service error rate will be collected until the error rate in the service scenario is reduced to a value lower than the threshold.

### 5.6.2 Creating a Service Scenario

You can create service scenarios based on your service process and requirements, and implement governance policies accordingly to ensure stable and efficient service running.

#### Prerequisites

- You have deployed an application by referring to "Creating and Deploying a Component" in *ServiceStage User Guide*.
- You need to understand the API design of the microservice to be governed and create service scenarios based on the API features.

#### Creating a Service Scenario

**Step 1** Click the target engine.

**Step 2** Choose **Service Scenario Governance**.

##### NOTE

If the ServiceComb engine version is 2.0.0 or later but earlier than 2.4.0, choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).


- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Choose **Service Scenarios > Create Service Scenario** and set parameters by referring to the following table.

Parameter	Description
Scenario	Enter a service scenario name. Enter up to 20 characters.
Environment	Select a microservice environment.
Application	Select the application to which the service scenario to be created belongs.
Matching Rule	<p>You can set <b>Method</b>, <b>Path</b>, and <b>Headers</b> rules to filter requests that meet specific conditions. Only requests that meet these rules are included in the created service scenario, which facilitates centralized governance of specific types of requests.</p> <p>Click <b>Add Matching Rule</b> to set the request marker.</p> <ul style="list-style-type: none"> <li>• <b>Method:</b> (Mandatory) Select the method of marking the traffic request feature. The <b>GET</b>, <b>PUT</b>, <b>POST</b>, <b>DELETE</b>, and <b>PATCH</b> methods are supported.</li> <li>• <b>Path:</b> (Mandatory) Set features contained in the traffic request URI.</li> <li>• <b>Headers:</b> (Optional) Click <b>Add Headers Rule</b> and set the marker of the traffic request header.</li> </ul>

**Step 5** Click **OK**. When a service scenario is created, the configuration starting with **servicecomb.matchgroup.** is automatically generated.

- Click  in the row that contains a service scenario to view the matching rule details.
- Click **Edit** in the **Operation** column of a service scenario to edit the service scenario.
- Click **Delete** in the **Operation** column of a service scenario to delete the service scenario.

----End

### 5.6.3 Creating a Governance Policy

A governance policy is a method used for microservice governance. Each governance policy can be bound to a service scenario. A policy cannot be bound to multiple service scenarios. Different governance policies can be bound to the same service scenario.

## Prerequisites

- You have created a service scenario by referring to [Creating a Service Scenario](#).
- You need to enable the dynamic configuration-based traffic feature governance function for the development framework of the microservice to be governed. If the function is not enabled, the microservice governance function can still be used, but the governance has no effects.

## Creating a Governance Policy

**Step 1** Click the target engine.

**Step 2** Choose **Service Scenario Governance**.

 **NOTE**

If the ServiceComb engine version is 2.0.0 or later but earlier than 2.4.0, choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Go to the **Governance Policy** page and click **Create Governance Policy**.

**Step 5** Select a governance mode, click **Create Policy**, and set parameters.

- Rate limiting

Parameter	Description
Policy	Enter a governance policy name. Enter up to 32 characters.
Service Scenarios	Set the service scenarios to which the governance policy applies. <ul style="list-style-type: none"> <li>– Click <b>Select Service Scenario</b> and select the created service scenario.</li> <li>– Click <b>Create Service Scenario</b>. For details, see <a href="#">Creating a Service Scenario</a>.</li> </ul>
Requests per Unit	Set the number of requests and time segment. When the number of requests sent by the rate limiting object to the current service instance within the specified period exceeds the specified value, the excess requests are limited and error code 429 is returned.

- Retry

Parameter	Description
Policy	Enter a governance policy name. Enter up to 32 characters.
Service Scenarios	Set the service scenarios to which the governance policy applies. <ul style="list-style-type: none"> <li>– Click <b>Select Service Scenario</b> and select the created service scenario.</li> <li>– Click <b>Create Service Scenario</b>. For details, see <a href="#">Creating a Service Scenario</a>.</li> </ul>
Response Error Code	Enter response error codes to define the error types that trigger retries.
Retry Attempts	Set the number of retries.
Retry Interval	Select a retry policy. <ul style="list-style-type: none"> <li>– <b>Fixed</b>: The retry interval is fixed. For example, if the retry interval is set to 500 ms, the interval between two consecutive retries remains 500 ms regardless of the number of retries.</li> <li>– <b>Exponential</b>: Each retry interval is randomly determined based on an algorithm. Generally, the retry interval increases exponentially with the number of retries. For example, the retry interval may be 100 ms for the first retry, 200 ms for the second retry, and 400 ms for the third retry.</li> </ul>
Interval Duration	Set the retry interval duration. <ul style="list-style-type: none"> <li>– If <b>Retry Interval</b> is set to <b>Fixed</b>, set the fixed retry interval. The value is an integer ranging from 1 to 60,000, in seconds or milliseconds.</li> <li>– If <b>Retry Interval</b> is set to <b>Exponential</b>, set the retry benchmark time. The value ranges from 1 to 60,000, in seconds or milliseconds.</li> </ul>

- Bulkhead

Parameter	Description
Policy	Enter a governance policy name. Enter up to 32 characters.
Service Scenarios	Set the service scenarios to which the governance policy applies. <ul style="list-style-type: none"> <li>– Click <b>Select Service Scenario</b> and select the created service scenario.</li> <li>– Click <b>Create Service Scenario</b>. For details, see <a href="#">Creating a Service Scenario</a>.</li> </ul>


Parameter	Description
Max. Concurrency	Set the maximum number of concurrent requests based on the actual service processing capability of the system.
Block Duration	When the number of concurrent requests exceeds the maximum, the requests are discarded after the blocking duration. The value is an integer ranging from 1 to 300,000, in seconds or milliseconds.

- Circuit breaker

Parameter	Description	
Policy	Enter a governance policy name. Enter up to 32 characters.	
Service Scenarios	Set the service scenarios to which the governance policy applies. <ul style="list-style-type: none"> <li>- Click <b>Select Service Scenario</b> and select the created service scenario.</li> <li>- Click <b>Create Service Scenario</b>. For details, see <a href="#">Creating a Service Scenario</a>.</li> </ul>	
Coverage	Sliding Window Type	Select a sliding window type. Sliding window: range of request call times. The system monitors calls within this range to trigger the circuit breaker when they exceed the baseline. <ul style="list-style-type: none"> <li>- <b>Time</b>: The window range is determined by time.</li> <li>- <b>Requests</b>: The window range is determined by the number of requests.</li> </ul>
	Sliding Window Size	Set the size of the sliding window. <ul style="list-style-type: none"> <li>- If <b>Sliding Window Type</b> is set to <b>Time</b>, the calls in the last n seconds or minutes are recorded and collected.</li> <li>- If <b>Sliding Window Type</b> is set to <b>Requests</b>, the latest n calls are recorded and collected.</li> </ul> n is the size of the sliding window.
	Calls Baseline	Set the baseline of the number of calls, that is, the minimum number of calls required for collecting statistics on the call error rate.  For example, if <b>Calls Baseline</b> is set to <b>10</b> , at least 10 call must be recorded to collect statistics on the error rate.

Parameter		Description
Triggers	Error Threshold	Percentage of call errors. This parameter is valid when <b>Set circuit breaker to trigger at an error threshold</b> is selected.  When the call error rate is greater than or equal to the error threshold, circuit breaker occurs and response code 429 is returned.
	Slow Request Ratio	This parameter is valid when <b>Set circuit breaker to trigger at a specific request speed and ratio</b> is selected. Set the following parameters: <ul style="list-style-type: none"> <li>- <b>Slow Speed:</b> defines the slow request threshold. If the response time of a request exceeds the threshold, the request is a slow request.</li> <li>- <b>Slow Threshold:</b> When the specified slow request ratio is reached, circuit breaker occurs and response code 429 is returned.</li> </ul>

**Step 6** Click **Create** to make the governance policy take effect.

In the governance policy list, click  in the row where the service scenario is located:

- Click **Enable** in the **Operation** column of a governance policy to enable the policy.
- Click **Disable** in the **Operation** column of a governance policy to disable the policy.
- Click **Edit** in the **Operation** column of a governance policy to edit the policy.
- Click **Delete** in the **Operation** column of a governance policy to delete the policy.

----End

## 5.7 Microservice Governance (Applicable to ServiceComb Engine 1.x and 2.4.0+)

### 5.7.1 Microservice Governance Overview

If an application is developed using the microservice framework, the microservice is automatically registered with the corresponding ServiceComb engine after the application is managed and started. You can perform service governance on the engine console.

After a microservice is deployed, you can govern the request traffic, troubleshooting, and load balancing of the microservice based on the microservice running status.

## Governance Policies

You can configure the following policies: load balancing, rate limiting, fault tolerance, service degradation, circuit breaker, fault injection, and black and white list. For details, see the following table.

Name	Description
Load balancing	<ul style="list-style-type: none"> <li>Application scenario Generally, multiple instances are deployed for a microservice. Load balancing controls the policy for a microservice consumer to access multiple instances of a microservice provider to balance traffic. It includes polling, random, response time weigh, and session stickiness.</li> </ul>
Rate limiting	<ul style="list-style-type: none"> <li>Application scenario This policy controls the number of requests for accessing microservices to prevent the system from being damaged due to traffic impact.</li> </ul>
Service degradation	<ul style="list-style-type: none"> <li>Application scenario When a microservice invokes other microservices, the default value is forcibly returned or an exception is thrown instead of sending the request to the target microservice. In this way, the access to the target microservice is shielded and the pressure on the target microservice is reduced.</li> </ul>
Fault tolerance	<ul style="list-style-type: none"> <li>Application scenario If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected, the request needs to be forwarded to another available instance. Fault tolerance is often referred to as retry.</li> </ul>
Circuit breaker	<ul style="list-style-type: none"> <li>Application scenario If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected or the request times out, and the exception accumulates to a certain extent, the consumer needs to stop accessing the provider and return an exception or a default value to prevent the avalanche effect.  Automatic circuit breaker is supported, which determines a circuit breaker according to the error rate.</li> </ul>
Fault injection	<p>This policy applies only to microservices accessed through Java chassis.</p> <ul style="list-style-type: none"> <li>Application scenario Fault injection can simulate an invoking failure, which is mainly used for function verification and fault scenario demonstration.</li> </ul>

Name	Description
Blacklist/Whitelist	<p>This policy applies only to microservices accessed through Java chassis.</p> <ul style="list-style-type: none"> <li>Application scenario Based on the public key authentication mechanism, the ServiceComb engine provides the blacklist and whitelist functions. The blacklist and whitelist can be used to control which services can be accessed by microservices.</li> <li>Governance of microservices accessed through Java chassis The blacklist and whitelist take effect only after public key authentication is enabled. For details, see <a href="#">Configuring Blacklist and Whitelist</a>.</li> </ul>

## 5.7.2 Configuring a Load Balancing Policy

Generally, multiple instances are deployed for a microservice. Load balancing controls the policy for a microservice consumer to access multiple instances of a microservice provider to balance traffic. It includes polling, random, response time weigh, and session stickiness.

### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

### Configuring Load Balancing

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Choose **Load Balancing**.

**Step 5** Click **New**. Select the microservices to be governed and select a proper load balancing policy. For details, see the following table.

Policy	Description
Round robin	Supports routes according to the location information about service instances.

Policy	Description
Random	Provides random routes for service instances.
Response time weigh	This configuration applies to microservices accessed through Java chassis. Provides weight routes with the minimum active number (latency) and supports service instances with slow service processing in receiving a small number of requests to prevent the system from stopping response. This load balancing policy is suitable for applications with low and stable service requests.
Session stickiness	This configuration applies to microservices accessed through Java chassis. Provides a mechanism on the load balancer. In the specified session stickiness duration, this mechanism allocates the access requests related to the same user to the same instance. <ul style="list-style-type: none"> <li>● <b>Stickiness Duration:</b> time limit for keeping a session. The value ranges from 0 to 86400, in seconds.</li> <li>● <b>Failures:</b> number of access failures. The value ranges from 0 to 10. If the upper limit of failures or the session stickiness duration exceeds the specified values, the microservice stops accessing this instance.</li> </ul>

**Step 6** Click **OK**.

----End

### 5.7.3 Configuring a Rate Limiting Policy

This policy controls the number of requests for accessing microservices to prevent the system from being damaged due to traffic impact.

#### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

#### Configuring Rate Limiting

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

- Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.
- Step 4** Click the target microservice. Click **Rate Limiting**.
- Step 5** Click **New**. The following table describes configuration items of rate limiting.

Parameter	Description
Rate Limiting Object	This configuration applies to microservices accessed through Java chassis. Scope for the rate limiting rule to take effect.
Upstream Microservice	This configuration applies to microservices accessed through Spring Cloud. In the microservice architecture, each microservice works with other microservices to implement service functions. Select another microservice that invokes the current microservice.
QPS	Requests generated per second. When the number of requests sent by the rate limiting object to the current service instance exceeds the specified value, the current service instance no longer accepts requests from the rate limiting object. The value ranges from 1 to 99999. If a microservice has three instances, the rate limiting of each instance is set to 2700 QPS, then the total QPS is 8100. In this case, rate limiting is triggered only when the QPS exceeds 8100.

- Step 6** Click **OK**.

----End

## 5.7.4 Configuring a Service Degradation Policy

When a microservice invokes other microservices, the default value is forcibly returned or an exception is thrown instead of sending the request to the target microservice. In this way, the access to the target microservice is shielded and the pressure on the target microservice is reduced.

### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

### Configuring Service Degradation

- Step 1** Click the target engine.


**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Click **Service Degradation**.

**Step 5** Click **New** and select a proper policy. The following table describes the configuration items of service degradation.

