ServiceStage

Service Overview

Issue 01
Date 2018-09-30
Copyright © Huawei Technologies Co., Ltd. 2018. 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 Technologies Co., Ltd.

Trademarks and Permissions

and other Huawei trademarks are trademarks 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 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.
# Contents

1 Product Overview.................................................................1
2 Product Functions.................................................................2
3 Product Advantages..............................................................3
4 Application Scenarios............................................................4
5 Product Terms........................................................................6
6 Tools.....................................................................................10
   6.1 LocalServiceCenter..........................................................10
   6.2 Java SDK........................................................................11
   6.3 Go SDK..........................................................................25
   6.4 Service Mesh.................................................................29
   6.5 Key Generation Tool......................................................33
   6.6 Local CSE.......................................................................33
7 Edition Description..............................................................34
ServiceStage is a one-stop PaaS service that provides cloud-based application hosting, simplifying application lifecycle management, from deployment, monitoring, O&M, to governance. It provides a microservice framework compatible with mainstream open-source ecosystems and enables quick building of distributed applications.
2 Product Functions

- Orchestration
  Application lifecycle management and graphical application orchestration.
- Microservice engine
  Multi-language microservice framework and microservice pipelines that allow access without changes.
- Performance management
  Multi-dimensional metric monitoring, elastic scaling, and analysis of microservice call chains.
- Software repository management
  Management of private and public container images and software packages.
3  Product Advantages

- Agile
  Provides a unified platform for managing the application lifecycle, from development, release, to O&M.

- Open
  Features framework- and platform-independent, and supports open source systems Kubernetes and Spring Cloud. Existing applications can be smoothly migrated to the cloud.

- Easy to use
  Supports graphical application and resource orchestration, and allows access without changes to existing applications. Applications are deployed on VMs or containers.

- Intelligent
  Supports non-intrusive probes and full-stack topology visualization. Application monitoring, alarms, and governance facilitate self-healing in the event of faults.
4 Application Scenarios

Application Hosting

ServiceStage supports automatic on-cloud deployment and O&M, improving deployment efficiency. It is specifically suited for enterprise IT systems and Internet applications.

ServiceStage has the following advantages:

- **Automatic deployment**
  Template-based and graphical application orchestration enables automatic deployment with a few clicks.

- **Hitless upgrade**
  Applications on a device are upgraded without taking the device offline, ensuring service continuity.

- **Automatic O&M**
  Rich functions, including application topology, monitoring, alarms, logs, and call chains, are provided for automatic O&M.

Microservice Applications

ServiceStage improves scalability and efficiency of development, deployment, and O&M, and accelerates service innovation for e-commerce and gaming applications.

ServiceStage has the following advantages:

- **Usable out-of-the-box**
  Integrates capabilities such as microservice registration, discovery, communication, and governance.

- **One-stop lifecycle management**
  Manages clusters and software repositories, and supports application development, release, and O&M.

- **Performance monitoring**
  Provides the commercial Application Performance Management (APM) service and supports application topologies and call chains.
SaaS Applications

ServiceStage provides multi-tenant solutions that enable enterprises to sell services. These solutions are specifically suited for SaaS applications in the energy and education domains.

ServiceStage has the following advantages:

- **Multi-tenant isolation**
  Provides a physical/logical multi-tenant solution to isolate tenant data.

- **Automatic deployment**
  Supports automatic deployment of applications by tenants.

