Cloud Service Engine

Service Overview

Issue 01

Date 2025-08-29

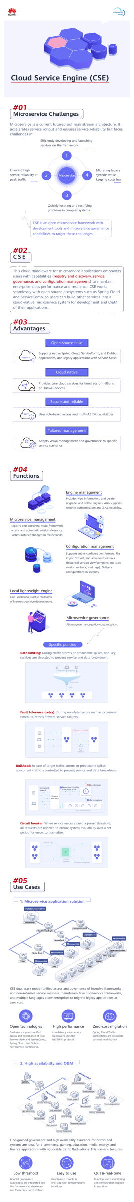
HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

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 Infographic



Cloud Service Engine (CSE)

#01 Microservice Challenges

Microservice is a current futureproof mainstream architecture. It accelerates service rollout and ensures service reliability but faces challenges in:

1 Efficiently developing and launching services on the framework

2 Ensuring high service reliability in peak traffic

3 Quickly locating and rectifying problems in complex systems

4 Migrating legacy systems while keeping costs low

CSE is an open microservice framework with development tools and microservice governance capabilities to target these challenges.

#02 CSE

This cloud middleware for microservice applications empowers users with capabilities (registry and discovery, service governance, and configuration management) to maintain enterprise-class performance and resilience. CSE works seamlessly with open-source ecosystems such as Spring Cloud and ServiceComb, so users can build other services into a cloud-native microservice system for development and O&M of their applications.

#03 Advantages

Open-source base

Supports native Spring Cloud, ServiceComb, and Dubbo applications, and legacy applications with Service Mesh.

Cloud native

Provides core cloud services for hundreds of millions of Huawei devices.

Secure and reliable

Uses role-based access and multi-AZ DR capabilities.

Tailored management

Adapts visual management and governance to specific service scenarios.

#04 Functions

Engine management

Includes view information, and create, upgrade, and delete engines. Also supports security authentication and 3-AZ reliability.

Microservice management

Registry and discovery, multi-framework access, and automatic version clearance. Pushes instance changes in milliseconds.

Configuration management

Supports many configuration formats, file import/export, and advanced features (historical version view/compare, one-click version rollback, and tags). Delivers configurations in seconds.

Local lightweight engine

One-click local startup facilitates offline microservice development.

Microservice governance

Allows governance policy customization.

Rate limiting: During traffic storms or predictable spikes, non-key services are throttled to prevent service and data breakdown.

Fault tolerance (retry): During non-fatal errors such as occasional timeouts, retries prevent service failures.

Bulkhead: In case of larger traffic storms or predictable spikes, concurrent traffic is controlled to prevent service and data breakdown.

Circuit breaker: When service errors exceed a preset threshold, all requests are rejected to ensure system availability over a set period for errors to normalize.

#05 Use Cases

1. Microservice application solution

CSE dual-stack mode (unified access and governance of intrusive frameworks and non-intrusive service meshes), mainstream Java microservice frameworks, and multiple languages allow enterprises to migrate legacy applications at zero cost.

Open technologies

Dual-stack supports unified access and governance of Istio Service Mesh and ServiceComb, Spring Cloud, and Dubbo microservice frameworks.

High performance

Low-latency microservice framework uses the REST/RPC protocol.

Zero-cost migration

Spring Cloud/Dubbo applications are accessible without modification.

2. High availability and O&M

Fine-grained governance and high availability assurance for distributed systems are ideal for e-commerce, gaming, education, media, energy, and finance applications with noticeable traffic fluctuations. This scenario features:

Low threshold

General governance capabilities are integrated into the framework so developers can focus on services instead.

Easy to use

Governance console is one-stop with comprehensive functions.

Quasi-real-time

Running status monitoring and configuration happen in real time.

2 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.

Features:

1. Compatible with mainstream technologies in the open-source community
   - Open-source frameworks such as Spring Cloud and Dubbo
   - Access to CSE without code modification
   - Mainstream open-source registry/configuration center
2. Unified service governance platform
   - Multi-framework, multi-technology stack, and multi-language applications
   - Interconnection, unified governance, and smooth evolution

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, 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 2-1** lists features supported in CSE 1.0 and CSE 2.0.

