Cloud Service Engine

Service Overview

Issue 01

Date 2024-04-11

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 CSE Infographics

2 What Is CSE?

Cloud Service Engine (CSE) is a cloud middleware used for microservice applications. It supports ServiceComb engines contributed to Apache and 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.

- Open-source base

  The open-source Apache microservice core framework ServiceComb supports Spring Cloud and commercial Service Mesh.
- Reliability and stability

  Using Huawei's cloud-native technology, CSE provides cloud core services to hundreds of millions of Huawei devices.
- Professional services

  CSE is developed in collaboration with 100+ industry partners with experience in microservice consulting services.
- Multiple programming languages

  Microservice solutions are compiled in multiple programing languages, such as Java, Go, .NET, and Node.js.

Nacos Engine

CSE Nacos is a microservice registry, discovery, and configuration management platform developed based on open-source Nacos 2.x. It supports multiple development languages and frameworks, and provides DNS-based service discovery.

NOTE

Nacos engines are supported only in CN-Hong Kong, AP-Singapore, ME-Riyadh, and LA-Mexico City2.

Key features of Nacos:

Table 2-1 Features supported by Nacos

Function	Sub-feature
Instance management	Creating a gateway
	Viewing instance specifications
	Viewing instance list
	Viewing instance details
	Deleting an instance
	Changing the billing mode from pay-per-use to yearly/monthly
	Enterprise project
Specification change	Expansion/Change
Connections	Using the IP address and domain name to access intranet
Namespace management	Viewing namespace list
	Viewing namespace details
	Creating a namespace
	Modifying a namespace
	Deleting a namespace
	Maximum number of namespaces **NOTE** Up to 50 namespaces can be created.
Service management	Filtering namespaces
	Searching for a service
	Filtering empty services
	Viewing service list
	Creating a service
	Viewing service providers
	Viewing service subscribers
	Distinguishing instances by cluster
	Filtering providers by metadata
	Getting online/offline of a service node
	Editing the weight of a service node
Configuration management	Creating a configuration

Function	Sub-feature
	Import configurations
	Editing a configuration
	Deleting configurations
	Configuring dark launch
	Maximum number of configuration items that can be configured in a namespace
	Searching for a configuration
	Viewing configuration list
	Viewing configuration details
	Viewing historical versions
	Roll back a version
	Comparing configuration contents
	Configuring interception query.

ServiceComb Engine

ServiceComb engine uses Apache ServiceComb Service Center, which is a RESTful-style, high-availability, and stateless service registry and discovery center and provides microservice discovery and management. Service providers can register their instance information with the registry and discovery center for consumers to discover and use. ServiceComb engine is seamlessly compatible with open-source ecosystems such as Spring Cloud and ServiceComb. 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**

NOTE

ServiceComb engines are supported only in CN-Hong Kong, ME-Riyadh, TR-Istanbul, AP-Singapore, AP-Jakarta, and LA-Mexico City2.

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, and these specifications cannot be changed once engines are created. 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 2-2** lists features supported in CSE 1.0 and CSE 2.0.