Parameter	Description
Fallback Object	Microservice to be degraded. When microservices are accessed through Java chassis, one or more methods in the selected microservice are used as the object.
Request Path	This configuration applies to microservices accessed through Spring Cloud.  You can click  and set <b>Method</b> (such as GET, POST, and PUT), <b>Path</b> (request path), and <b>Headers</b> to filter requests.
Fallback	<ul style="list-style-type: none"> <li>• <b>Open</b>: enables service degradation and sets parameters.</li> <li>• <b>Close</b>: disables service degradation.</li> </ul>

**Step 6** Click **OK**.

----End

## 5.7.5 Configuring a Fault Tolerance Policy

If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected, the request needs to be forwarded to another available instance. Fault tolerance is often referred to as retry.

### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

### Configuring Fault Tolerance

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Click **Fault Tolerance**.

**Step 5** Click **New** and select a proper policy. The following table describes the configuration items of fault tolerance.

Parameter	Description
Downstream Microservice	This configuration applies to microservices accessed through Spring Cloud. Configure fault tolerance for the microservice to invoke the downstream microservice. You can select a downstream microservice from the drop-down list.
Fault Tolerance Object	This configuration applies to microservices accessed through Java chassis. Microservice or method that the application relies on.
Fault Tolerance	<b>Open:</b> The system processes a request sent to the fault tolerance object based on the selected fault tolerance policy when the request encounters an error. <b>Close:</b> The system waits until the timeout interval expires and then returns the failure result even though the service request fails to be implemented.

Parameter	Description
FT Policy	<p>This parameter is mandatory when <b>Fault Tolerance</b> is set to <b>Open</b>.</p> <p>For microservices accessed through Spring Cloud, set the following parameters:</p> <ul style="list-style-type: none"> <li>• Number of attempts to the same microservice instance</li> <li>• Number of attempts to the new microservice instance</li> </ul> <p>For microservices accessed through Java chassis, set the following parameters:</p> <ul style="list-style-type: none"> <li>• Failover The system attempts to connect to different servers.</li> <li>• Failfast The system does not attempt to connect to a server. After a request fails, a failure result is returned immediately.</li> <li>• Failback The system attempts to connect to the same server.</li> <li>• custom <ul style="list-style-type: none"> <li>- <b>Request Attempts of One Server:</b> number of attempts to connect to the same server</li> <li>- <b>Request Attempts of a New Server:</b> number of attempts to connect to another server</li> </ul> </li> </ul>

**Step 6** Click **OK**.

----End

## 5.7.6 Configuring a Circuit Breaker Policy

If an exception occurs when a microservice consumer accesses a provider, for example, the instance network is disconnected or the request times out, and the exception accumulates to a certain extent, the consumer needs to stop accessing the provider and return an exception or a default value to prevent the avalanche effect.

### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

### Configuring Circuit Breaker

**Step 1** Click the target engine.


**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Click **Circuit Breaker**.

**Step 5** Click **New** and select a proper policy. The following table describes the configuration items of circuit breaker.

Parameter	Description
Downstream Microservice	This configuration applies to microservices accessed through Spring Cloud. Configure circuit breaker for the microservice to invoke the downstream microservice. You can select a downstream microservice from the drop-down list.
Circuit Breaker Object	This configuration applies to microservices accessed through Java chassis. Microservice or method called by the application.
Request Path	This configuration applies to microservices accessed through Spring Cloud.  You can click  and set <b>Method</b> (such as GET, POST, and PUT), <b>Path</b> (request path), and <b>Headers</b> to filter requests.
Circuit Breaker Time Window	Circuit breaker duration. No response is sent within the time window. Unit: ms.
Circuit Breaker Rate	Percentage of failed requests within the specified number of window requests. For example, if <b>Window Requests</b> is set to 20 and <b>Circuit Breaker Rate</b> is set to 50%, circuit breaker is triggered when 10 of the 20 (50%) requests fail.
Window Requests	Number of requests received within the time window. Circuit breaker is triggered only when <b>Circuit Breaker Rate</b> and <b>Window Requests</b> both reach their thresholds.

**Step 6** Click **OK**.

----End

## 5.7.7 Configuring a Fault Injection Policy

Fault injection can simulate an invoking failure, which is mainly used for function verification and fault scenario demonstration. This policy applies only to microservices accessed through Java chassis.

### Prerequisites

You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.

### Configuring Fault Injection

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Click **Fault Injection**.

**Step 5** Click **New** and select a proper policy. The following table describes the configuration items of fault injection.

Parameter	Description
Injection Object	Microservices for which fault injection is required. You can specify a method for this configuration item.
Type	Type of the fault injected to the microservice. <ul style="list-style-type: none"> <li>• Delayed</li> <li>• Fault</li> </ul>
Protocol	Protocol for accessing the microservice when latency or fault occurs. <ul style="list-style-type: none"> <li>• Rest</li> <li>• Highway</li> </ul>
Occurrence Probability	Probability of latency or fault occurrence.
Delay Time	Duration of the latency during microservice access. This parameter is required when <b>Type</b> is set to <b>Delayed</b> .

Parameter	Description
HTTP Error Code	HTTP error code during microservice access. This parameter is required when <b>Type</b> is set to <b>Fault</b> . This error code is an HTTP error code.

**Step 6** Click **OK**.

----End

## 5.7.8 Configuring Blacklist and Whitelist

Based on the public key authentication mechanism, the ServiceComb engine provides the blacklist and whitelist functions. The blacklist and whitelist can be used to control which services can be accessed by microservices. This policy applies only to microservices accessed through Java chassis.

### Prerequisites

- You have created a microservice by referring to [Creating a Microservice](#). After the microservice starts, the service instance is registered with the corresponding service based on the configurations in the YAML file. If the microservice is not created in advance or has been deleted, the microservice is automatically created when the service instance is registered.
- The blacklist and whitelist take effect only after public key authentication is enabled. For details, see [Configuring Public Key Authentication](#).

### Configuring Blacklist and Whitelist

**Step 1** Click the target engine.

**Step 2** Choose **Microservice Governance**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target microservice. Click **Black and white list**.

**Step 5** Click **New** to add a blacklist or whitelist for the application. The following table describes configuration items of blacklist and whitelist.

Parameter	Description
Type	<ul style="list-style-type: none"> <li><b>Blacklist:</b> Microservices that match the matching rule are not allowed to access the current service.</li> <li><b>Whitelist:</b> Microservices that match the matching rule are allowed to access the current service.</li> </ul>

Parameter	Description
Rule	Use a regular expression. For example, if <b>Rule</b> is set to <b>data*</b> , services whose names start with <b>data</b> in the blacklist are not allowed to access the current service, or services whose names start with <b>data</b> in the whitelist are allowed to access the current service.

**Step 6** Click **OK**.

----End

## 5.7.9 Configuring Public Key Authentication

Public key authentication is a simple and efficient authentication mechanism between microservices provided by CSE. Its security is based on the reliable interaction between microservices and the service center. That is, the authentication mechanism must be enabled between microservices and the service center. The procedure is as follows:

1. When a microservice starts, a key pair is generated and the public key is registered with the service center.
2. Before accessing the provider, the consumer uses its own private key to sign a message.
3. The provider obtains the public key of the consumer from the service center and verifies the signed message.

To enable public key authentication, perform the following steps:

1. Enable public key authentication for both the consumer and provider.

```
servicecomb:
  handler:
    chain:
      Consumer:
        default: auth-consumer
      Provider:
        default: auth-provider
```

2. Add the following dependency to the **pom.xml** file:

```
<dependency>
  <groupId>org.apache.servicecomb</groupId>
  <artifactId>handler-publickey-auth</artifactId>
</dependency>
```

## 5.8 Configuration Management (Applicable to ServiceComb Engine 2.x)

### 5.8.1 Creating a Configuration for ServiceComb Engine 2.x

ServiceComb engines define a configuration mechanism that is irrelevant to development frameworks. A configuration item consists of a key, label, and value. The label is used to identify whether a configuration item belongs to global

configuration or microservice configuration. The label can also indicate the value type.

## Restrictions

- Configuration items are stored in plaintext. Do not include sensitive data.
- When the configuration item quota specified by the engine specifications is about to be used up, the engine allows new configuration items that exceed the remaining quota to be created to ensure capacity availability. Expand the capacity of the engine as soon as possible to prevent configuration creation failures.
- When editing or deleting a configuration item used by a microservice, the microservice may fail to read the configuration or read an incorrect configuration, causing service exceptions. Therefore, back up the configuration before modifying or deleting configuration items.

## Creating an Application-Level Configuration

Associates the new configuration with an application, and adds the application name and environment label.

### NOTE

**Configuration items are stored in plaintext. Do not include sensitive data.**

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Create Configuration Item** and set parameters by referring to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Configuration Item	<p>Enter a configuration item.</p> <p>The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, <b>cse.service.registry.address</b>) to ensure the readability and uniqueness of the configuration.</p> <p>Configuration items starting with <b>servicecomb.matchGroup</b> cannot be created during application-level configuration creation. Such configuration items conflict with the configuration generated during service scenario governance creation, so the service scenario cannot be displayed.</p>

Parameter	Description
Configuration Scope	Select <b>Application-level</b> .
*Application	1. Select or enter an application name. 2. Select an environment.
Configuration Format	Select a configuration format. Common configuration formats such as TEXT, YAML, JSON, Properties, INI and XML can be edited online. The default value is <b>TEXT</b> .
*Configuration Content	Enter the configuration content.
Enable Configuration	Determine whether to enable the configuration item. <ul style="list-style-type: none"> <li>• <b>Enable now:</b> The configuration item takes effect immediately once being created.</li> <li>• <b>Not Enabled:</b> The configuration item does not take effect.</li> </ul>

**Step 5** Click **Create Now** to enable the configuration item.

----End

## Creating a Microservice-Level Configuration

Associates the new configuration with a microservice, and adds the microservice name, application name, and environment.

### NOTE

**Configuration items are stored in plaintext. Do not include sensitive data.**

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Create Configuration Item** and set parameters by referring to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Configuration Item	Enter a configuration item. The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, <b>cse.service.registry.address</b> ) to ensure the readability and uniqueness of the configuration.
Configuration Scope	Select <b>Microservice-level</b> .
*Microservice	<ol style="list-style-type: none"> <li>1. Select or enter a microservice name.</li> <li>2. Select or enter an application name.</li> <li>3. Select an environment.</li> </ol>
Configuration Format	Select a configuration format. Common configuration formats such as TEXT, YAML, JSON, Properties, INI and XML can be edited online. The default value is <b>TEXT</b> .
*Configuration Content	Enter the configuration content.
Enable Configuration	Determine whether to enable the configuration item. <ul style="list-style-type: none"> <li>• <b>Enable now:</b> The configuration item takes effect immediately once being created.</li> <li>• <b>Not Enabled:</b> The configuration item does not take effect.</li> </ul>

**Step 5** Click **Create Now** to enable the configuration item.

----End

## Creating a Customized Configuration Item

If application-level and microservice-level configurations cannot meet service requirements, you can customize configuration files.

### NOTE

**Configuration items are stored in plaintext. Do not include sensitive data.**

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Create Configuration Item** and set parameters by referring to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Configuration Item	Enter a configuration item. The configuration item is the global ID of the configuration. In the coding phase, the configuration item is used to index and operate the configuration. You are advised to use the Java package naming rule (for example, <b>cse.service.registry.address</b> ) to ensure the readability and uniqueness of the configuration.
Configuration Scope	Select <b>Customize</b> .
Tag	If application-level and microservice-level configurations cannot meet service requirements, you can use labels to customize configurations.
Configuration Format	Select a configuration format. Common configuration formats such as TEXT, YAML, JSON, Properties, INI and XML can be edited online. The default value is <b>TEXT</b> .
*Configuration Content	Enter the configuration content.
Enable Configuration	Determine whether to enable the configuration item. <ul style="list-style-type: none"> <li>• <b>Enable now:</b> The configuration item takes effect immediately once being created.</li> <li>• <b>Not Enabled:</b> The configuration item does not take effect.</li> </ul>

**Step 5** Click **Create Now** to enable the configuration item.

----End

## 5.8.2 Managing Configurations for ServiceComb Engine 2.x

### Editing a Configuration

You can edit the configuration information of a ServiceComb engine as required. The configuration can be dynamically updated without restarting the service.

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Edit** in the **Operation** column of the target configuration item. You can also click the target configuration item and then click **Edit** on the displayed configuration details page.

**Step 5** Enter the configuration information in the **Configuration Content** text box and click **Save**.

----End

## Importing Configurations

You can import a local JSON configuration file of a ServiceComb engine on the console to simplify the process of creating configurations manually. To reuse configurations among development, test, and pre-release environments, or to migrate a ServiceComb engine across clouds or platforms, you can export the configurations of the production environment first and then import the configuration file to the target engine to initialize the configurations in batches. This shortens the environment setup. If the configurations are deleted or tampered with, or the ServiceComb engine is faulty, you can import the backup configuration file to quickly rebuild the service configuration environment.

### NOTE

- After importing the configuration file, the handling of existing configurations will be processed based on the specified processing policy, such as terminating the import, skipping the conflicting configurations, or overwriting them.
- Before importing the configuration file, back up the existing target environment configurations. Import the file incrementally, and verify core service configurations in a small scope before proceeding with batch imports.

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Import** in the upper right corner to Import files in different formats as required, and set parameters by referring to the following table.

**Table 5-6** V2.0 file

Parameter	Description
File Format	Select a format of the file to be imported. The default format is <b>V2.0</b> .