- **Multi-tenant O&M**
  Supports unified upgrade, monitoring, alarms, logs, and performance analysis for multiple tenants.
## 5 Product Terms

### Table 5-1 ServiceStage terms

<table>
<thead>
<tr>
<th>Full Expansion</th>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster</td>
<td>Cluster</td>
<td>A cluster is a combination of cloud resources required for container running. A cluster contains cloud resources such as elastic cloud servers (ECSs) and elastic load balancers (ELBs).</td>
</tr>
<tr>
<td>Node</td>
<td>Node</td>
<td>A node is a VM on which containerized applications run. A node agent kubelet runs on each node. The kubelet is used to manage container instances running on nodes.</td>
</tr>
<tr>
<td>Container</td>
<td>Container</td>
<td>A container is a running instance created from a container image. Multiple containers can run on one node.</td>
</tr>
<tr>
<td>Orchestration</td>
<td>Orchestration</td>
<td>An orchestration template contains the definitions of a group of container services and their dependencies. Orchestration templates can be used to deploy and manage multi-container applications and non-containerized applications.</td>
</tr>
<tr>
<td>Stack</td>
<td>Stack</td>
<td>A stack is a deployment instance composed of applications, services, and resources. ServiceStage manages these elements by means of stacks.</td>
</tr>
<tr>
<td>Containerized Application</td>
<td>Containerized Application</td>
<td>A containerized application can be created using a container image or a template. Each containerized application contains one or multiple containers.</td>
</tr>
<tr>
<td>Repository</td>
<td>Repository</td>
<td>Both image repositories and software repositories are called repositories. Images are used to create containerized applications; while software packages are used to create non-containerized applications. Before creating an application, ensure that the required image and software package has been uploaded to the repository.</td>
</tr>
<tr>
<td>Full Expansion</td>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Dark Launch    | -       | CSE provides two dark launch policies:  
  * Traffic diversion based on traffic percentage  
    Set traffic diversion rules, diverting a certain percentage of traffic to a specific service version.  
  * Traffic diversion based on specific requests  
    Set traffic diversion rules based on the request header, diverting traffic having certain characteristics to a specific service version.  
  Dark launch helps with smooth service upgrade. |
| Application    | -       | An application is a logical computer software entity that provides specific functions to users. An application built using the microservice architecture is generally composed of multiple microservices. |
| Circuit Breaker| -       | Circuit breaker is designed to ensure service availability. Microservices generally depend on each other. A service call chain may involve multiple microservices. If the access delay of one or more services is too high, incoming service requests will accumulate and consume more threads and I/O resources, which hinders system running and make more services unavailable. To solve this problem, when a target service responds slowly or a large number of timeout events occur, circuit breaker breaks the service call and returns subsequent call requests to quickly release resources. The service will not be called until it can respond properly. |
| Cloud Service Engine | CSE       | CSE, short for Cloud Service Engine, is a HUAWEI CLOUD service. It provides a group of tools and development frameworks at the development layer to facilitate local microservice development, and a group of basic services that support microservice operation and O&M on the cloud, facilitating the deployment and O&M of microservices in the cloud production environment. |
| Consumer       | -       | A consumer is the caller of a service. |
| Distributed Transaction | -       | The participator, resource server, and transaction manager of a distributed transaction are deployed on different nodes in a distributed system. CSE supports two distributed transaction solutions:  
  * The final consistency solution based on the TCC protocol  
  * The strong consistency solution based on two-phase commit (2PC) |
<p>| Fault Injection | -       | Fault injection constructs a scenario where the service API returns an exception or the response time is long, which is used by microservices to test abnormal scenarios. |</p>
<table>
<thead>
<tr>
<th>Full Expansion</th>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Tolerance</td>
<td>-</td>