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

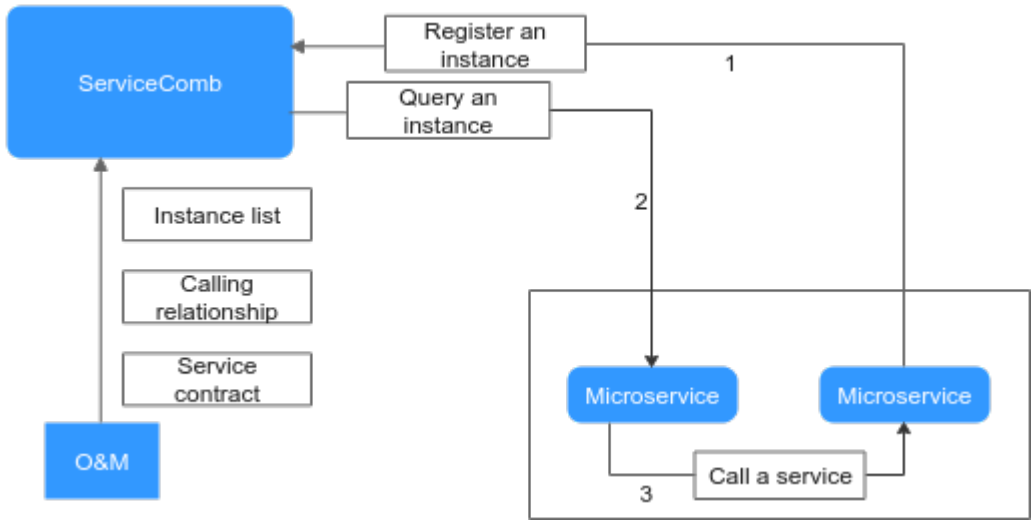
Feature	Sub-feature		2.x	1.x	Remarks
Engine management	Security	Security authentication	√	√	-
	Reliability	3-AZ high reliability	√	√	-
Microservice management	Basic capability	Registry and discovery	√	√	-
		Multi-frame access	√	√	Supports Spring Cloud and ServiceComb Java Chassis.
		Automatic clear of 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.
	Advanced features	History version management	√	x	-
		Version comparison	√	x	-

Feature	Sub-feature		2.x	1.x	Remarks
		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.	√	√	-
	Governance policy: repository isolation	The server controls the request concurrency capability based on the semaphore.	√	x	-

Feature	Sub-feature		2.x	1.x	Remarks
Development tool	Local lightweight engine	One-click local startup, facilitating offline microservice development	√	√	-

3 Application Scenarios

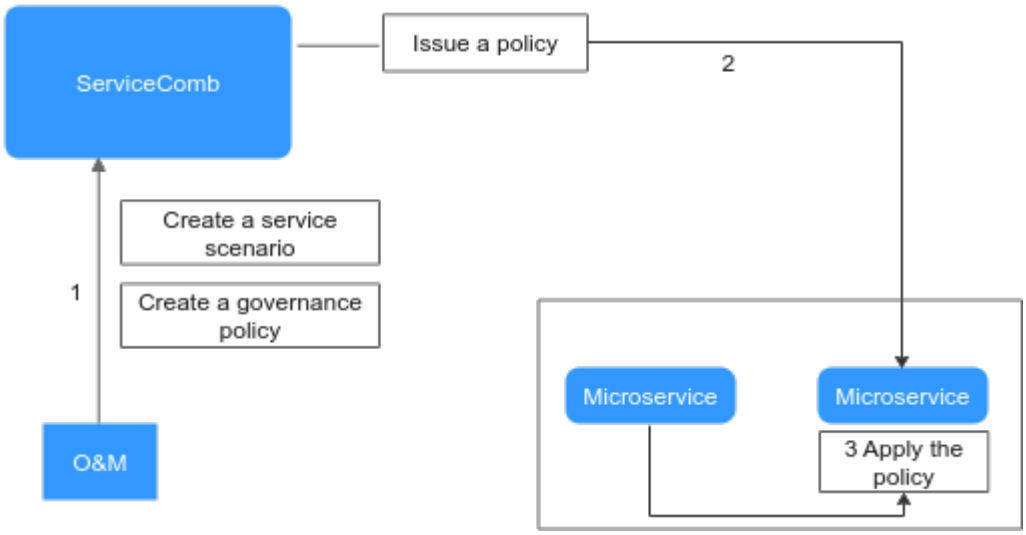
Microservice Registry and Discovery



When a microservice starts, the instance information is registered with CSE, including the basic instance information, such as the application name, microservice name, version number, service contract, and instance address. When a microservice needs to call the APIs of other microservices, it queries instance information from CSE and caches the information locally. The cache is updated through mechanisms such as event notification and scheduled query. The locally cached address information is used to implement point-to-point calling between microservices. When a microservice has multiple instances, different load balancing policies can be configured, including polling, weight, and dark launch.