Parameter	Description
Import to Specific Environment	<ul style="list-style-type: none"> <li>• Disabled: The imported configuration does not change the environment label.</li> <li>• Enabled: Importing the configuration to a specific environment will change the environment label. Select an environment from the drop-down list.</li> </ul>
Same Configuration	<ul style="list-style-type: none"> <li>• <b>Terminate</b>: If a configuration is the same as that in the system, the import terminates.</li> <li>• <b>Skip</b>: During import, if a configuration is the same as that in the system, the configuration is skipped and other configurations are imported.</li> <li>• <b>Overwrite</b>: During import, if a configuration is the same as that in the system, the value of the configuration will be replaced.</li> </ul>
Configuration File	<p>Click <b>Import</b> and select the target file.</p> <p>Only JSON files can be imported and the file size cannot exceed 2 MB.</p>

**Table 5-7** V1.0 file

Parameter	Description
File Format	Select <b>V1.0</b> .
*Import to Specific Environment	Select a microservice environment.
Microservice	Select the microservice to which the configuration is imported from the drop-down list.
Microservice Version	Select a version of the microservice to which the configuration is imported from the drop-down list.
*Configuration File	<p>Click <b>Import</b> and select the target file.</p> <p>Only JSON files can be imported and the file size cannot exceed 2 MB.</p>

**Step 5** Click **Close**.

----End

## Exporting Configurations

You can export the selected configuration file to a local path to prevent data loss (due to such as server faults and manual deletion) and periodically export configurations for restoration.

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Select the configuration items to be exported and click **Export**.

- Click **Export** above the configuration items. In the displayed dialog box, click **Export**.
- Click **Export** in the upper right corner. In the displayed dialog box, select the file format (V2.0 by default) and click **OK**.

 **NOTE**

If the format of the exported configuration file is V1.0, you need to select the microservice environment from the drop-down list.

----End

## Viewing Configuration Details

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click the target configuration item. The configuration item details page shows the configuration details.

----End

## Viewing Configurations of a Historical Version

You can view the configurations of different historical versions.

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

- Step 4** Click **View Historical Version** in the **Operation** column of a configuration item. Alternatively, click **Historical Versions** on the configuration details page. On the **Historical Versions** page that is displayed, you can view the historical versions of the configuration item. On this page, you can compare the configuration version with the rollback version.

----End

## Comparing Configuration Versions

You can compare different historical versions.

- Step 1** Click the target engine.

- Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

- Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

- Step 4** Click the configuration item to be compared.

- Step 5** Click **View Historical Version**.

- Step 6** In **Historical Versions** on the left, select the historical version to be viewed. A maximum of 100 historical versions can be displayed.

In the **Configuration file** on the right, you can view the differences between the current and historical versions.

----End

## Rolling Back a Version

Roll back to the selected historical version.

- Step 1** Click the target engine.

- Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

- Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

- Step 4** Click the target configuration item.

- Step 5** Click **View Historical Version**.

- Step 6** In **Historical Versions** on the left, select the target historical version.

**Step 7** In **Configuration file** on the right, click **Roll Back to the Selected Version**.

----End

## Disable a configuration item

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** In the **Operation** column of the target configuration item, click **More > Disable**.

**Step 5** Click **OK**.

----End

## Deleting a Configuration Item

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Delete** in the **Operation** column of the target configuration item. You can also click the target configuration item and then click **Delete** on the displayed configuration details page.

**Step 5** Click **OK**.

----End

# 5.9 Configuration Management (Applicable to ServiceComb Engine 1.x)

## 5.9.1 Creating a Configuration for ServiceComb Engine 1.x

Configuration management provides common configurations for microservices, such as log levels and running parameters. After being added, the configuration item is used as the default one if no same configuration items are defined for microservices.

## Creating a Configuration

Configuration management provides common configurations for microservices, such as log levels and running parameters. After being added, the configuration item is used as the default one if no same configuration items are defined for microservices.

 **NOTE**

**Configuration items are stored in plaintext. Do not include sensitive data.**

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Create Configuration Item**.

**Step 5** On the **Create Configuration Item** page, select a microservice environment and enter **Configuration Item** and **Value**.

**Step 6** Click **OK**.

----End

## 5.9.2 Managing Configurations for ServiceComb Engine 1.x

### Editing a Configuration

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Edit** in the **Operation** column of the target configuration item and edit the values of the configuration item.

**Step 5** Click **OK**.

----End

## Importing Configurations

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Import**.

**Step 5** Select a microservice environment, click **Import**, and select the target file. A maximum of 150 configuration items can be imported at a time.

**Step 6** Click **Close**.

----End

## Exporting Configurations

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Click **Export All**.

----End

## Deleting a Configuration

**Step 1** Click the target engine.

**Step 2** Choose **Configuration Management**.

- For engines with security authentication disabled, go to [Step 4](#).
- For engines with security authentication enabled, if the login user is the user imported in [Importing an IAM Account](#), go to [Step 4](#). For other users, go to [Step 3](#).

**Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.

**Step 4** Select the target configuration item and click **Delete**. You can also click **Delete** in the **Operation** column of the target configuration item.

**Step 5** Click **OK**.

----End

## 5.10 System Management

### 5.10.1 Overview

A ServiceComb engine may be used by multiple users. Different users must have different engine access and operation permissions based on their responsibilities and permissions.

The exclusive ServiceComb engine with security authentication enabled provides the system management function using the role-based access control (RBAC) through the microservice console.

The exclusive ServiceComb engine with security authentication enabled supports the access of Spring Cloud and Java chassis microservice frameworks.

#### NOTE

- The RBAC-based system management function is irrelevant to IAM permission management. It is only an internal permission management mechanism of CSE.
  - To operate a ServiceComb engine on CSE, you must have both the IAM and RBAC permissions, and the IAM permission takes precedence over the RBAC permission.
  - If you perform operations on a ServiceComb engine through APIs or the microservice framework, you only need to have the RBAC permissions.
1. You can use an account associated with the **admin** role to create an account and associate a proper role with the account based on service requirements. The user who uses this account has the access and operation permissions on the ServiceComb engine.
    - When you create an exclusive ServiceComb engine with security authentication enabled, the system automatically creates the **root** account associated with the **admin** role. The **root** account cannot be edited or deleted.
    - You can create an account using the **root** account of the ServiceComb engine or an account associated with the **admin** role of the engine. For details about how to create and manage an account, see [Accounts](#).
  2. You can create a custom role using an account associated with the **admin** role and grant proper ServiceComb engine access and operation permissions to the role based on service requirements.
    - The system provides two default roles: administrator (**admin**) and developer (**developer**). Default roles cannot be edited or deleted.
    - You can use the **root** account created for the ServiceComb engine or the account associated with the admin permissions of the ServiceComb engine to create a custom role . For details about how to create and manage a role, see [Roles](#).
    - For details about role permissions, see [Table 5-8](#).

**Table 5-8** Role permissions

Role	Permission Description
Admin	Full permissions for all microservices, accounts, and roles of the ServiceComb engine.
Developer	Full permissions for all microservices of the ServiceComb engine.
Custom role	You can create roles based on service requirements and grant microservice operation and configuration permissions to the roles.

## 5.10.2 Accounts

You can use an account associated with the **admin** role to log in to the ServiceComb engine console and create an account or manage a specified account created in the engine based on service requirements.

**Table 5-9** Account management operations

Operation	Description
<b>Adding an Account</b>	Creates an account and associates a proper role with the account. Users who use the account have the access and operation permissions on the microservice engine.  You can create up to 1000 accounts, including new accounts and the imported IAM account.
<b>Importing an IAM Account</b>	Imports an IAM account and associates roles with it. Users who use this IAM account have the access and operation permissions on the microservice engine.  If the imported IAM account needs to connect microservice applications to the engine through programming interface authentication, <b>reset</b> its password and then use the new password to configure security authentication parameters.  When you use this IAM account to log in to the CSE console with security authentication enabled, you do not need to enter the account and password.  CSE can manage up to 1000 accounts, including new accounts and the imported IAM account.
<b>Viewing Role Permissions</b>	Displays the permissions of the role associated with a specified account.
<b>Editing an Account</b>	Adds or deletes roles for an account. The <b>root</b> account cannot be edited.

Operation	Description
<p><b>Changing the Password</b></p>	<p>Changes the password of an account that has logged in to the ServiceComb engine based on service requirements or security regulations.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If the login user is an admin user, the user can change its own password and the password of a non-admin user and must enter the original password.</li> <li>• If the login user is not an admin user, the user can change its own password and must enter the original password.</li> <li>• If the account and password are used to register a microservice in the SDK, changing the account and password may affect the service running of the microservice (the microservice cannot be registered with the ServiceComb engine). As a result, the service system will be damaged. Exercise caution when performing this operation.</li> <li>• After the password is changed, update the microservice authentication configuration in a timely manner.</li> <li>• After the password is changed, the account may be locked due to three consecutive incorrect password attempts. The account will be unlocked after 15 minutes.</li> </ul>
<p><b>Resetting a Password</b></p>	<p>Based on service requirements or security regulations, you can use the account that has logged in to the ServiceComb engine to reset the passwords of other accounts under the engine.</p> <ul style="list-style-type: none"> <li>• If the account and password are used to register a microservice in the SDK, resetting the account and password may affect the service running of the microservice (the microservice cannot be registered with the ServiceComb engine). As a result, the service system will be damaged. Exercise caution when performing this operation.</li> <li>• After the password is reset, update the microservice authentication configuration in a timely manner.</li> <li>• After the password is reset, the account may be locked due to three consecutive incorrect password attempts. The account will be unlocked after 15 minutes.</li> </ul>
<p><b>Deleting an Account</b></p>	<p>Deletes an account that is no longer used. The <b>root</b> account cannot be deleted.</p> <p>If the account and password are used to register a service in the SDK, deleting the account will affect the service running (the service cannot be registered with the engine). As a result, the service system will be damaged. Exercise caution when performing this operation.</p>

## Adding an Account

Before adding an account, you can create a role based on service requirements. For details, see [Creating a Role](#).

- Step 1** Click the target ServiceComb engine with security authentication enabled.
- Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

 **NOTE**

If you connect to the ServiceComb engine for the first time, enter the account name **root** and the password entered when [Creating a ServiceComb Engine](#).

- Step 3** Choose **Accounts > Create Account** and configure account parameters by referring to the following table:

Parameter	Description
Account	Enter an account name. The account name cannot be changed once the account is created.
Role	Select a role based on service requirements. An account can be associated with up to five roles.
Password	Enter the password.
Confirm Password	Enter the password again.

- Step 4** Click **OK**.

----End

## Importing an IAM Account

Before importing an IAM account, you can create a role based on service requirements. For details, see [Creating a Role](#).

- Step 1** Click the target ServiceComb engine with security authentication enabled.
- Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.
- Step 3** Choose **Accounts > Import IAM User Name**.
- Step 4** Select the IAM account to be imported and select account roles. An account can be associated with up to five roles.
- Step 5** Click **Confirm**.

The imported account cannot be used for login using a password. If you want to use the imported IAM account to connect microservice applications to the engine through programming interface authentication, [Resetting a Password](#) first.

----End

## Viewing Role Permissions

- Step 1** Click the target ServiceComb engine with security authentication enabled.
- Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.
- Step 3** Click the role in the **Role** column of the account to be viewed in the account list. On the displayed page, view the role and permission configuration associated with the account.

----End

## Editing an Account

- Step 1** Click the target ServiceComb engine with security authentication enabled.
- Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.
- Step 3** On the **Accounts** tab page, click **Edit Account** in the **Operation** column of the account to be edited.
- Step 4** Select a role based on service requirements. An account can be associated with up to five roles.
- Step 5** Click **Save**.

----End

## Changing the Password

- Step 1** Click the target ServiceComb engine with security authentication enabled.
- Step 2** Choose **System Management**.
- Step 3** In the displayed **Security Authentication** dialog box, enter the account name and password, and click **OK**.
- Step 4** On the **Accounts** tab, select the account for logging in to the ServiceComb engine and click **Reset Own Password** in the **Operation** column.
  1. Enter the old password and a new password, and confirm the password.
  2. After confirming that the password needs to be changed, select **I Understand**.

### NOTE

You can also click **Reset Own Password** in the upper right corner of the **System Management** page to change the password of the current login account.

**Step 5** Click **Save**.

----End

## Resetting a Password

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Accounts** tab page, select the account whose password is to be reset, and click **Reset Password** in the **Operation** column.

1. Enter a new password and confirm the password.
2. After confirming that the password needs to be reset, select **I Understand**.

**Step 4** Click **Save**.

----End

## Deleting an Account

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Accounts** tab page, click **Delete** in the **Operation** column of the account to be deleted.

**Step 4** In the displayed dialog box, enter **DELETE** and click **OK**.

----End

## 5.10.3 Roles

In addition to the default roles **admin** and **developer**, you can use a ServiceComb engine account associated with the **admin** role to log in to the CSE console and perform operations based on service requirements.

### Creating a Role

Creates a role and configures permission actions for the role in different service and configuration groups.

A maximum of 100 roles can be created.

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Roles** tab page, click **Create Role**.

**Step 4** Enter a role name. The role name cannot be changed once the role is created.

**Step 5** Configure permissions.

1. Set **Permission Group**.



a. Set the service permissions.

▪ If you select **All Services**:

You can perform corresponding permission actions on all microservices of the ServiceComb engine.

▪ If you select **Custom Service Groups**, set the parameters according to [Table 5-10](#).



**Table 5-10** Custom service group operations

Operation	Description
Adding a Matching Rule	<p>Click <b>Add Service Group Matching Rule</b>. Select <b>Application</b>, <b>Environment</b>, and <b>Service</b> based on service requirements to filter the microservices on which the role can perform permission actions.</p> <ul style="list-style-type: none"> <li>○ A maximum of 20 microservice matching rules can be set for a custom service group.</li> <li>○ If multiple matching rules are set for a custom service group, the role has the operation permission on those microservices which meet any of the matching rules.</li> </ul> <p><b>Application</b>, <b>Environment</b>, and <b>Service</b> are three parameters of a microservice:</p> <ul style="list-style-type: none"> <li>○ If only one parameter is set for a single matching rule, the role has the operation permission on the microservice that matches the parameter value. For example, if you add <b>Environment: production</b>, the role has the operation permission only on the microservice whose environment name is <b>production</b>.</li> <li>○ If more than one parameter is set for a single matching rule, the role has the operation permission on the microservices that match all parameter values. For example, if you add <b>Environment: production Application: abc</b>, the role has the operation permission on the microservice whose environment name is <b>production</b> and application name is <b>abc</b>.</li> <li>○ When automatic discovery is enabled, microservices query the instance addresses of services such as the registry center, configuration center, and dashboard through the registry center. When you grant the query permission to a microservice, the permission of the default application must be included. In this case, add the matching rule <b>Application: default</b>.</li> </ul> <p>After the microservice matching rule is set, click <b>OK</b>.</p>
Editing a Matching Rule	<p>Click  next to the matching rule to be edited. You can reconfigure <b>Service Group</b> and <b>Action</b> of the matching rule based on service requirements.</p> <p>After the service group matching rule is set, click <b>OK</b>.</p>
Deleting a Matching Rule	<p>Click  next to the matching rule to be deleted. You can delete the matching rule based on service requirements.</p>

- b. Set the configuration permissions.
  - If you select **All Configurations**:  
You can perform corresponding permission actions on all configurations of the ServiceComb engine.
  - If you select **Custom Configuration Groups**, set the parameters according to [Table 5-11](#).