Table 2-1 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	√	√	-

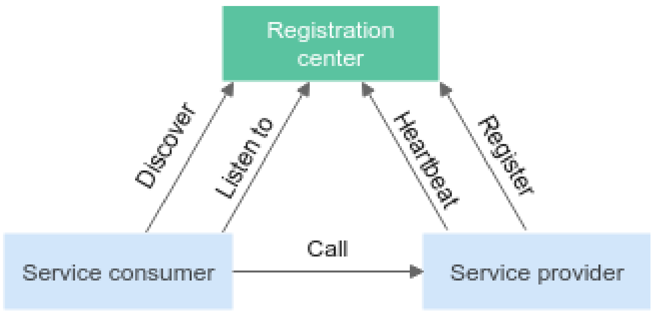
Feature	Sub-feature		2.x	1.x	Remarks
		Diversified configuration formats	√	Only text is supported.	2.x supports YAML, JSON, TEXT, Properties, INI and XML.
		Import and export	√	√	2.x supports configuration import of the same policy.
	Advanced features	History version management	√	x	-
		Version comparison	√	x	-
		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.	√	√	-

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

3 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.

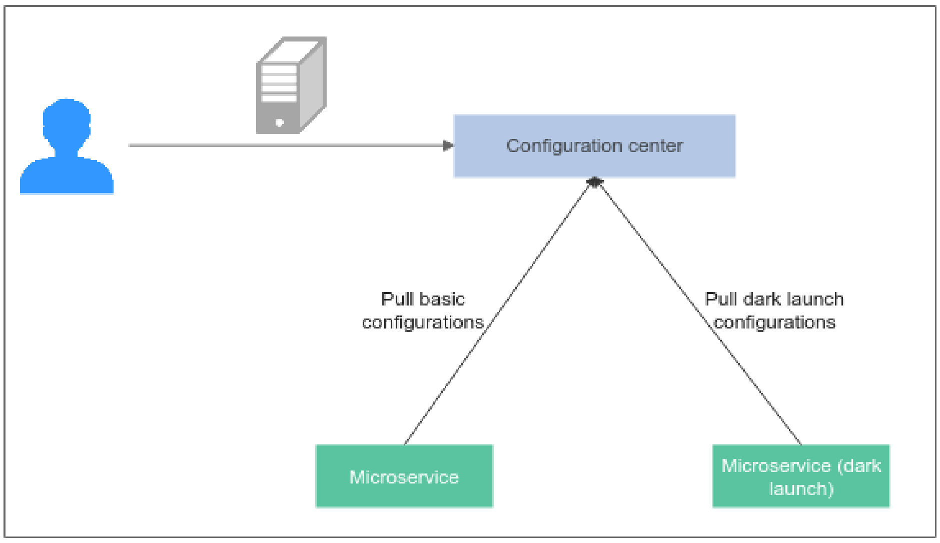
- **Service discovery**: Service consumers directly interact with the registration center to obtain service provider's instance information and implement load balancing locally. In the microservice architecture, service provider instances may change due to various reasons (such as scale-in, scale-out, 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.


Configuration center
Pull basic configurations
Pull dark launch configurations
Microservice
Microservice (dark launch)

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.

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.