For O&M, you can view the instance list, microservice calling relationship, and service contract through CSE to help customers understand the application system composition and running status.

Microservice Governance

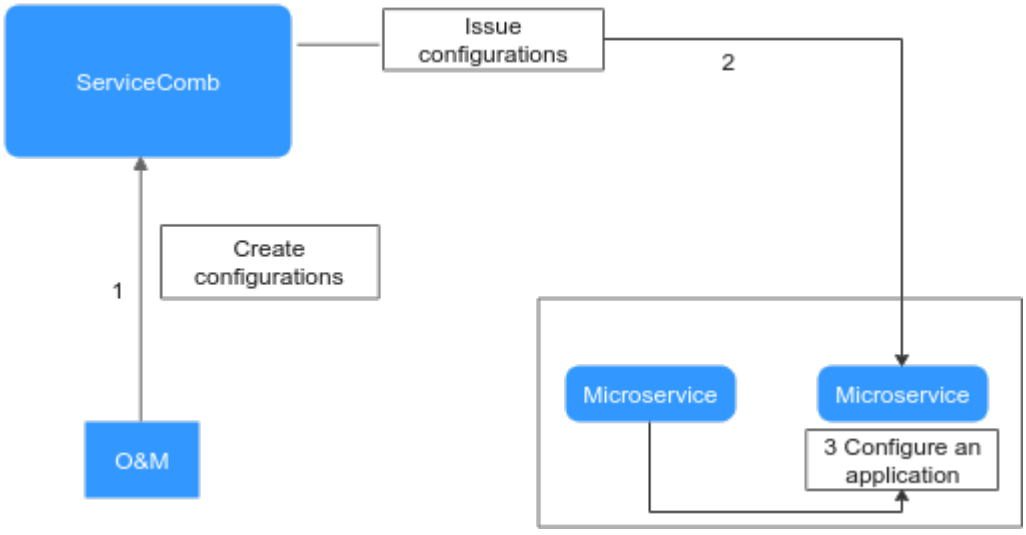


Service governance can be enabled using either of the following ways:

1. Configured in microservice development. In this way, the policy is configured for all service scenarios, for example, the load balancing policy.
2. Configured in microservice running. In this way, the policy is configured for scenarios where services are dynamically changed, for example, the rate limiting policy.

These two ways have the same internal implementation mechanism. The configuration management system delivers governance policies and the runtime SDK executes the governance policies. The runtime governance SDK is included in the microservice development framework and extension package selected by the user and is compiled and integrated with microservices. The CSE console provides common governance policy management to help users adjust governance policies based on services.

Configuration Management



Configuration management centrally manages microservice configurations. Configurations are delivered to specific microservices based on configuration item attributes, such as scopes and labels.

Configuration management provides a series of user-friendly functions, such as historical versions view, configurations rollback, configurations import and export, and flexible scope management, meeting users' requirements for managing complex environments and a massive number of microservices.

4 Glossary

General

Concept	Description
Microservice	Microservice is a service concept, that is, a process that provides a service. Each service has its own service functions. APIs that are not restricted by languages are open to other systems (HTTP frequently used). Multiple microservices form an application.
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	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.

Glossary (Nacos Engine)

Concept	Description
Namespace	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.
Configuration set	A set of configuration items is called a configuration set. Generally, a configuration file is a configuration set that contains all system configurations.
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.

Concept	Description
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	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. **NOTE** • By default, service contract is used in Java chassis. • By default, service contract is not used in Spring Cloud. If it is used, the following dependencies need to be introduced: <dependency> <groupId>com.huaweicloud</groupId> <artifactId>spring-cloud-starter-huawei-swagger</artifactId> </dependency>
Application	A software system that completes a complete service scenario. An application consists of multiple microservices, which can discover and call each other.

Concept	Description
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.

5 Version Support by ServiceComb Engines

This section describes the versions supported by ServiceComb engines.