**Table 5-11** Custom configuration group operations

Operation	Description
Adding a Matching Rule	<p>Click <b>Add Configuration Group Matching Rule</b>. Select <b>Application</b>, <b>Environment</b>, and <b>Service</b> based on service requirements to filter the configurations on which the role can perform permission actions. If application-level and microservice-level configurations cannot meet service requirements, you can customize a matching rule to match the configured custom labels and filter the permission actions that can be performed by the role.</p> <ul style="list-style-type: none"> <li>○ A maximum of 20 matching rules can be set for a custom configuration group.</li> <li>○ If multiple matching rules are set for a configuration service group, the role has the operation permission on those configurations which meet any of the matching rules.</li> </ul> <p><b>Application</b>, <b>Environment</b>, and <b>Service</b> are three parameters of a configuration:</p> <ul style="list-style-type: none"> <li>○ If only one parameter is set for a single matching rule, the role has the operation permission on the configuration that matches the parameter value. For example, if you add <b>Environment: production</b>, the role has the operation permission only on the configuration whose environment name is <b>production</b>.</li> <li>○ If more than one parameter is set for a single matching rule, the role has the operation permission on the configurations that match all parameter values. For example, if you add <b>Environment: production Application: abc</b>, the role has the operation permission on the configuration whose environment name is <b>production</b> and application name is <b>abc</b>.</li> </ul> <p>After the configuration matching rule is set, click <b>OK</b>.</p>

Operation	Description
Editing a Matching Rule	Click  next to the matching rule to be edited. You can reconfigure <b>Configuration Group</b> and <b>Action</b> of the matching rule based on service requirements. After the configuration group matching rule is set, click <b>OK</b> .
Deleting a Matching Rule	Click  next to the matching rule to be deleted. You can delete the matching rule based on service requirements.

2. Set **Action**.

Configure the permission actions that can be performed by the role on the selected service group and configuration group based on service requirements. You can select multiple permission actions.

- **All**: Add, delete, modify, and query resources in the service group and configuration group.
- **Add**: Add resources to the service group and configuration group.
- **Delete**: Delete resources from the service group and configuration group.  
If only **Delete** is selected, you cannot delete resources in the service group and configuration group. You must select **View** at the same time.
- **Modify**: Modify resources in the service group.  
If only **Modify** is selected, you cannot modify resources in the service group and configuration group. You must select **View** at the same time.
- **View**: View resources in the service group and configuration group.

**Step 6** Click **Create**.

----End

## Editing a Role

Modifies the permissions of the created role.

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Roles** tab page, click **Edit** in the **Operation** column of the role to be edited.

**Step 4** Modify **Service Group**, **Configuration Group**, and **Action** based on service requirements.

**Step 5** Click **Save**.

----End

## Deleting a Role

Deletes a role that is no longer used.

- Deleted roles cannot be restored. Exercise caution when performing this operation.
- Before deleting a role, ensure that the role is not associated with any account. For details about how to cancel the association between a role and an account, see [Editing an Account](#).

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Roles** tab, click **Delete** in the **Operation** column of the role to be deleted. In the displayed dialog box, enter **DELETE** and click **OK**.

- Deleted roles cannot be restored. Exercise caution when performing this operation.
- Before deleting a role, ensure that the role is not associated with any account. For details about how to cancel the association between a role and an account, see [Editing an Account](#).


----End

## Viewing a Role

View the created roles of the ServiceComb engine based on the keyword of the role name.

**Step 1** Click the target ServiceComb engine with security authentication enabled.

**Step 2** Choose **System Management**. In the displayed **Security Authentication** dialog box, enter the name and password of the account associated with the **admin** role under the ServiceComb engine, and click **OK**.

**Step 3** On the **Roles** tab page, click  next to the role to be viewed to expand the role details.

**Service Group**, **Configuration Group**, and **Action** of the role are displayed.

----End

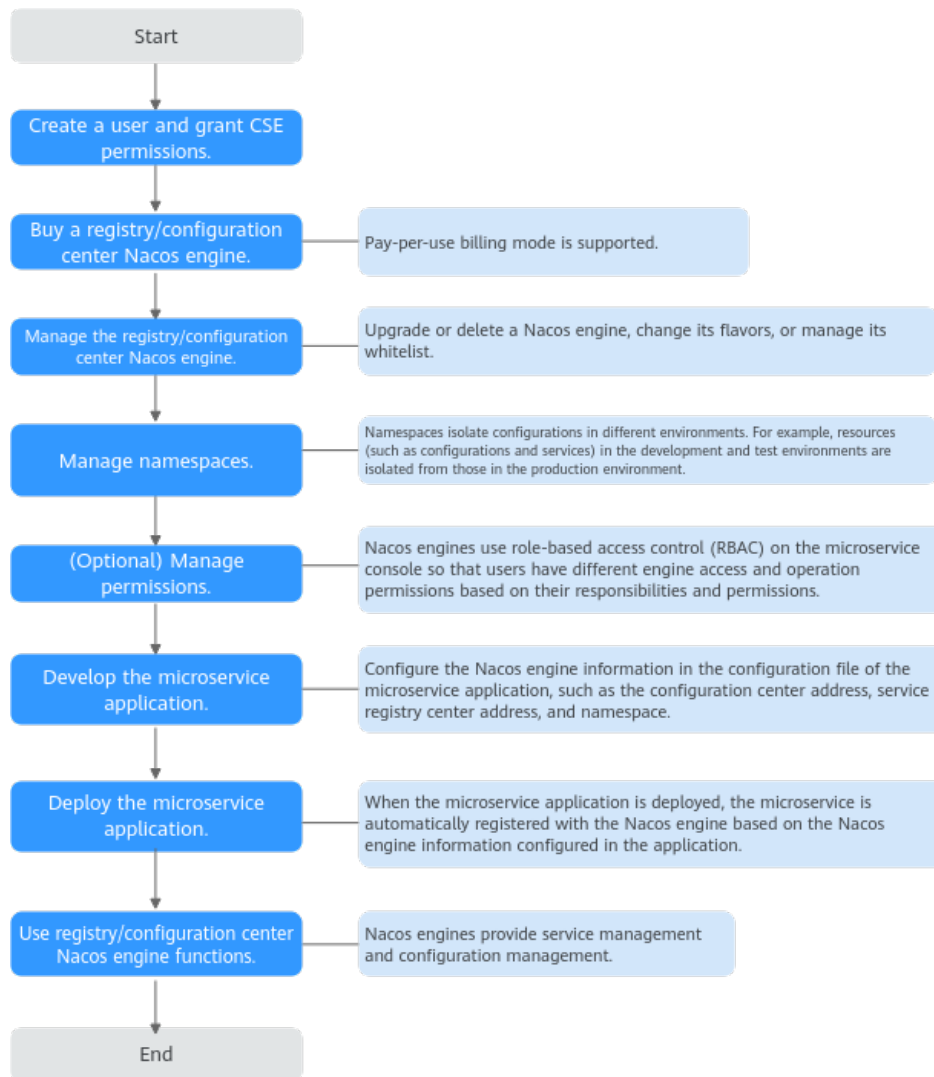
# 6 Nacos Engines

---

## 6.1 Nacos Engine Overview

CSE Nacos is compatible with open-source Nacos and Eureka clients. It provides registry and discovery, dynamic configuration management, access permission control, and observability. It can be used to build highly available and easy-to-manage microservice middleware. Nacos is the infrastructure for service governance and configuration management in the microservice architecture. It is especially suitable for complex distributed systems that require dynamic adjustment, high availability, and multi-environment adaptation.

The following figure shows how to use Nacos as the registry/configuration center.



1. **Create a user and grant CSE permissions.**
2. **Create a registry/configuration center.**
3. Upgrade, back up, restore, or delete a Nacos engine or change its flavors by referring to **Managing Nacos Engines**.
4. **Manage namespaces** to isolate configurations in different environments.
5. (Optional) **Manage permissions** so that different users have different engine access and operation permissions based on their responsibilities and permissions.
6. Develop the microservice application. Configure the Nacos engine information in the configuration file of the microservice application, such as the configuration center address, service registry center address, and namespace.
7. Complete microservice deployment. When the microservice starts, it is automatically registered with the Nacos engine.
8. Use service management and configuration management provided by the Nacos engine by referring to **Managing Nacos Engine Services** and **Managing Nacos Engine Configurations**.

## 6.2 Creating a Nacos Engine

Nacos engines resolve the pain points of registration discovery and configuration management in the microservice architecture and provide lightweight service governance capabilities. You can create a CSE Nacos engine on the console with just a few clicks, reducing O&M costs and allowing you to focus on your services. This section describes how to create an engine whose registry/configuration center is Nacos.

### Restrictions

- Cluster nodes in the registry/configuration center are evenly distributed to different AZs. A failure of a single node does not affect external services.
- You cannot use a shared VPC to create a Nacos engine. Otherwise, the engine fails to be created.
- The VPC cannot be changed once the Nacos engine is created.
- You may fail in buying an engine for insufficient underlying resources. To prevent this, delete engines in time so that you will not be charged if your engines restore upon sufficient underlying resources.
- By default, a maximum of five ServiceComb engines can be created for each project. ServiceComb engines share quotas (five engines) with Nacos engines.

### Prerequisites

- The login account has the permission to create a microservice engine. For details, see [Creating a User and Granting CSE Permissions](#).
- A Nacos engine runs on a VPC. Before creating a Nacos engine, ensure that VPCs and subnets are available. For details, see "Creating a VPC with a Subnet" in *Virtual Private Cloud User Guide*.
- The user has the CSE FullAccess and DNS Full Access permissions.

### Creating a Registry/Configuration Center

**Step 1** Log in to CSE.

**Step 2** In the left navigation pane, choose **Registry/Configuration Center** and click **Buy Registry/Configuration Center Instance**.

**Step 3** Set parameters according to the following table. Parameters marked with an asterisk (\*) are mandatory.

Parameter	Description
*Billing Mode	Billing mode. Currently, <b>Pay-per-use</b> is supported.

Parameter	Description
*Enterprise Project	<p>Select the project where the Nacos engine is located. You can search for and select the required enterprise project from the drop-down list.</p> <p>Enterprise projects let you manage cloud resources and users by project.</p> <p>An enterprise project can be used after it is created and enabled. For details, see . By default, <b>default</b> is selected.</p> <p>After a registry/configuration center Nacos engine is created, you can remove Nacos engine resources out of the current enterprise project and into a new enterprise project. For details, see "Removing Resources from an Enterprise Project" in <i>Enterprise Management User Guide</i> and "Adding Resources to an Enterprise Project" in <i>Enterprise Management User Guide</i>.</p>
*Name	<p>Enter a Nacos engine name. The name contains 3 to 24 characters, including letters, digits, and hyphens (-), and starts with a letter but cannot end with a hyphen (-). The name cannot be <b>default</b>.</p>
*Registry/Configuration Center Instance	<p>Select <b>Nacos</b>.</p>
*Instances	<p>Select the required capacity specifications.</p>
Version	<p>Only the latest version can be created.</p>
*Network	<p>Select a VPC and subnet to provision logically isolated, configurable, and manageable virtual networks for your engine.</p> <ul style="list-style-type: none"> <li>• To use a created VPC, search for and select a VPC created under the current account from the drop-down list.</li> <li>• To use a new VPC, create one on the VPC console. For details, see "Creating a VPC with a Subnet" in <i>Virtual Private Cloud User Guide</i>.</li> </ul>
Tag	<p>Tags are used to identify cloud resources. When you have multiple cloud resources of the same type, you can use tags to classify them based on usage, owner, or environment.</p> <p>Click <b>+</b> <b>Add Tag</b>. In the <b>Add Tag</b> dialog box, enter a tag key and value. For details about tag naming rules, see <b>Managing Tags</b>. In the <b>Add Tag</b> dialog box, you can click <b>+</b> <b>Add Tag</b> to add multiple tags at a time, or click <b>-</b> next to a tag to delete the tag.</p>

**Step 4** Click **Buy**. The page for confirming the engine information is displayed.

**Step 5** Click **Submit**. When the status becomes **Available**, the engine is created.

----End

## 6.3 Managing Nacos Engines



### 6.3.1 Viewing Nacos Engine Information

This section describes how to view details about a Nacos instance on the CSE console.

#### Viewing Nacos Engine Information

- Step 1** In the left navigation pane, choose **Registry/Configuration Center**.
- Step 2** Click the target Nacos instance. You can click an **Available** instance to go to the **Basic Information** page.
- Step 3** View the Nacos engine information shown in [Table 6-1](#).

**Table 6-1** Engine details

Type	Parameter	Description
Basic Information	Name	Engine name entered when <a href="#">Creating a Nacos Engine</a> . Click  to copy it.
	ID	Engine ID. Click  to copy it.
	Running Status	Engine status.
	Registry/Configuration Center Instance	Select <b>Nacos</b> .
	Version	Engine version.
	Capacity Specifications	Engine specifications selected when <a href="#">Creating a Nacos Engine</a> . If the engine is a small-scale engine, click <b>Change Specifications</b> on the right to expand the capacity. For details, see <a href="#">Increasing Nacos Engines</a> .
	Enterprise Project	Enterprise project selected when <a href="#">Creating a Nacos Engine</a> .
	Billing Mode	Billing mode selected when <a href="#">Creating a Nacos Engine</a> . Only pay-per-use billing is supported.
	Created	Time when <a href="#">Creating a Nacos Engine</a> .