<td>Fault tolerance is a way to handle exceptions that occur while consumers access a service. If an exception occurs, the service framework automatically selects a new service route to be called.</td>
</tr>
<tr>
<td>Identity and Access Management</td>
<td>IAM</td>
<td>Identity and Access Management (IAM) is a security management service for enterprise tenants in FusionStage. It manages account information, role permissions, access control, and logs, as well as providing diversified account management and sign-in authorization and authentication.</td>
</tr>
<tr>
<td>Instance</td>
<td>-</td>
<td>An instance is the minimum running and deployment unit of a microservice. Generally, it corresponds to an application process.</td>
</tr>
<tr>
<td>Independent Software Vendor</td>
<td>ISV</td>
<td>The independent software vendor (ISV) is a type of complementary service provider. Develop and sell the software system that can be operated by yourself. ISVs have an operable software system on their own, which supports software-as-a-service (SaaS). They can deploy the software system in the carrier's environment and develop their own customers.</td>
</tr>
<tr>
<td>Isolation</td>
<td>-</td>
<td>Isolation is an exception detection mechanism. It is used when a request timeout or large traffic occurs. Its common parameters include timeout duration and the maximum number of concurrent requests. An exception is recorded when a timeout occurs or when the maximum number of concurrent requests is exceeded. The error rate and error request quantity are calculated in the circuit breaker mechanism based on the records.</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>-</td>
<td>Route load balance is required when an application accesses a microservice that has multiple instances. CSE provides a Ribbon-based load balancing solution. The load balancing policy can be set in a configuration file. Currently, the policy can be random or based on polling, session stickiness, or response time weight.</td>
</tr>
<tr>
<td>Legacy</td>
<td>-</td>
<td>A legacy is a system that is in use but reaching the end of its software lifecycle.</td>
</tr>
<tr>
<td>Microservice</td>
<td>-</td>
<td>A microservice is a lightweight service-oriented architecture (SOA) that is widely used to structure cloud applications and Internet applications as a collection of loosely coupled distributed services.</td>
</tr>
<tr>
<td>Provider</td>
<td>-</td>
<td>A provider is a called microservice.</td>
</tr>
<tr>
<td>Full Expansion</td>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Rate Limiting</td>
<td>-</td>
<td>When resources are insufficient for service processing, the service framework needs to limit the access requests from consumers, enabling the flow control protection mechanism. Traffic can be controlled at both the consumer and the provider ends. At the consumer end, you can limit the frequency of request sending to a certain provider. At the provider end, you can control the request sending frequency of each consumer or limit the overall request sending frequency based on the provider's resource consumption, preventing service breakdown caused by resource exhaustion.</td>
</tr>
<tr>
<td>Service Degradation</td>
<td>-</td>
<td>Service degrade includes block degrade and tolerance degrade policies. In a block degrade, maintenance personnel or developers forcibly degrade a type of services or a certain service when an external trigger condition reaches a certain threshold. In a fault tolerance degrade, when non-core services are unavailable, the service logic automatically bypasses the faulty services to ensure that core services run properly.</td>
</tr>
<tr>
<td>Service Mesh</td>
<td>-</td>
<td>Service Mesh provides an infrastructure-layer service. To develop applications based on microservices, developers need to solve the issues in application running on a distributed network, such as fault tolerance, rate limiting, load balancing, registration discovery, and monitoring. Service Mesh is used as a proxy at L4/L7 to address these issues.</td>
</tr>
<tr>
<td>Service</td>
<td>-</td>
<td>A service is an on-demand function object. In an application model, services are generally oriented to applications. Applications need to subscribe to services and bind to them before using them. In certain business scenarios, pay-as-you-go services are provided.</td>
</tr>
<tr>
<td>Try/Confirm/Cancel</td>
<td>TCC</td>
<td>Try/Confirm/Cancel (TCC) is a transaction management framework of the eventual consistency. TCC transactions can be classified as compensation transactions.</td>
</tr>
<tr>
<td>Web Services Atomic</td>
<td>WSAT</td>
<td>Web Services Atomic Transaction (WSAT) is a transaction management framework of the strong consistency.</td>
</tr>
</tbody>
</table>
6 Tools