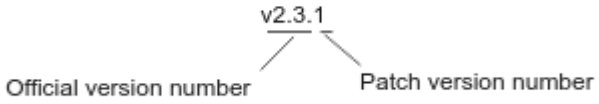
Version Description

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.

You can submit a **service ticket** to evaluate risks before the upgrade.

- 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.

6 Security

6.1 Shared Responsibilities

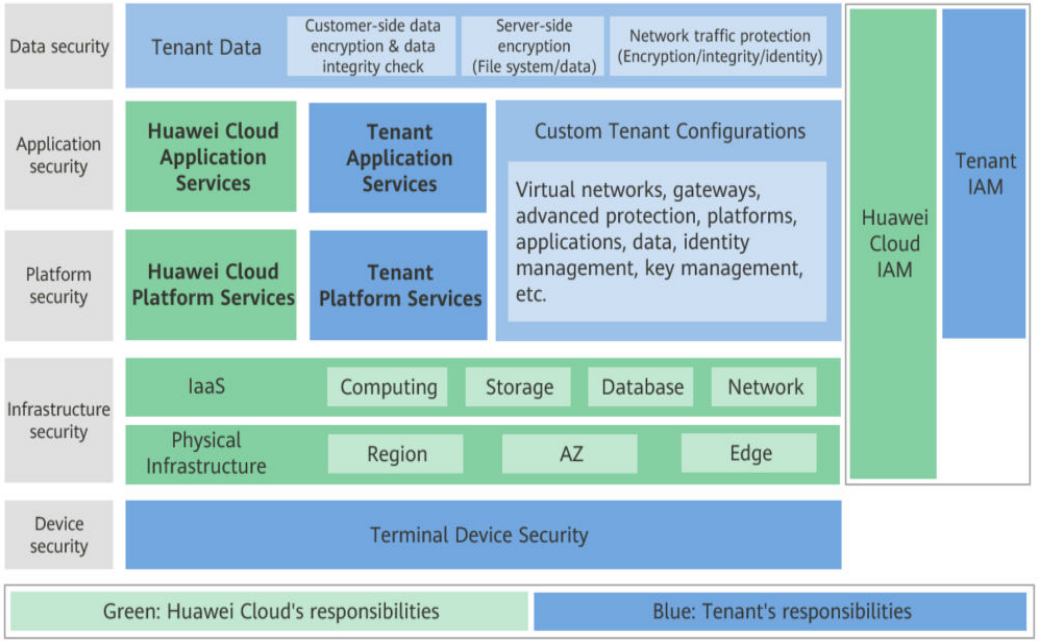
Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To cope with emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system for different regions and industries based on Huawei's unique software and hardware advantages, laws, regulations, industry standards, and security ecosystem.

Figure 6-1 illustrates the responsibilities shared by Huawei Cloud and IAM users.

- **Huawei Cloud**: Ensure the security of cloud services and provide secure clouds. Huawei Cloud's security responsibilities include ensuring the security of our IaaS, PaaS, and SaaS services, as well as the physical environments of the Huawei Cloud data centers where our IaaS, PaaS, and SaaS services operate. Huawei Cloud is responsible for not only the security functions and performance of our infrastructure, cloud services, and technologies, but also for the overall cloud O&M security and, in the broader sense, the security compliance of our infrastructure and services.
- **Tenant**: Use the cloud securely. Tenants of Huawei Cloud are responsible for the secure and effective management of the tenant-customized configurations of cloud services including IaaS, PaaS, and SaaS. This includes but is not limited to virtual networks, the OS of virtual machine hosts and guests, virtual firewalls, API Gateway, advanced security services, all types of cloud services, tenant data, identity accounts, and key management.

Huawei Cloud Security White Paper elaborates on the ideas and measures for building Huawei Cloud security, including cloud security strategies, the shared responsibility model, compliance and privacy, security organizations and personnel, infrastructure security, tenant service and security, engineering security, O&M security, and ecosystem security.