Type	Parameter	Description
Connection Information	Private IP	Private address of a Nacos engine.
	Private Port	Internal port of a Nacos engine.
	Virtual Private Cloud	VPC selected when you <a href="#">create a registry/configuration center</a> .
	Subnet	Subnet selected when you <a href="#">create a registry/configuration center</a> .
	Whitelist Access	Nacos supports whitelist control. Multiple IP addresses or IP address segments can be configured. IP addresses that are not in the IP address segments cannot be accessed. For details about whitelist access, see <a href="#">Managing the Nacos Engine Whitelist</a> .
More Settings	Tag	Tags added to the Nacos engine. You can also click <b>Tag Management</b> and perform operations on tags as required. For details, see <a href="#">Managing Nacos Engine Tags</a> .

----End

## 6.3.2 Managing Nacos Engine Tags

Tags facilitate Nacos engine identification and management.

If your organization has configured tag policies for Nacos engines, add tags to engines based on the policies. If a tag does not comply with the tag policies, engine creation may fail. Contact your administrator to learn more about tag policies.

You can add tags to a Nacos engine when creating the engine or add tags on the details page of the created engine. Up to 20 tags can be added to an engine. Tags can be modified and deleted.

A tag consists of a tag key and a tag value. [Table 6-2](#) lists the tag key and value requirements.

**Table 6-2** Tag naming rules

Tag	Rule
Key	<ul style="list-style-type: none"> <li>• Cannot be left blank.</li> <li>• Must be unique for the same instance.</li> <li>• Contain a maximum of 128 characters.</li> <li>• Contain only letters, digits, spaces, and special characters ( _ . : = + - @ ).</li> <li>• Cannot start with a space or <b>_sys_</b> or end with a space.</li> </ul>
Value	<ul style="list-style-type: none"> <li>• Contain a maximum of 255 characters.</li> <li>• Contain only letters, digits, spaces, and special characters ( _ . : = + - @ ).</li> </ul>

## Managing Tags

Adding or modifying tags will affect Nacos engine services for about 10 seconds. Add or modify tags during off-peak hours.

**Step 1** Click the target engine. The details page is displayed.

**Step 2** In **Tags** of **More Settings**, you can perform the following operations as required:

- Adding a tag
  - a. Click **Tag Management**. The **Edit Tag** dialog box is displayed.
  - b. Click **⊕ Add Tag** and enter a tag key and value in the text boxes.
  - c. Click **OK**.
- Modifying a tag
  - a. Click **Tag Management**. The **Edit Tag** dialog box is displayed.
  - b. You can modify the tag key and value in the original text boxes.
  - c. Click **OK**.
- Deleting a tag
 

Click **⊖** in the row that contains the tag to be deleted. In the dialog box that is displayed, click **OK** to delete the tag.

----End

### 6.3.3 Managing the Nacos Engine Whitelist

The following describes how to manage whitelists of a Nacos engine to allow access only from whitelisted IP addresses.


If no whitelists are added to the engine whitelist or the whitelist function is disabled, all IP addresses that can communicate with the VPC can access the engine.

## Setting a Whitelist

**Step 1** Click the target engine. The details page is displayed.

 **NOTE**

You can click an **Available** engine to go to the **Basic Information** page.

**Step 2** In the **Connection Information** area, click . In the **Set Access Whitelist** dialog box, enter **IP Address/Address Segment**. Use commas (,) to separate multiple whitelists.

 **NOTE**

A maximum of 20 IP addresses/address segments can be added for each engine.

- To modify or delete an IP address/range, modify or delete it in the displayed **Set Access Whitelist** dialog box.
- To add an IP address/range, add it in the displayed **Set Access Whitelist** dialog box.

**Step 3** Click **OK**. When the engine status changes from **Configuring** to **Available**, the whitelist takes effect.

----End

## 6.3.4 Increasing Nacos Engines

As service traffic increases and the service scale expands, you can adjust the specifications of Nacos engines to ensure stable and high-performance microservice registration, discovery, and configuration management.

You can increase the number of Nacos engines online. Only low-capacity engines support this operation.

### Restrictions

During increase, the interface request may fail for a short period of time. The Nacos framework provides the retry function. If the request fails, the interface will be called again. You are advised to perform the operation in your own change maintenance time window.

### Increasing Nacos Engines

**Step 1** Click **More > Change Specifications** in the **Operation** column of the target Nacos engine instance. Alternatively, click the target engine and click **Change Specifications** next to **Capacity Specifications** in the **Basic Information** area on the **Basic Information** page.

**Step 2** On the **Change Nacos Engine Specifications** page, select the target capacity.

**Step 3** Click **Change Now**, confirm the information, and click **Submit**. When the instance status changes from **Changing** to **Available**, the operation is successful.

----End


## 6.3.5 Upgrading a Nacos Engine

Nacos engines are created using the latest engine version. When a later version is released, you can upgrade your engine.

### Restrictions

- During the Nacos engine upgrade, the microservice and engine are intermittently disconnected, but services of running microservices are not affected. You are advised not to upgrade, restart, or change microservices when upgrading a Nacos engine.
- You can only upgrade to the latest version.

### Upgrading a Nacos Engine

**Step 1** Click  in the **Version** column of the target Nacos engine.

 **NOTE**

If the engine version is the latest,  does not exist in the **Version** column.

**Step 2** In the displayed dialog box, confirm the current and target versions, and click **OK**.

If the upgrade fails, click **Retry** to perform the upgrade again.

----End

## 6.3.6 Deleting a Nacos Engine

You can delete a Nacos engine if it is no longer used. Deleted engines cannot be restored. Exercise caution when performing this operation. You can delete Nacos engines in the following states:

- Available
- Unavailable
- Creation failed
- Resizing failed
- Upgrade failed
- Unknown

### Deleting a Nacos Engine

**Step 1** Click **More > Delete** in the **Operation** column of the target Nacos engine instance. Alternatively, click the target Nacos engine, click **Delete** in the upper right corner on the **Basic Information** page, and enter **DELETE** and click **OK** in the displayed dialog box.

 NOTE

If the deletion fails, click **Force Delete**.

----End

## 6.4 Managing Namespaces

Namespaces isolate configurations in different environments. For example, resources (such as configurations and services) in the development and test environments are isolated from those in the production environment. Different namespaces can have the same group or data ID.

### Restrictions

- The namespace ID is entered in the SDK connected to a Nacos engine. The namespace name is only the identifier used for viewing on the console.
- If your service SDK uses a namespace ID that is not created on the Nacos server for service registration and discovery, the service can be registered and discovered, but cannot be viewed on the service management page of the registry/configuration center. You need to create the corresponding namespace before viewing the service. For details, see [Creating a Namespace](#).

### Prerequisites

You have created a Nacos engine instance. For details, see [Creating a Nacos Engine](#).

### Creating a Namespace

When an instance is created, a default namespace **public** (reserved space) is automatically generated. This namespace cannot be edited or deleted. You can use this namespace to isolate resources and services. Up to 50 namespaces can be created.

- Step 1** Click the target Nacos instance.
- Step 2** Choose **Namespaces** and click **Create Namespace**.
- Step 3** In the displayed dialog box, set the parameters as follows. Configuration items marked with an asterisk (\*) are mandatory.

**Table 6-3** Namespace parameters

Parameter	Description
*Namespace	The namespace name can be customized and can contain a maximum of 128 characters except @ # \$% ^ & *.

Parameter	Description
Namespace ID	<p>Enter a maximum of 128 characters. Use only uppercase letters, lowercase letters, digits, hyphens (-), and underscores (_). The namespace ID must be unique.</p> <p>If no ID is entered during creation, the system randomly generates an ID.</p>

**Step 4** Click **OK**.

----End

## Editing a Namespace

**Step 1** Click the target Nacos instance.

**Step 2** Choose **Namespaces**.

**Step 3** Click **Edit** in the **Operation** column of the namespace to be edited to edit the namespace name.

 **NOTE**

The automatically generated namespace **public** cannot be edited.

**Step 4** Click **OK**.

----End

## Deleting a Namespace

**Step 1** Click the target Nacos instance.

**Step 2** Choose **Namespaces**.

**Step 3** Click **Delete** in the **Operation** column of the namespace to be deleted.

**Step 4** In the displayed dialog box, click **OK**.

----End

# 6.5 Permission Control

## 6.5.1 Permission Control Overview

A Nacos engine may be used by many users. Nacos engines with security authentication enabled provide permissions management using role-based access control (RBAC) on the microservice console, so that users have different engine access and operation permissions based on their responsibilities and permissions.

The Nacos engine with security authentication enabled supports microservice access.

 NOTE

- Only engine 2.1.0.1 and later support this function. For engine earlier than 2.1.0.1, upgrade it to the latest version. For details, see [Upgrading a Nacos Engine](#).
- If the Nacos engine version is upgraded from 2.1.0 to 2.1.0.1 or later, you need to enable security authentication to initialize the key information before using the permission control function.
- Eureka-compatible instances do not support security authentication.

## 6.5.2 Enabling and Disabling Security Authentication

### Enabling Security Authentication

By default, security authentication is disabled for Nacos engines. You can enable security authentication on the console.

After security authentication is enabled, only accessible namespaces are displayed on the console. Clients without usernames and passwords cannot access Nacos instances. Exercise caution when performing this operation.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** Click **Set Authentication** and enable **Authenticate Programming Interface**.

**Step 4** Click **OK**. After the Nacos engine is updated and the engine status changes from **Configuring** to **Available**, security authentication is enabled successfully.

----End

### Disabling Security Authentication

After security authentication is disabled, permissions of each user cannot be controlled. Clients can access Nacos instances without passwords, and all namespaces are displayed on the console. Exercise caution when performing this operation.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** Click **Set Authentication**. On the **Security Settings** page, disable **Authenticate Programming Interface**.

**Step 4** In the displayed dialog box, click **OK**. When the status of the engine changes to **Available**, security authentication is disabled.

----End

## 6.5.3 Accounts

You can log in to the engine console and create an account or manage a specified account created on the engine based on service requirements.

## Creating an Account

Create an account and associate a proper role with the account. Users who use the account have the access and operation permissions on the Nacos engine.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** Choose **Accounts > Create Account** and configure account parameters by referring to the following table:

Parameter	Description
Account	Enter an account name. The account name cannot be changed once the account is created.
Password	Enter a password.
Confirm Password	Enter the password again.

**Step 4** Click **OK**.

----End

## Resetting a Password

For security purposes, you can reset your password on the console.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** On the **Accounts** tab page, click **Reset Password** in the **Operation** column of the target account.

**Step 4** Enter and confirm a new password, select **I Understand**, and click **Save**.

----End

## Deleting an Account

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** On the **Accounts** tab page, click **Delete** in the **Operation** column of the target account. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

## 6.5.4 Roles

You can log in to the engine console and create, edit, delete, and view roles of a Nacos engine based on service requirements. Permission control by namespace or finer granularity is supported.

### Creating a Role

**Step 1** Click the target Nacos engine.


**Step 2** Choose **Permission Control**.

**Step 3** On the **Roles** tab page, click **Create Role**.

**Step 4** Enter a role name. The role name cannot be changed once the role is created.

**Step 5** Set **Associated user**. Select the user created in [Creating an Account](#) from the drop-down list.

**Step 6** Add permission configurations. The role also has the permissions configured here.

Click  **Add Permission Configuration** and select a namespace and action (**Read only**, **Write only**, or **Read/write**). You can add multiple permission configurations at a time or click **Delete** in the **Operation** column of a permission configuration to delete it.

**Step 7** Click **Create**.

----End

### Editing a Role

**Step 1** Click the target Nacos engine with security authentication enabled.

**Step 2** Choose **Permission Control**.

**Step 3** On the **Roles** tab page, click **Edit** in the **Operation** column of the role to be edited.

**Step 4** Modify **Namespace** and **Action** based on service requirements.

**Step 5** Click **Edit**.

----End

### Deleting a Role

Deleted roles cannot be restored. Exercise caution when performing this operation.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** On the **Roles** tab, click **Delete** in the **Operation** column of the role to be deleted. In the displayed dialog box, enter **DELETE** and click **OK**.

----End

## 6.5.5 Console Resource Management


Nacos engines support the association between namespaces and enterprise projects. The relationship is N:1, that is, N namespaces can be associated with one enterprise project.

By default, the namespace created in [Creating a Namespace](#) is not associated with any enterprise project. You can associate the namespace with an enterprise project by editing the enterprise project.

When editing an enterprise project, you can only change the enterprise project and cannot leave the enterprise project empty.

**Step 1** Click the target Nacos engine.

**Step 2** Choose **Permission Control**.

**Step 3** On the **Console Resource Management** tab, click  in the **Enterprise Project** column of the target namespace. In the **Edit Enterprise Project** dialog box, select an enterprise project from the drop-down list and click **OK**.

----End

## 6.6 Managing Nacos Engine Services

You can use the CSE console to manage services registered with Nacos.

### Prerequisites

A Nacos engine instance has been created.

### Creating a Service

You can create a service on the console. The newly created service is an empty service (that is, the number of providers is 0). By default, the empty service is displayed in the service list. If you do not want to display the empty service, click

 next to **Hide Empty Service**.

**Step 1** Click the target Nacos instance.

**Step 2** Choose **Service Management**.

**Step 3** Select a namespace from the Namespace drop-down list. The ID is automatically filled in the Namespace ID box.

#### NOTE

If the selected namespace is **public**, the namespace ID is empty by default.

**Step 4** Click **Create Service**. In the displayed dialog box, set configuration items as follows. Configuration items marked with an asterisk (\*) are mandatory.