6.1 LocalServiceCenter
6.2 Java SDK
6.3 Go SDK
6.4 Service Mesh
6.5 Key Generation Tool
6.6 Local CSE

### 6.1 LocalServiceCenter

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.73</td>
<td>2018.11.8</td>
<td>Fixed bugs.</td>
</tr>
<tr>
<td>2.2.72</td>
<td>2018.09.21</td>
<td>Added the command tool.</td>
</tr>
<tr>
<td>2.2.67</td>
<td>2018.08.02</td>
<td>Added new features.</td>
</tr>
<tr>
<td>2.2.65</td>
<td>2018.07.03</td>
<td>Fixed bugs.</td>
</tr>
<tr>
<td>2.2.63</td>
<td>2018.06.05</td>
<td>Improved usability and fixed bugs.</td>
</tr>
<tr>
<td>2.2.60</td>
<td>2018.06.01</td>
<td>Added new features.</td>
</tr>
</tbody>
</table>

#### 2.2.73

- Fixed bugs
  - The service contract failed to be tested by using an instance having a 5-digit port number.
  - The extended attributes could not be cleared when microservices and instance extended attributes were updated.
2.2.72

- New features
  - Adds the scctl command tool.

2.2.67

- New features
  - The instance query list is added.
  - The topology of microservice dependencies is added.

2.2.65

- Fixed bugs
  - The contract test page occasionally failed to be displayed.
  - The number of records on the service information query page was incorrect.

2.2.63

- New features
  - Supports contract tests on the UI.
  - Microservices can be directly deleted on the UI.
  - Starts UI offline.

2.2.60

- New features
  - Supports UI.
  - Supports IPv6.