Figure 6-1 Huawei Cloud shared security responsibility model



6.2 Identity Authentication and Access Control

Identity authentication

CSE can be authenticated by role-based access control (RBAC).

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. Users using this account have the permissions to access and perform operations on the microservice engine. For details, see **Security Authentication Overview**.

Configuring Access Control

If you need to assign different permissions to employees in your enterprise to access your CSE resources, Identity and Access Management (IAM) is a good choice for fine-grained permissions management.

Access Policy	Description	Documentation
IAM permissions	IAM permissions define which actions on your cloud resources are allowed and which actions are denied, to control access to your resources. After creating an IAM user, the administrator needs to add it to a user group and grant the permissions required by CSE to the user group. Then, all users in this group automatically inherit the granted permissions.	**Permissions**
Custom permissions	A microservice engine may be used by multiple users. Different users must have different microservice engine access and operation permissions based on their responsibilities and permissions. The exclusive microservice engine with security authentication enabled provides the RBAC-based system management through the microservice console. You can customize policies for roles based on service requirements.	**System Management Overview**

6.3 Data Protection

CSE uses multiple data protection measures to ensure the security and reliability of data stored. The following table describes the data protection measures.

Measure	Description	Documentation
HTTPS transmission	CSE uses HTTPS to ensure information transmission security.	**Making an API Request**

Measure	Description	Documentation
Cross-AZ engine	CSE supports cross-AZ deployment. To ensure reliability, you are advised to use the cross-AZ CSE engine.	**Creating a ServiceComb Engine**
Versioning	CSE can store multiple configuration versions, helping you view, manage, and quickly roll back configurations.	**Comparing Configuration Versions**
Configuration file encryption	CSE supports encrypted storage of configuration files to ensure sensitive data security.	**Configuration File Encryption Scheme**

6.4 Fault Recovery

Backup and Restoration

You can customize backup policies to automatically back up microservice engines periodically or manually back up microservice engines at a specified time point. For details, see **Configuring Backup and Restoration of an Exclusive ServiceComb Engine**.

Multi-AZ

An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. CSE supports cross-AZ deployment to provide AZ-level high availability. For details, see **Creating a ServiceComb Engine**.

6.5 Audit and Logs

Audit

Cloud Trace Service (CTS) records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, track resource changes, audit compliance, and locate faults. After you enable CTS and configure a tracker, CTS can record management and data traces of CSE for auditing. For details about how to enable and configure CTS, see **Enabling CTS**. For details about CSE management traces and data traces that can be traced by CTS, see **CSE 2.0 Operations That Can Be Recorded by CTS**.

Logs

Log in to the CSE console. In the **Operation** area, you can view CSE operation logs. For details, see **Viewing Microservice Engine Operation Logs**.

7 Specifications

Nacos engine instance specifications

You can select proper Nacos instance specifications as required.

Table 7-1 Nacos engine instance specifications

Instances	Capacity Unit	Consumer Connections
500	10	1,000
1,000	20	2,000
2,000	40	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 7-2**.

As shown in **Table 7-2**, ServiceComb engine instances with different microservice instances are rewarded with corresponding configuration items and the maximum number of microservice versions supported.

Table 7-2 ServiceComb Engine Instance Specifications

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

Microservice Instances	Configuration Items
2,000	12,000

8 Restrictions

Nacos engine version compatibility

Nacos Engine Version	Nacos Community Compatibility	Compatibility
2.1.0.x	2.1.0	100%

The Nacos engine is backward compatible with Nacos 2.1.0 and earlier versions. For example, if microservices use open-source Nacos 1.x.x, Nacos can also be used.

The configuration center is compatible with all versions from Nacos 1.0, and service discovery is compatible with all versions from Nacos 1.2. Therefore, you are advised to use Nacos 1.2.0 and later versions. gPRC connection is supported in Nacos 2.x.x.

In Nacos 2.1.0 and later versions, the dual-write capability is disabled by default. Therefore, Nacos 1.x cannot be smoothly upgraded to Nacos 2.1.0+. To use smooth upgrade, from example, directly upgrade Nacos 1.x to Nacos 2.1.0+, **nacos.core.support.upgrade.from.1x** should be set to **true** in the **application.properties** file.