**Table 6-4** Parameters

Parameter	Description
*Service Name	Enter a service name. The value can contain a maximum of 236 characters, including digits, letters, and special characters "_-:.".
Group	Set the group to which the service belongs. The value can contain a maximum of 128 characters, including digits, letters, and special characters "_-:.".
*Protection Threshold	If the ratio of healthy instances to the total instances is less than the threshold, a protection threshold is triggered. The value ranges from 0 to 1. The default value is 0.

**Step 5** Click **OK**.

----End

## Viewing the Service List

**Step 1** Click the target Nacos instance.

**Step 2** Choose **Service Management**. Select a namespace from the **Namespace** drop-down list. The ID is automatically filled in the **Namespace ID** box.

 **NOTE**

If the selected namespace is **public**, the namespace ID is empty by default.

**Step 3** View all services in the namespace of the engine.

You can search for the target service by service name or group name.

 **NOTE**

Target service fuzzy search supports characters: ,\$.+.|?

----End

## Viewing Service Details

**Step 1** Click the target Nacos instance.

**Step 2** Choose **Service Management**.


**Step 3** Click the target service to view its details.

- View basic service information, including the service name, namespace name, service group, namespace ID, protection threshold, and number of clusters.

- The **Instances** tab displays the instance information, including the IP address, port number, cluster, health status, online/offline status, weight, and metadata. You can also perform **Instance Operations**, such as searching for instances based on metadata, bringing instances online/offline, and modifying weights.
- The **Subscribers** tab displays the list of all client instances that subscribe to the current service. Versions of subscribers and clients are displayed in the list.

----End

## Instance Operations

- Search by metadata: On the **Instances** tab, select a cluster from **Clusters**, enter the metadata key and value in **Search Metadata**, and click **Filter** to display the instances that meet the search criteria. Click **Clear** to clear the search data.
- Bring an instance online or offline: On the **Instances** tab, click **Online** or **Offline** in the **Operation** column of the target instance. The instance status will be updated accordingly.
- Modify instance weight: On the **Instances** tab, move the cursor to the **Weight** column of the target instance, click  to modify the weight (ranging from 1 to 99), and click **OK**.

### NOTE

To use the Nacos weight function for traffic load balancing, register **NacosRule** provided by Nacos as a bean on the client.

```
@Bean
NacosRule nacosRule() {
    return new NacosRule();
}
```

Add the following configuration item to the **application.properties** configuration file:  
xxx-service.ribbon.NFLoadBalancerRuleClassName=com.alibaba.cloud.nacos.ribbon.NacosRule

**xxx-service** indicates the service name of the client, that is,  
spring.application.name=xxx-service

## Deleting a Service

- Only empty services can be deleted. If the number of instances is not 0, the services cannot be deleted.
- If a service remains empty for more than 1 minute, the Nacos automatically deletes the service.

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Service Management** and click **Delete** in the **Operation** column of the target service.

**Step 3** In the displayed dialog box, click **OK**.

----End

## 6.7 Managing Nacos Engine Configurations

## 6.7.1 Nacos Engine Configuration Overview

In Nacos engines, configuration management is one of the core functions. It is used to solve problems such as centralized configuration management, dynamic update, and environment isolation in the microservice architecture. It uses a unified configuration center to adjust application behavior in real time without modifying code or restarting services, improving system flexibility and maintainability.

Nacos configuration management functions:

- Centralized configuration management
  - Application configurations are stored on the Nacos server, which prevents configurations from being scattered in code or local files and facilitates centralized maintenance and management.
  - Configurations can be managed by application, environment (development, test, and production), and cluster, improving configuration isolation.
- Dynamic configuration update

Configurations can be pushed to applications in real time after being modified, without restarting services. (This feature requires the support of the application SDK.)
- Configuration version management

Configuration changes are automatically recorded, and configurations can be rolled back to a previous version, preventing configuration faults caused by misoperations.
- Configuration listening and pushing

Applications listen to Nacos configuration changes through SDKs, and Nacos proactively pushes updates to ensure that configurations take effect in real time.
- Multi-environment support

Configurations can be isolated in different environments (development, test, and production). The same application can load different configurations in different environments.

## 6.7.2 Creating a Nacos Engine Configuration

Creating configurations in Nacos is the basis for centralized management, dynamic update, and multi-environment isolation of microservice configurations. This prevents scattered and disordered configurations, improving O&M efficiency and system stability. This section describes how to create a configuration file for the Nacos engine.

---

**NOTICE**

**Configuration items are stored in plaintext. Encrypt sensitive data.**

---

**Step 1** Click the target Nacos instance.

- Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.
- Step 3** Select a namespace from the Namespace drop-down list. The ID is automatically filled in the Namespace ID box.

 **NOTE**

If the selected namespace is **public**, the namespace ID is empty by default.

- Step 4** Click **Create Configuration**. In the displayed dialog box, set the following parameters. Parameters marked with an asterisk (\*) are mandatory.

**Table 6-5** Parameters

Parameter	Description
*Data ID	The data ID is one of the dimensions for identifying configurations, and usually identifies configuration sets of a system. A system or application can contain multiple configuration sets, and each one can be identified by a name. A unique data ID is generally named like a Java package. This naming rule is optional.  The value can contain a maximum of 255 characters, including digits, letters, and special characters "_-:.".
Group	It is a set of configurations in Nacos and one of the dimensions for identifying configurations.  The value can contain a maximum of 128 characters, including digits, letters, and special characters "_-:.".
Namespace	Namespace to which the configuration belongs.
Configuration Format	Nacos supports online editing of common configuration formats such as YAML, Properties, TEXT, JSON, XML and HTML. The default value is <b>TEXT</b> .
*Configuration Content	Enter the configuration content.  The configuration content cannot exceed 100 KB. If the configuration content is too large, split the configuration.  Adjusting the configuration content size may affect Nacos stability. Exercise caution when performing this operation.

Parameter	Description
Description	Enter the description. Enter up to 128 characters.
Application	Enter the application to which the configuration belongs. The value can contain a maximum of 128 characters, including digits, letters, and special characters "_-:".
Label	Enter a label. The value can contain a maximum of 64 characters, including digits, letters, and special characters "_-:".

**Step 5** Click **Release**.

----End


## 6.7.3 Managing Nacos Engine Configurations

### Querying Nacos Engine Configurations

CSE Nacos allows you to query configurations by data ID, group, application, and label.

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** In the filter box above the configuration list, filter configurations by data ID, group, application, and label, and click  to display the configurations that meet the filter criteria.

----End

### Viewing Nacos Engine Configuration Details

You can view configuration details about a Nacos engine on the CSE console.

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Click the target data ID. On the **Configuration Details** page displayed, view the configuration details. In the **Configuration Content** area, click **search** to query the configurations.

----End

## Editing Nacos Engine Configurations

When the configuration is in a dark launch status, roll back or release the configuration first.

- Step 1** Click the target Nacos instance.
- Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.
- Step 3** Edit a configuration in either of the following methods:
  - Click **Edit** in the **Operation** column of the target data ID.
  - Click the target data ID. On the **Configuration Details** page displayed, click **Edit**.
- Step 4** On the **Edit Configuration** page, modify the configuration content, format, description, application, and label. Click **Release**. The **Configuration Content Comparison** dialog box is displayed. You can view the differences between the historical and current versions.
- Step 5** Click **Release**. The **Edit Configuration** page also provides dark launch. For details, see [Managing Dark Launch of Nacos Engine Configurations](#).

----End

## Importing Nacos Engine Configurations

You can import a local ZIP configuration file on the console to simplify the process of creating configurations manually. To reuse configurations among development, test, and pre-release environments, or to migrate a Nacos engine across clouds or platforms, you can export the configurations of the production environment first and then import the configuration file to the target engine to initialize the configurations in batches. This shortens the environment setup. If the configurations are deleted or tampered with, or the Nacos cluster is faulty, you can import the backup configuration file to quickly rebuild the service configuration environment.

### NOTE

- After importing the configuration file, the handling of existing configurations depends on the specified processing policy, such as terminating the import, skipping the conflicting configurations, or overwriting them.
- Before importing the configuration file, back up the existing target environment configurations. Import the file incrementally, and verify core service configurations in a small scope before proceeding with batch imports.
- If you need to modify the configurations, you are advised to import the exported configuration file to the target engine and modify the configurations on the console. For details, see [Editing Nacos Engine Configurations](#).

- Step 1** Click the target Nacos instance.
- Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.
- Step 3** Click **Import Configuration** and set parameters by referring to the following table.

**Figure 6-1** Importing Configurations

**Import Configuration**

Same Configuration

Configuration File

Parameter	Description
Same Configuration	<ul style="list-style-type: none"> <li>• <b>Terminate:</b> If a configuration is the same as that in the system, the import terminates.</li> <li>• <b>Skip:</b> During import, if a configuration is the same as that in the system, the configuration is skipped and other configurations are imported.</li> <li>• <b>Overwrite:</b> During import, if a configuration is the same as that in the system, the value of the configuration will be replaced.</li> </ul>
Configuration File	<p>Click <b>Import</b> and select the target file.</p> <p>The file size cannot exceed 2 MB. If the file is too large, divide it into smaller files and import them individually.</p>

**Step 4** Click **Close**.

 **NOTE**

- If **Same Configuration** is **Terminate**, the **Terminate** dialog box will be displayed if a configuration is the same as that in the system during the import. Click **OK** to terminate the import.
- If **Same Configuration** is **Skip**, the configuration that is the same as that in the system will be skipped during the import, and other configurations are imported. Then, a dialog box is displayed showing the imported configurations. Click **OK**.

----End

## Exporting Nacos Engine Configurations

Export Nacos configurations to a file (such as properties, YAML, or JSON) as offline backup storage to prevent data loss caused by Nacos service faults, misoperations, or configuration deletion by mistake. During engine migration, export the original engine configurations and import them to the new engine in batches, avoiding the trouble of manually rebuilding configurations.

**Step 1** Click the target Nacos instance.

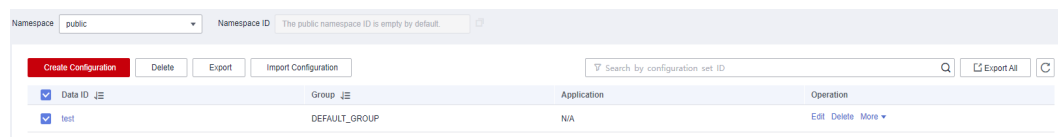
**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Select the target configuration and click **Export**.

 **NOTE**

- Click **Export All** to export all configurations.
- You are advised to export the configurations separately to ensure that the size of an exported configuration file does not exceed 2 MB.

**Figure 6-2** Exporting Configurations



**Step 4** In the displayed dialog box, click **Export**.

----End

## Deleting Nacos Engine Configurations

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Delete a configuration in either of the following methods:

- Click **Delete** in the **Operation** column of the target data ID.
- Select the target data ID and click **Delete** above.

**Step 4** Click **OK**.

----End

## 6.7.4 Managing Dark Launch of Nacos Engine Configurations

The CSE Nacos configuration center supports dark launch. That is, configurations can be partially verified before official release. After verification, configurations will be officially released to reduce the risk of configuration push.

### Configuring Dark Launch

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Click **Edit** in the **Operation** column of the target configuration item.

**Step 4** On the **Edit Configuration** page, click  to enable dark launch.

**Step 5** Select the IP address of the instance to be pushed in dark launch in the text box or manually enter the IP address of the instance for dark launch. Click **Enter**.

 **NOTE**

Multiple instance IP addresses can be configured at the same time.

**Step 6** Click **Release**. In the displayed **Configuration Content Comparison** dialog box, compare the configurations between the historical and current versions.

**Step 7** Click **Release**.

----End

## Viewing Dark Launch Version Configurations

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Click **Edit** in the **Operation** column of the target configuration item that is being dark launched.

**Step 4** On the **Dark Launch Version** tab of the **Edit Configuration** page, you can view the dark launch version configuration, and roll back and release dark launch. For details, see [Related Operations](#).

----End

## Related Operations

- Roll back dark launch: On the **Dark Launch Version** tab of the **Edit Configuration** page, click **Roll Back** to cancel dark launch and roll back to the historical version.
- Release dark launch: On the **Dark Launch Version** tab of the **Edit Configuration** page, click **Release**. In the **Configuration Content Comparison** dialog box, confirm the configuration and click **Release**. The dark launch version becomes an official version.

## 6.7.5 Managing Historical Nacos Engine Versions

CSE Nacos allows you to view details about historical versions and roll back historical versions.

### Viewing Historical Versions

You can view the configuration changes of historical versions to trace and audit the changes. If a problem occurs after the configuration is modified, for example, a service error is reported or the performance deteriorates, you can view the historical versions, compare the current configuration with the normal configuration, and quickly locate the configuration item and modification operation. In addition, Nacos supports one-click rollback to a stable version, which can quickly restore configurations and reduce the impact of faults on services.

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Go to the **Historical Versions** page in either of the following methods. Click the data ID of a historical version in a time segment to view the historical version information of the configuration item. Historical versions can be retained for a maximum of 30 days.

- In the **Operation** column of the target data ID, choose **More > Historical Versions**.
- Click the target data ID. On the **Configuration Details** page displayed, click the **Historical Versions** tab.

----End

## Rolling Back a Historical Version

CSE Nacos allows you to roll back a historical version to help you quickly restore incorrect configurations, reducing potential risks in configuration management of the microservice system.

**Step 1** Click the target Nacos instance.

**Step 2** In the left navigation pane, choose **Configuration Management > Configurations**.

**Step 3** Go to the **Historical Versions** page in either of the following methods.

- In the **Operation** column of the target data ID, choose **More > Historical Versions**.
- Click the target data ID. On the **Configuration Details** page displayed, click the **Historical Versions** tab.

**Step 4** Click **Roll Back** in the **Operation** column of the target historical version. The **Historical Version Details** page is displayed.