6.2 Java SDK

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.52</td>
<td>2018.11.10</td>
<td>This version maps to CSE2.3.22. Simplified Spring and Spring Boot integration, supported proxy information encryption, and enhances features such as Spring MVC development mode and isolation, and fixed some bugs.</td>
</tr>
<tr>
<td>2.3.47</td>
<td>2018.09.05</td>
<td>This version maps to CSE2.3.20. Provided new features related to zero interruption in service calling in the rolling upgrade scenario, event mechanism-based non-intrusive service tracing, integration with Spring Boot 2.0, and governance capability of accessing the third-party service clients. Optimized the configuration update procedure, load balancing control procedure, and exception handling mechanism, and fixed some bugs.</td>
</tr>
<tr>
<td>Version</td>
<td>Release Date</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.3.35</td>
<td>2018.08.02</td>
<td>This version maps to CSE2.3.17. Supported BeanParam, weak typing calling, full-path URL, and Form encoding objects in development mode. Supported instance status diagnosis and monitoring, and enhanced protection against the number of Highway connections. Upgraded the Jackson tool, and corrected spelling errors of some configuration items, and fixed some bugs.</td>
</tr>
<tr>
<td>2.3.30</td>
<td>2018.07.19</td>
<td>This version maps to CSE2.3.16. In development mode, supported reliability mechanisms, such as the default value, cross-domain access configuration, isolation, circuit breaker event reporting, public key-based blacklist and whitelist, and instance status detection. Adjusted and optimized the load balancing filter mechanism, that is, supported DiscoveryTree and ServerListFilterExt extensions, and enabled reliability mechanisms by default, such as instance isolation and retry upon errors. Fixed some bugs.</td>
</tr>
<tr>
<td>2.3.27</td>
<td>2018.06.19</td>
<td>This version maps to CSE2.3.14. Supported uploading and downloading on the Servlet running environment, client timeout setting based by API, and Produces type extension through SPI. Optimized management and control process when a service API changes, resolved the issue of start failure if the version number is not changed upon an API change, and fixed a few bugs.</td>
</tr>
<tr>
<td>2.3.25</td>
<td>2018.06.05</td>
<td>This version maps to CSE2.3.13. Supported new functions and scenarios for the three modules: graceful exit, edge service, and access log. Optimized the client microservice routing mechanism and upgraded the third-party software related to validation.</td>
</tr>
<tr>
<td>2.3.23</td>
<td>2018.06.01</td>
<td>This version maps to CSE 2.3.12. Supported HTTP 2.0 communication, and self-reference data type, upgraded Spring and Netty, and optimized API management related to service instance reliability and contract registry.</td>
</tr>
<tr>
<td>2.3.20</td>
<td>2018.05.16</td>
<td>This version maps to CSE 2.3.11. Optimized the experience of integration between the configuration center and service center, also optimized Spring Cloud, and fixed a few bugs.</td>
</tr>
<tr>
<td>2.3.18</td>
<td>2018.04.26</td>
<td>This version maps to CSE 2.3.10. Provided JSR Validation and file downloading capabilities, optimized metrics, monitoring data reporting, and error control functions, and fixed some bugs.</td>
</tr>
<tr>
<td>Version</td>
<td>Release Date</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.3.12</td>
<td>2018.04.03</td>
<td>This version maps to CSE 2.3.9. Supported communication using OpenSSL, service instance isolation based on the environment, API-level rate limiting for the provider, and fault injection for the consumer. Also, modified some features and fixed some bugs.</td>
</tr>
<tr>
<td>2.3.9</td>
<td>2018.03.06</td>
<td>This version maps to CSE 2.3.8. The mapping service center version is upgraded to 2.2.45, which provided new functions such as reporting the version of the framework component that supports microservice O&amp;M.</td>
</tr>
<tr>
<td>2.3.8</td>
<td>2018.02.03</td>
<td>This version maps to CSE 2.3.7. Changed the open-source package name from <code>io.servicecomb</code> to <code>org.apache.servicecomb</code>, provided generic support for REST, modified some features, and fixed some bugs.</td>
</tr>
<tr>
<td>2.2.39</td>
<td>2018.01.11</td>
<td>This version maps to CSE 2.3.5. Interconnected Spring Cloud features with CSE, improved the reliability of instance query, optimized reactive invocation and other features, and fixed some bugs.</td>
</tr>
<tr>
<td>2.2.31</td>
<td>2018.01.09</td>
<td>This version maps to CSE 2.3.4. Fixed some bugs. No features added.</td>
</tr>
<tr>
<td>2.2.28</td>
<td>2017.12.25</td>
<td>This version maps to CSE 2.3.3. Added a starter to Spring Cloud to interconnect with the service center. Fixed bugs on the governance module and dark launch module.</td>
</tr>
<tr>
<td>2.2.21</td>
<td>2017.12.18</td>
<td>This version maps to CSE 2.3.2. Added password authentication when the TCC transaction uses the Redis database, repaired the strong consistency transaction, and fixed some bugs on the common module.</td>
</tr>
<tr>
<td>2.2.13</td>
<td>2017.12.13</td>
<td>This version maps to CSE 2.3.1. Fixed some bugs. No features added.</td>
</tr>
<tr>
<td>2.2.11</td>
<td>2017.12.08</td>
<td>This version maps to CSE 2.3.0. The change history is provided based on version 2.2.8. Added the public and private key pair authentication mechanism between microservices.</td>
</tr>
</tbody>
</table>

### 2.3.52

- **New features**
  - The shortest isolation time can be configured for the `servicecomb.loadbalance.[serviceName].isolation.minIsolationTime` configuration item. This resolves the
issue that instances are quickly restored after being isolated in the scenario where multiple threads access a server at the same time.

- You can set the service instance address through LocalContext instead of through service discovery.
- The collection format parameter can be used to specify the cvs format for the query parameter of the List type.
- The ContextUtils can obtain InvocationContext through getFromCompletableFuture.
- The .pom files cse-dependency-spring4, cse-dependency-spring5, cse-dependency-spring-boot4, and cse-dependency-spring-boot5 are added to simplify the use of these components.
- spring-boot2-starter-discovery is added so that you can use service center to discover services in Spring Boot 2 (for example, Zuul).
- When Spring Boot is integrated with CSE, the RestController is scanned and identified as a REST API, reducing code volume in REST framework reconstruction. You can disable this feature through the servicecomb.provider.rest.scanRestController configuration item.
- Proxy supports password encryption.