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

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

Requirements for Microservice Development Framework of a

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 engines, you are advised to use the Java chassis framework.

Framework	Recommended Versions	Description
Spring Cloud Huawei	1.10.9-2021.0.x or later	Uses **Spring Cloud Huawei** for connection. • Spring Cloud version 2021.0.5 • Spring Boot 2.6.13 Version description of the Spring Cloud microservice development framework: **https://github.com/huaweicloud/spring-cloud-huawei/releases**
Java Chassis	2.7.10 or later	Uses the software package provided by the open-source project for connection without introducing third-party software packages. Version description of the Java chassis microservice development framework: **https://github.com/apache/servicecomb-java-chassis/releases**.

NOTICE

During system upgrade and reconstruction, third-party software conflict is the most common issue. Traditional software compatibility management policies do not adapt to software development for fast software iteration. In this case, see **Third-Party Software Version Management Policy** for version compatibility.

Function Comparison Between Spring Cloud Huawei, ServiceComb, and Sermant

Level-1 Feature	Level-2 Feature	serviccomb-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.
Configuration 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	√	×	×	Java chassis supports the following communication protocols for the consumer and provider: • Provider: JAX-RS, SpringMVC, and transparent RPC. • Consumer: transparent RPC, RestTemplate, and InvokerUtils.

	Expansion	• Allows users to process traffic by custom tracing. • Supports governance for expanded traffic.	• Supports native Spring Cloud expansion. • Supports governance for expanded traffic.	New functions are added based on plug-in development.	-

Quotas

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

- **Table 8-1** lists the maximum number of resources that can be created in engine instances.

Table 8-1 Resource quota limits of 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	-

Function	Resource	Quota	Modifiable	Precaution
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.

9 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 secure access to your resources.

With IAM, you can use your public cloud 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 public cloud account does not need individual IAM users for permissions management, you may skip over this chapter.

IAM is free of charge. You pay only for the resources in your account. For more information about IAM, see the **IAM Service Overview**.

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 9-1 lists all the system policies supported by CSE.

Table 9-1 CSE system permissions

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

If the listed permissions in Table 9-1 do not meet actual requirements, you can **Creating a Custom Policy for a Microservice Engine**.

For details about the service permissions required by CSE functions, see Table 9-2.

Table 9-2 Dependent service permissions

Dependent Service	Permission
VPC	VPC ReadOnlyAccess
ELB	ELB ReadOnlyAccess
AOM	AOM ReadOnlyAccess
TMS	TMS ReadOnlyAccess

Table 9-3 lists the common operations for each system-defined policy or role of CSE. Select policies or roles as needed.

Table 9-3 Common operations supported by each system policy

Operation	CSE ReadOnlyAccess	CSE FullAccess
Create a microservice engine	x	√
Maintain a microservice engine	x	√

Operation	CSE ReadOnlyAccess	CSE FullAccess
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 9-4** describes fine-grained permission dependencies of CSE.

Table 9-4 Fine-grained permission dependencies of CSE

Permission	Description	Permission Dependency	Application Scenario
cse:engine:list	List all microservice engines	None	• View ServiceComb engine list • View Nacos engine list

Permission	Description	Permission Dependency	Application Scenario
cse:engine:get	View engine information	cse:engine:list	• View ServiceComb engine details • View Nacos engine details
cse:engine:modify	Modify an engine	• cse:engine:list • cse:engine:get	• Enable/Disable public network access to ServiceComb engines • Enable/Manage security authentication for ServiceComb engines • Retry a ServiceComb engine task • Enable/Disable security authentication for Nacos engines • Manage a Nacos user and role • Manage relationship between namespaces and enterprise projects
cse:engine:upgrade	Upgrade an engine	• cse:engine:list • cse:engine:get	• Upgrade a ServiceComb engine • Upgrade a Nacos engine Upgrades include version upgrade and specification change.
cse:engine:delete	Delete an engine	• cse:engine:list • cse:engine:get • vpc:ports:get • vpc:ports:delete	• Delete a ServiceComb engine • Delete a Nacos engine