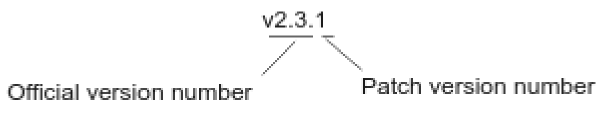
Version Description

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

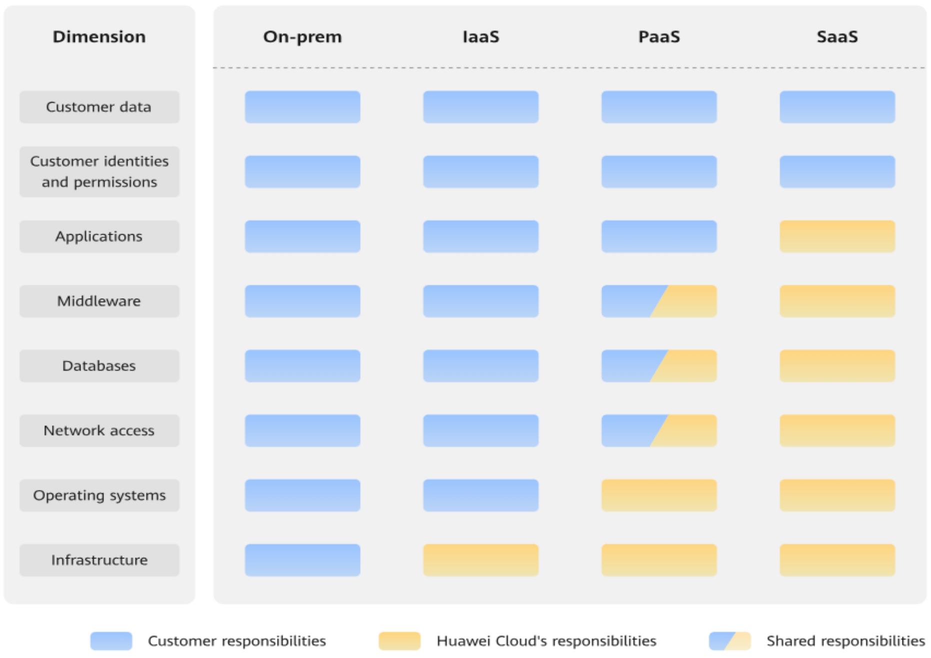
6.1 Shared Responsibilities

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.

Unlike traditional on-premises data centers, cloud computing separates operators from users. This approach not only enhances flexibility and control for users but also greatly reduces their operational workload. For this reason, cloud security cannot be fully ensured by one party. Cloud security requires joint efforts of Huawei Cloud and you, as shown in **Figure 6-1**.

- **Huawei Cloud**: Huawei Cloud is responsible for infrastructure security, including security and compliance, regardless of cloud service categories. The infrastructure consists of physical data centers, which house compute, storage, and network resources, virtualization platforms, and cloud services Huawei Cloud provides for you. In PaaS and SaaS scenarios, Huawei Cloud is responsible for security settings, vulnerability remediation, security controls, and detecting any intrusions into the network where your services or Huawei Cloud components are deployed.
- **Customer**: As our customer, your ownership of and control over your data assets will not be transferred under any cloud service category. Without your explicit authorization, Huawei Cloud will not use or monetize your data, but you are responsible for protecting your data and managing identities and access. This includes ensuring the legal compliance of your data on the cloud, using secure credentials (such as strong passwords and multi-factor authentication), and properly managing those credentials, as well as monitoring and managing content security, looking out for abnormal account behavior, and responding to it, when discovered, in a timely manner.

Figure 6-1 Huawei Cloud shared security responsibility model



Dimension	On-prem	IaaS	PaaS	SaaS
Customer data	🟦	🟦	🟦	🟦
Customer identities and permissions	🟦	🟦	🟦	🟦
Applications	🟦	🟦	🟦	🟨
Middleware	🟦	🟦	◩	🟨
Databases	🟦	🟦	◩	🟨
Network access	🟦	🟦	◩	🟨
Operating systems	🟦	🟦	🟨	🟨
Infrastructure	🟦	🟨	🟨	🟨

🟦 Customer responsibilities 🟨 Huawei Cloud's responsibilities ◩ Shared responsibilities

Cloud security responsibilities are determined by control, visibility, and availability. When you migrate services to the cloud, assets, such as devices, hardware, software, media, VMs, OSs, and data, are controlled by both you and Huawei Cloud. This means that your responsibilities depend on the cloud services you select. As shown in **Figure 6-1**, customers can select different cloud service types (such as IaaS, PaaS, and SaaS services) based on their service requirements. As control over components varies across different cloud service categories, the responsibilities are shared differently.

- In on-premises scenarios, customers have full control over assets such as hardware, software, and data, so tenants are responsible for the security of all components.
- In IaaS scenarios, customers have control over all components except the underlying infrastructure. So, customers are responsible for securing these components. This includes ensuring the legal compliance of the applications, maintaining development and design security, and managing vulnerability remediation, configuration security, and security controls for related components such as middleware, databases, and operating systems.
- In PaaS scenarios, customers are responsible for the applications they deploy, as well as the security settings and policies of the middleware, database, and network access under their control.
- In SaaS scenarios, customers have control over their content, accounts, and permissions. They need to protect their content, and properly configure and protect their accounts and permissions in compliance with laws and regulations.