### NOTE

Only the configuration whose **Operation** is **Update** can be rolled back.

**Step 5** In the **Configuration Content** area, click **Roll Back to the Selected Version**. In the displayed dialog box, click **OK**.


----End

## 6.7.6 Using the Listening and Query Function of the Nacos Engine

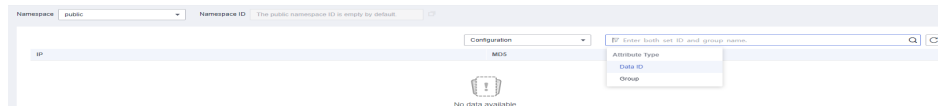
CSE Nacos provides listening query. That is, after modifying a configuration, you need to check whether the modified configuration information has been pushed to the host that listens to the configuration. This helps you better check whether the configuration change has been pushed to the client.

**Step 1** Click the target Nacos instance.

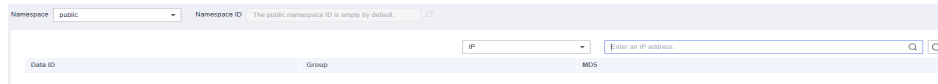
**Step 2** In the left navigation pane, choose **Configuration Management > Listening Queries**.

**Step 3** Select a namespace from the **Namespace** drop-down list, select search criteria, and click  to query listening information.

- If you select **Configuration** from the drop-down list, enter the data ID and group name in the text box to query the hosts to which the configuration is pushed and the push status.



- If you select **IP** from the drop-down list, enter the IP address of the listened host is configured in the text box to query all listened configurations of the host.



----End

## 6.8 Viewing Monitoring of a Nacos Engine

When using the Nacos engine, you can view common metrics related to the Nacos engine in the configuration center and registry center on the running monitoring page provided by the CSE console. This helps you learn about the running status of the Nacos cluster in real time, identify performance bottlenecks and fault risks in advance, and ensure the stability and availability of the service registration and discovery and configuration management functions. This section describes how to view monitoring metrics of Nacos engines.

### Restrictions

To view the running Nacos monitoring data, ensure that you have the AOM FullAccess permission.

### Viewing Monitoring of a Nacos Engine

**Step 1** Click the target Nacos engine.

**Step 2** In the left navigation pane, choose **Monitoring**. You can view Nacos monitoring metrics.

- On the **Dashboard** tab page, you can view **Microservice Instances** and **Configurations**. In addition, **Microservice Instance Use** and **Configuration Use** are displayed in graphics.
  - **Microservice Instance Use**: Ratio of the number of connected microservice instances to the recommended max. number of microservice instances. Increase instances for a high ratio.
  - **Configuration Use**: Ratio of created configurations to the max. configurations allowed.
- On the **Configuration Center Monitoring** tab page, you can select a time from the drop-down list in the upper right corner to view the monitoring data of the configuration center in a specified period, including Configurations, Long Connections, Write Request Frequency, Read Request Frequency, Average Response Time for Write Request, Average Response Time for Read Request, and Configuration Push Time Required. The value can be Last 30 minutes, Last 1 hour, Last 6 hours, Last 1 day, or Last week. By default, the monitoring data of the last 30 minutes is displayed.
- On the **Registry Center Monitoring** tab page, you can select a time from the drop-down list in the upper right corner to view the monitoring data of the

registry center in a specified period, including Microservices, Microservice Instances, Write Request Frequency, Read Request Frequency, Average Response Time for Write Request, Average Response Time for Read Request, and Service Push Time Required. The value can be Last 30 minutes, Last 1 hour, Last 6 hours, Last 1 day, or Last week. By default, the monitoring data of the last 30 minutes is displayed.

----**End**

# 7 Key Operations Recorded by CTS

## 7.1 CSE Operations That Can Be Recorded by CTS

With Cloud Trace Service (CTS), you can query, audit, and review operations performed on cloud resources. Traces include the operation requests sent using the management console or APIs as well as the results of these requests.

To collect, record, or query operation logs of Nacos and ServiceComb engines, first. With CTS, you can view operation records of Nacos and ServiceComb engines in the last seven days. [Table 7-1](#) and [Table 7-2](#) list the supported operation logs.

**Table 7-1** Nacos engine operations that can be recorded by CTS

Operation	Resource Type	Event Name
Creating an engine	engine	CreateEngineJob
Deleting an engine	engine	DeleteEngineJob
Creating a service	service	createService
Modifying a service	service	modifyService
Deleting a service	service	deleteService
Releasing a configuration	config	publishConfig
Deleting a Configuration	config	deleteConfig
Creating a namespace	namespace	createNamespace
Modifying a namespace	namespace	modifyNamespace
Deleting a namespace	namespace	deleteNamespace

**Table 7-2** ServiceComb engine operations that can be recorded by CTS

Operation	Resource Type	Event Name
Creating an engine	engine	createEngine
Deleting an engine	engine	deleteEngine
Upgrading or modifying an engine	engine	upgradeOrModifyEngine
Creating an engine backup task	engine	createEngine_backup
Deleting an engine backup task	engine	deleteEngine_backup
Creating an engine restoration task	engine	createEngine_recovery
Creating an engine backup policy	engine	createEngine_backup_strategy
Deleting an engine backup policy	engine	deleteEngine_backup_strategy
Updating an engine backup policy	engine	updateEngine_backup_strategy
Updating a dark launch rule	engine	ModifyDarklaunch
Deleting a dark launch rule	engine	DeleteDarklaunch
Modifying a configuration item	engine	ModifyConfig
Creating a configuration item	engine	CreateConfig
Deleting a configuration item	engine	DeleteConfig
Updating a governance rule	engine	ModifyGovern_policy
Updating a microservice	engine	modifyMicroservice
Creating a microservice	engine	createMicroservice
Deleting a microservice	engine	deleteMicroservice
Creating a microservice tag	engine	createMicroserviceTag
Updating a microservice tag	engine	updateMicroserviceTag

Operation	Resource Type	Event Name
Deleting a microservice tag	engine	deleteMicroserviceTag
Creating a microservice rule	engine	createMicroserviceRule
Updating a microservice rule	engine	updateMicroserviceRule
Deleting a microservice rule	engine	deleteMicroserviceRule
Creating a microservice schema	engine	createMicroserviceSchema
Updating a microservice schema	engine	updateMicroserviceSchema
Deleting a microservice schema	engine	deleteMicroserviceSchema
Updating microservice dependencies	engine	updateMicroserviceDependency
Updating microservice attributes	engine	updateMicroserviceProperty
Updating a microservice	engine	updateMicroservice
Updating monitoring thresholds	engine	updateThreshold
Updating a custom rule	engine	updateItem_meta
Deleting a custom rule	engine	DeleteItem_meta
Clearing configuration items	engine	executeConfig_cleanup
Updating status of a microservice instance	engine	updateInstanceStatus
Updating attributes of a microservice instance	engine	updateInstanceProperty
Creating a microservice instance	engine	createInstance
Deleting a microservice instance	engine	deleteInstance

## 7.2 Viewing CTS Traces in the Trace List

### Scenarios

Cloud Trace Service (CTS) records operations performed on cloud service resources. A record contains information such as the user who performed the operation, IP address, operation content, and returned response message. These records facilitate security auditing, issue tracking, and resource locating. They also help you plan and use resources, and identify high-risk or non-compliant operations.


### What Is a Trace?

A trace is an operation log for a cloud service resource, tracked and stored by CTS. Traces record operations such as adding, modifying, or deleting cloud service resources. You can view them to identify who performed operations and when for detailed tracking.

### Constraints

- You can only query operation records of the last seven days on the CTS console. They are automatically deleted upon expiration and cannot be manually deleted. To store them for longer than seven days, configure transfer to Object Storage Service (OBS) or Log Tank Service (LTS) so that you can view them in the OBS buckets or LTS log streams.
- After creating, modifying, or deleting a cloud service resource, you can query management traces on the CTS console 1 minute later and query data traces 5 minutes later.
- Data traces are not displayed in the trace list of the new version. To view them, you need to go to the old version.




### Viewing Real-Time Traces in the Trace List of the New Edition

- Step 1** Log in to the management console, click  in the upper left corner, and choose **Management & Deployment > Cloud Trace Service**.
- Step 2** In the navigation pane, choose **Trace List**.
- Step 3** In the time range drop-down list above the trace list, select a desired query time range: **Last 1 hour**, **Last 1 day**, or **Last 1 week**. You can also select **Custom** to specify a custom time range within the last seven days.
- Step 4** The search box above the trace list supports advanced queries. Combine one or more filters to refine your search.


**Table 7-3** Trace filtering parameters

Parameter	Description
Trace Name	<p>Name of a trace.</p> <p>The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.</p> <p>For details about the operations that can be audited for each cloud service, see section "Supported Services and Operations" in the <i>Cloud Trace Service User Guide</i>.</p> <p>Example: <b>updateAlarm</b></p>
Trace Source	<p>Cloud service name abbreviation.</p> <p>The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.</p> <p>Example: <b>IAM</b></p>
Resource Name	<p>Name of a cloud resource involved in a trace.</p> <p>The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.</p> <p>If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty.</p> <p>Example: <b>ecs-name</b></p>
Resource ID	<p>ID of a cloud resource involved in a trace.</p> <p>The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.</p> <p>Leave this field empty if the resource has no resource ID or if resource creation failed.</p> <p>Example: <i>{VM ID}</i></p>
Trace ID	<p>Value of the <b>trace_id</b> parameter for a trace reported to CTS.</p> <p>The entered value requires an exact match. Fuzzy matching is not supported.</p> <p>Example: <b>01d18a1b-56ee-11f0-ac81-*****1e229</b></p>
Resource Type	<p>Type of a resource involved in a trace.</p> <p>The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported.</p> <p>For details about the resource types of each cloud service, see section "Supported Services and Operations" in the <i>Cloud Trace Service User Guide</i>.</p> <p>Example: <b>user</b></p>

Parameter	Description
Operator	User who triggers a trace. Select one or more operators from the drop-down list. If the value of <b>trace_type</b> in a trace is <b>SystemAction</b> , the operation is triggered by the service and the trace's operator may be empty.
Trace Status	Select one of the following options from the drop-down list: <ul style="list-style-type: none"> <li>● <b>normal</b>: The operation succeeded.</li> <li>● <b>warning</b>: The operation failed.</li> <li>● <b>incident</b>: The operation caused a fault that is more serious than a normal failure, for example, causing other faults.</li> </ul>

- Step 5** On the **Trace List** page, you can also export and refresh the trace list, and customize columns to display.
- Enter any keyword in the search box and press **Enter** to filter desired traces.
  - Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5,000 records.
  - Click  to view the latest information about traces.
  - Click  to customize the information to be displayed in the trace list. If **Auto wrapping** is enabled (  ), excess text will move down to the next line; otherwise, the text will be truncated. By default, this function is disabled.
- Step 6** (Optional) On the **Trace List** page of the new edition, click **Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.
- End

## Viewing Traces in the Trace List of the Old Edition


- Step 1** Log in to the management console, click  in the upper left corner, and choose **Management & Deployment > Cloud Trace Service**.
- Step 2** In the navigation pane, choose **Trace List**.
- Step 3** Each time you log in to the CTS console, the new edition is displayed by default. Click **Old Edition** in the upper right corner to switch to the trace list of the old edition.
- Step 4** In the upper right corner of the page, set a desired query time range: **Last 1 hour**, **Last 1 day**, or **Last 1 week**. You can also click **Customize** to specify a custom time range within the last seven days.
- Step 5** Set filters to search for your desired traces.


**Table 7-4** Trace filtering parameters

Parameter	Description
Trace Type	<p>Select <b>Management</b> or <b>Data</b>.</p> <ul style="list-style-type: none"> <li>• Management traces record operations performed by users on cloud service resources, including creation, modification, and deletion.</li> <li>• Data traces are reported by OBS and record operations performed on data in OBS buckets, including uploads and downloads.</li> </ul>
Trace Source	Select the name of the cloud service that triggers a trace from the drop-down list.
Resource type	<p>Select the type of the resource involved in a trace from the drop-down list.</p> <p>For details about the resource types of each cloud service, see section "Supported Services and Operations" in the <i>Cloud Trace Service User Guide</i>.</p>
Search By	<p>Select one of the following options:</p> <ul style="list-style-type: none"> <li>• <b>Resource ID:</b> ID of the cloud resource involved in a trace. Leave this field empty if the resource has no resource ID or if resource creation failed.</li> <li>• <b>Trace name:</b> name of a trace. For details about the operations that can be audited for each cloud service, see section "Supported Services and Operations" in the <i>Cloud Trace Service User Guide</i>.</li> <li>• <b>Resource name:</b> name of the cloud resource involved in a trace. If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty.</li> </ul>
Operator	<p>User who triggers a trace.</p> <p>Select one or more operators from the drop-down list.</p> <p>If the value of <b>trace_type</b> in a trace is <b>SystemAction</b>, the operation is triggered by the service and the trace's operator may be empty.</p>
Trace Status	<p>Select one of the following options:</p> <ul style="list-style-type: none"> <li>• <b>Normal:</b> The operation succeeded.</li> <li>• <b>Warning:</b> The operation failed.</li> <li>• <b>Incident:</b> The operation caused a fault that is more serious than a normal failure, for example, causing other faults.</li> </ul>

**Step 6** Click **Query**.

**Step 7** On the **Trace List** page, you can also export and refresh the trace list.

- Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5,000 records.
- Click  to view the latest information about traces.

**Step 8** Click  on the left of a trace to expand its details.

**Step 9** Click **View Trace** in the **Operation** column. The trace details are displayed.

**Step 10** (Optional) On the **Trace List** page of the old edition, click **New Edition** in the upper right corner to switch to the **Trace List** page of the new edition.

----End

## Helpful Links

- For details about the key fields in the trace structure, see section "Trace References" > "Trace Structure" in the *Cloud Trace Service User Guide* and section "Trace References" > "Example Traces" in the *Cloud Trace Service User Guide*.