Permission	Description	Permission Dependency	Application Scenario
cse:engine:create	Create an engine	• cse:engine:get • cse:engine:list • ecs:cloudServerFlavors:get • vpc:vpcs:get • vpc:vpcs:list • vpc:subnets:get • vpc:ports:get • vpc:ports:create	• Create a ServiceComb engine • Back up a ServiceComb engine/Restore task creation • Create a Nacos engine
cse:config:modify	Modify ServiceComb engine configuration management	• cse:engine:list • cse:engine:get • cse:config:get	Modify global and governance configurations of a ServiceComb engine
cse:config:get	View ServiceComb engine configuration management	• cse:engine:list • cse:engine:get	View service configurations for a ServiceComb engine
cse:governance:modify	Modify the governance center of a ServiceComb engine	• cse:engine:list • cse:engine:get • cse:config:get • cse:config:modify • cse:registry:get • cse:registry:modify • cse:governance:get	Create and modify service governance of a ServiceComb engine
cse:governance:get	View the governance center of a ServiceComb engine	• cse:engine:list • cse:engine:get • cse:config:get • cse:registry:get	View service governance on a ServiceComb engine
cse:registry:modify	Modify service registry and management of a ServiceComb engine	• cse:engine:list • cse:engine:get • cse:registry:get	Modify a ServiceComb engine

Permission	Description	Permission Dependency	Application Scenario
cse:registry:get	View service registry and management of a ServiceComb engine	• cse:engine:list • cse:engine:get	View service catalog on a ServiceComb engine
cse:namespace:read	View namespace resources of a Nacos engine	cse:engine:get	View Nacos service list and configuration list
cse:namespace:write	Modify namespace resources of a Nacos engine	• cse:engine:get • cse:namespace:read	Modify Nacos service list and configuration list resources

References

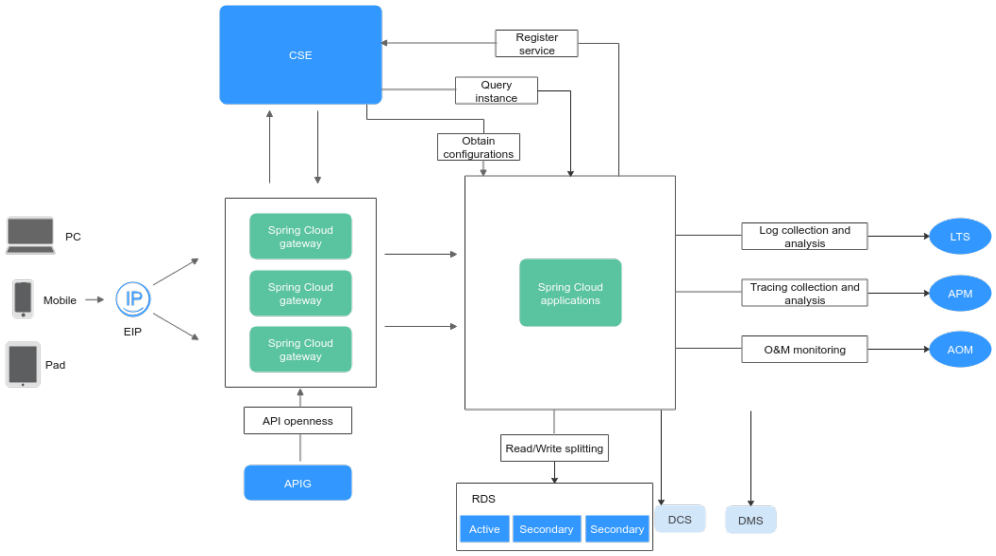
- **IAM Service Overview**
- **Creating a User and Granting Permissions**

10 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.