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**

Access Policy	Description	Documentation
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**
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	Number of configuration files
500	10	1,000	1,000
1,000	20	2,000	2,000
2,000	40	4,000	4,000
3,000	60	6,000	6,000
5,000	100	10,000	10,000

NOTE

- To create a Nacos engine with more than 2,000 microservice instances, submit a **service ticket**.
- 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
2,000	12,000

8 Restrictions

Relationship Between the Nacos Engine and Microservice Framework

CSE 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

CSE Nacos Engine Version	Spring Cloud Alibaba Version	Spring Cloud Version	Spring Boot Version
	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

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	1.10.9-2021.0.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**

Framework	Recommended Versions	Description
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	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.
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 Nacos engine instances.

Table 8-1 Nacos engine resource quota

Resource	Quota	Modifiable	Precaution
Number of namespaces in a single Nacos instance	50	No	-
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 8-2** lists the maximum number of resources that can be created in ServiceComb engine instances.

Table 8-2 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	-

Function	Resource	Quota	Modifiable	Precaution
	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.

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 securely access cloud 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 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, Nacos engine, or application gateway, an IAM user must be granted the TMS ReadOnlyAccess permission to identify and manage ServiceComb engines, Nacos engines, or application gateways.
IAM user import	Identity and Access Management (IAM)	To import IAM users, the IAM ReadOnlyAccess permission is required.

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	√
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	Scenario
cse:engine:list	List all microservice engines	None	• View ServiceComb engine list • View Nacos engine list
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.

Permission	Description	Permission Dependency	Scenario
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
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:dashboard:get	View the ServiceComb engine dashboard	• cse:engine:list • cse:engine:get	View service dashboard information of a ServiceComb engine
cse:dashboard:modify	Modify the ServiceComb engine dashboard	• cse:engine:list • cse:engine:get • cse:dashboard:get	Modify the dashboard configuration of a ServiceComb engine

Permission	Description	Permission Dependency	Scenario
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
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:list • 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

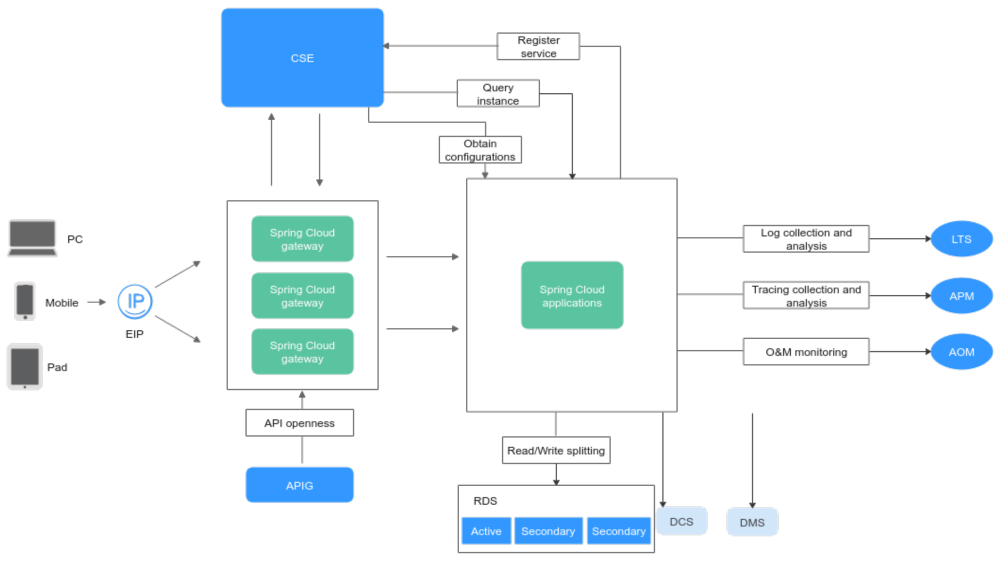
- **IAM Service Overview**
- **Creating a User and Granting Permissions**

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.