# 8 FAQs

---

## 8.1 Precautions When Using CSE

### 8.1.1 Why Can't I See Information About Cloud Services?

#### Symptom

When viewing the ServiceComb engine and Nacos engine, I can't see information about cloud services such as VPC and ELB.

#### Possible Cause

You do not have the permission to view the enterprise project the cloud services (such as VPC and ELB) belong to. Contact the tenant administrator to assign the permission.

### 8.1.2 What Should I Do If an Agency Creation Failed?

#### Symptom

After I log in to the CSE console, a message is displayed prompting me to enable and authorize the service. After I click **Authorize**, another message is displayed indicating that the authorization failed.

#### Possible Cause

You have no permission to create an agency, or your agency quota is insufficient.

#### Solution

- If the system displays a message indicating insufficient permissions when you log in to the IAM console, you have no permission to create an agency. In this case, see "How Can I Obtain Permissions to Create an Agency?" in *Identity and Access Management User Guide*.

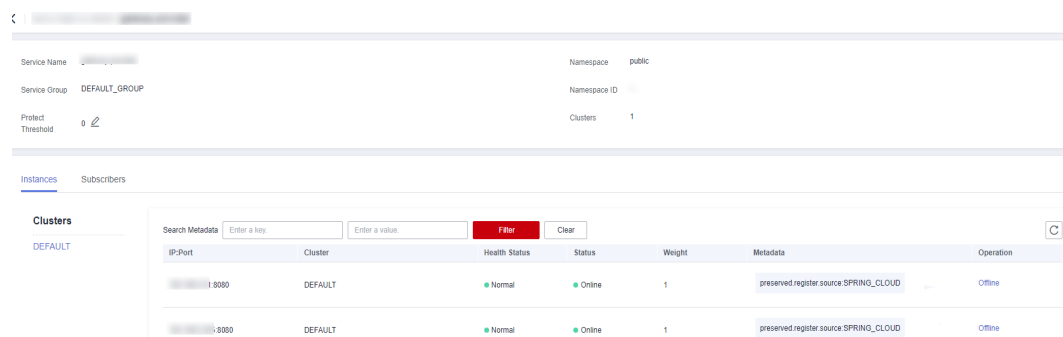
- If you have the permission to create an agency, your agency quota is insufficient. Log in to the IAM console and choose **Agencies** to check the agency quota and either adjust it or delete agencies that are no longer in use.

## 8.2 Nacos Engines

### 8.2.1 Why Do I Get an Error When I Request gRPC?

#### Symptom

When the service starts, two instances with ports 8080 and 9090 are registered as HTTP and gRPC ports. However, since only the instance with port 8080 is displayed in the service discovery list, it returns an error when gRPC is requested.



#### Possible Cause

When a client installed with Go SDK connects to Nacos engines, two instances are registered with a microservice at the same time. This dual registry is currently not available for Nacos engines.

#### Solution

Do not register two instances with the same microservice at the same time.

## 8.3 ServiceComb Engines

### 8.3.1 How Can I Handle a Certificate Loading Error?

#### Symptom

```

019/02/21 09:04:16 read ca cert file failed
019/02/21 09:04:16 Init chassis fail: read ca cert file failed
019/02/21 09:04:16 chassis init failed.
019/02/21 09:04:16 SetUp:RoomService init fail: read ca cert file failed

```

If the preceding error is displayed, the verifyPeer value is set to **true** but no certificate is configured. By default, the verifyPeer value in Go-chassis is **false** indicating that no certificate is configured and is not changed automatically.

## Solution

- If a certificate is not necessary but a certificate loading error occurs, the verifyPeer value changes during development or continuous integration. In this case, check the code from committing to deployment, and change value **true** to its default value **false**.
- If a certificate is necessary, configure it.

### 8.3.2 What If the Header Name Is Invalid?

#### Symptom

```
{
  "level": "[0:31mERROR]",
  "time": "2018-12-19 10:28:43.145 +08:00",
  "file": "handler/transport_handler.go:46",
  "msg": "Call got Error, err [Post http://172.16.0.140:31007/v1/domains: net/http: invalid header field name \"\\\""]"
}
{
  "level": "[0:31mERROR]",
  "time": "2018-12-19 10:28:43.145 +08:00",
  "file": "handler/transport_handler.go:46",
  "msg": "Call got Error, err [Post http://172.16.0.140:31007/v1/domains: net/http: invalid header field name \"\\\""]"
}
{
  "level": "[0:31mERROR]",
  "time": "2018-12-19 10:28:43.146 +08:00",
  "file": "handler/transport_handler.go:46",
  "msg": "Call got Error, err [Post http://172.16.0.140:31007/v1/domains: net/http: invalid header field name \"\\\""]"
}
{
  "level": "[0:31mERROR]",
  "time": "2018-12-19 10:28:43.146 +08:00",
  "file": "handler/transport_handler.go:46",
  "msg": "Call got Error, err [Post http://172.16.0.140:31007/v1/domains: net/http: invalid header field name \"\\\""]"
}
```

This problem is irrelevant to Go-chassis.

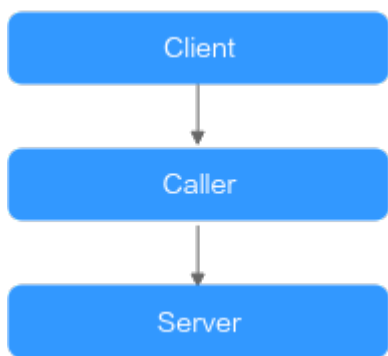
#### Solution

Check the service code, particularly the customized Go-chassis handler. Specifically, check whether there is an empty header or a non-standard header name.

### 8.3.3 What Is the Performance Loss of Mesher?

The service mesh technology uses network traffic hijacking to manage inter-service traffic. In addition to the performance loss in internal logic processing of Mesher, service mesh also causes extra conversion between the user mode and kernel mode (the CPU consumes extra resources). Compared with the latter, the former has little impact on performance. Therefore, the performance loss depends on the size of the payload transmitted over the network. For example, the size of the header and body affects the HTTP transmission speed. Latency of an end-to-end calling of Mesher is 1 ms. For example, if Mesher performance loss is added to the latency in an actual service calling, the total latency is 4 ms longer. However, such a latency is acceptable to the user.

The following table lists the performance test results before and after Mesher is used. The payload is small, that is, the character string **helloworld**. However, certain code is added to increase the computing time of the server to simulate the service code execution time.



**Table 8-1** Test results

Metric	Before	After
TPS	1749	1496
Latency	2.8 ms	3.34 ms
CPU	50%	100%
Concurrency	5	5

The preceding results show that the performance loss of Mesher is low. The performance bottleneck lies in the service code. If the payload content is increased, the performance deteriorates further.

It is recommended that this technology be used to call services during initial technical selection and POC to test the real performance loss.

### 8.3.4 Why Is "Version validate failed" Displayed When I Attempt to Connect to the Service Center?

#### Abnormal Message

```
{"errorCode":"400001","errorMessage":"Invalid parameter(s)","detail":{"Version validate failed, rule: {Length: 64,Length: ^[a-zA-Z0-9_\-.]*$}}"}
```

#### Possible Cause

A new SDK is used, but an earlier SDK of the service center is connected.

#### Fault Locating

Check the version of the service center. You can download the latest service center from the official website of the cloud platform or ServiceComb.

### 8.3.5 Why Is "Not enough quota" Displayed When I Attempt to Connect to the Service Center?

#### Abnormal Message

```
{"errorCode":"400100","errorMessage":"Not enough quota","detail":{"no quota to create instance, ..."}}
```

#### Possible Cause

There is insufficient quota to add a service instance.

#### Fault Locating

Log in to CSE and view the instance quota. If the quota is sufficient, check the service center address and region information configured in the code.

## 8.3.6 What Is Service Name Duplication Check?

### Question

What Is Service Name Duplication Check?

### Answer

Microservice names, applications, versions, and environments are checked.

A primary key uniquely identifies a microservice.

Ensure that each primary key is unique.

## 8.3.7 Why Do I Have to Define Service Contracts?

The enterprise-level systems are in large scale use and involve many microservice components. Therefore, unified API management is a key requirement of enterprises. CSE uses contract management to meet this requirement.

For management: Through contract management, API definition files that comply with API description standards for microservices are defined. In this way, API development of multiple development teams can be standardized and coordinated. This reduces communication costs and facilitating management.

For development: During microservice development, different teams or even different independent software vendors (ISVs) can develop the same application or system based on the unified API definition file. This facilitates consistency maintenance for the overall system. For example, modules in a monolithic application are called using code and any API incompatibility can be resolved economically during early compilation. When microservices are decoupled, services are remotely called. Therefore, API inconsistency cannot be found during early compilation, resulting in high bug fixing costs. Service contracts ensure that the architect will strictly review changes and reverse-generate code for API compatibility.

In addition, for small-scale systems that do not have high requirements on unified management, API definition files can be automatically generated through API code.

## 8.3.8 Why Are Microservice Development Framework and Netty Versions Unmatched?

### Symptom

During development of a microservice application, the following error is displayed:

```
"Caused by: java.lang.NoSuchMethodError:
io.netty.handler.codec.http.websocketx.WebSocketClientHandshakerFactory.newHandshaker(Ljava/net/
URI;Lio/netty/handler/codec/http/websocketx/WebSocketVersion;Ljava/lang/String;ZLio/netty/handler/codec/
http/HttpHeaders;IZZ)Lio/netty/handler/codec/http/websocketx/WebSocketClientHandshaker;"
```

### Possible Cause

Third-party software introduced a mismatched version dependency.

## Solution

Run the **mvn dependency:tree** command in the development environment to view the dependency tree and check whether the microservice development framework and Netty versions are matched.

For ServiceComb 2.0.1 development framework, the matched Netty version is 4.1.45.Final.

## 8.3.9 Why Is "Duplicate cluster name" Displayed?

### Abnormal Message

You receive the following message when creating an engine:

```
{"error_code":"SVCSTG.00500500","error_message":{"kind":"Status":"Failure"...}message":"duplicate cluster name","reason":"Conflict"}}
```

### Possible Cause

The CCE service on which the microservice engine depends is abnormal, causing engine creation, deletion, and configuration to be unavailable.

### Solution

Contact O&M personnel.

## 8.3.10 Error Message "the subnet could not be found" Is Displayed When the Access Address Fails to Be Processed During Engine Creation

### Symptom

During engine creation, the access address fails to be processed, and the following error message is displayed:

```
{"error_code":"SVCSTG.00500404","error_message":{"code":"VPC.0202","message":"Query resource by id xxx fail.the subnet could not be found."}}
```

### Possible Cause

CSE is not authorized in the user's project.

### Solution

- When you use CSE instances provisioned from ServiceStage and want to create new instances in CSE, you need to grant permissions to CSE. For details, see "Creating a User and Granting Permissions" in the *Cloud Service Engine User Guide*.
- Since it depends on VPC, CSE needs permissions. Create a cloud service agency **cse\_admin\_trust** by referring to "Creating an Agency" in *Identity and Access Management User Guide*.

## 8.3.11 What Should I Do If SpringCloud Applications Fail to Connect to the Configuration Center of ServiceComb Engine 2.x?

### Symptom

The center is configured in the code, but the configuration items cannot be obtained.

```
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1247)
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1236)
at com.huawei.financialsolutionassetcenter.console.ConsoleApplication.main(ConsoleApplication.java:29)
Caused by: java.lang.IllegalArgumentException: Create breakpoint : Could not resolve placeholder 'cse.example.key1' in value "${cse.example.key1}"
at org.springframework.util.PropertyPlaceholderHelper.parseStringValue(PropertyPlaceholderHelper.java:186)
```

### Possible Cause

The type and address of the configuration center in the configuration file are incorrect.

### Fault Location

1. The engine uses the 2.x key, but the configuration center uses the 1.x key.
2. The environment is specified in the configuration file, but the environment key obtained from the code is incorrect.

### Solution

1. Set **spring.cloud.servicecomb.config.serverType** to **kie** in the configuration file to change the configuration center type to kie.
2. Modify **server.env** to change the environment key in the configuration file.

## 8.3.12 Why Could My the Global Configuration Not Be Modified?

### Symptom

After the global configuration is modified, the configuration obtained by the service is not changed, and the keyword **changed** is not recorded in the log.

```
2021-11-15 10:13:33.710 INFO 6212 --- [ntloop-thread-0] o.a.s.ConfigCenterConfigurationSourceImpl : Config value cache changed; action:set; item:[servicecomb.credentials.akskEn
2021-11-15 10:13:33.710 INFO 6212 --- [ntloop-thread-0] o.a.s.config.client.ParseConfigUtils : Updating remote config is done. revision has changed from default9602643 to
2021-11-15 10:13:36.538 INFO 6212 --- [ctator-poller-0] o.a.s.a.c.publish.DefaultLogPublisher :
vertx:
instances:
  name      eventLoopContext-created
  registry  0
  transport 0
  config-center 0
```

### Possible Cause

The dependency of config-cc was not configured.

## Fault Location

1. Check for the address of the configuration center in the **application.yaml** file.
2. Check for the config-cc dependency in the **pom.xml** file.

## Solution

Add the config-cc dependency to the **pom.xml** file.

```
<dependency>  
<groupId>org.apache.servicecomb</groupId>  
<artifactId>config-cc</artifactId>  
</dependency>
```

## 8.3.13 Obtain Configurations Failed

### Symptom

After a microservice is connected to the corresponding microservice development framework (such as spring-cloud-huawei and java-chassis), it fails to obtain configuration items from the ServiceComb engine by calling the configuration query API through SDK.

### Possible Cause

If the connection between a microservice and the registration center jitters due to network and CPU problems, the request may be abnormal.

### Solution

The microservice framework has the self-healing capability. If the configuration fails to be obtained, the retry mechanism takes effect for pulling configurations. Service exceptions do not occur. Check whether configuration items can be obtained next time. If not, contact customer service.