**Modified features**

- NoRouteToHostException supports retry, that is, retry is performed when a container is offline.
- The SwaggerGeneratorContext loading process is optimized, that is, logs are added to facilitate fault locating.
- Retry is supported for IOException if the error message is "connection reset".
- Spring 4 can be upgraded to 4.3.18.RELEASE, and spring 5 can be upgraded to 5.0.7.RELEASE. Spring Boot 1 can be upgraded to 1.5.14.RELEASE, and Spring Boot 2 can be upgraded to 2.0.4.RELEASE.
- When all retries fail, the system returns the exception of the last invocation instead of returning 490.
- The default value of servicecomb.loadbalance.continuousFailureThreshold is changed from 2 to 5 to solve the issue that instances are easily isolated under retry conditions.
- The output format of metrics data is optimized, that is, more detailed performance statistics are output.
- The TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 and TLS_RSA_WITH_AES_128_GCM_SHA256 insecure algorithms are removed from the default SSL algorithms. If you need to use such an insecure algorithm, you must specify it in the configuration file.
- Multiple default settings are supported by the HTTP2 protocol to ensure that the HTTP2 protocol runs stably by default.
- A change is introduced during resolution of the DoS attack issue of the Jackson project. Specifically, if a service implements AbstractRestObjectMapper, the JsonFactory must be set. For details, see the implementation of RestObjectMapper. This change is temporary. After the DoS attack vulnerability is rectified and Jackson is upgraded, the code may be adjusted accordingly.

**Fixed bugs**
In the Spring loading process, some beans do not have the implClass attribute. As a result, NPE occurs.

- The body parameter did not support the enum type.
- The repeated Produces or consumes statement was not correctly overwritten.
- A deadlock occurred when the service center instance failed to be queried immediately after the service was successfully registered.
- DoS issues caused by data types such as BigInteger.
- By default, DynamicPropertyUpdater log output was disabled, and configuration items were not printed.
- API calling failed because the local cache was not refreshed once the temporary AK/SK expired.
- The response was slow when the vert.x was used to download large-sized files.
- Release of the API with type CustomGeneric<Map<String or String>> (the type description contains comma) failed.

### 2.3.47

- New features
  - Supports hitless rolling upgrades.
  - Uses the revision mechanism to query configuration items in the configuration center. If the revision does not change, no configuration is returned, reducing network transmission.
  - The instance cache time and instance detection polling time related to instance isolation can be configured using servicecomb.loadbalance.stats.serverExpiredInSeconds and servicecomb.loadbalance.stats.timerIntervalInMils.
  - Enhances blacklist and whitelist functions. Specifically, supports matching between attributes of the String type in any microservice and those in the Properties.
  - For the primitive type, the system provides the default value if you do not set this parameter in the case of `required=false`.
  - Supports customization of Consumer ObjectMapper so that the client does not serialize objects of the null field. The server can use the customized default value.
  - Supports serialization and deserialization of error messages of the customized types, serialization and deserialization of unknown error messages, and conversion of error messages.
  - Supports BootListener loading by SPI.
  - Supports Spring Bean loading.
  - Supports configuration of the maximum waiting time of messages in a processing queue through servicecomb.rest.server.requestWaitInPollTimeout and servicecomb.highway.server.requestWaitInPollTimeout. If the waiting times out, the system does not process the request and returns a timeout error to the client.
  - Specifies consumes and produces using `@Api`.
  - Sets the value to `null` when the servicecomb.rest.parameter.query.emptyAsNonNull parameter is used to parse query parameters and the parameter is empty or does not exist.
  - Obtains SCBEngine reference from BootEvent.
- The TraceID is generated in the calling entry (edge, provider, or consumer) to trace request calling. Obtains the TraceID from Invocation.getTraceId.
- Supports configuration of the waiting time (from the time when the service starts to the time the service is registered successfully) through servicecomb.boot.waitUp.timeoutInMilliseconds to reduce the possibility of a request failure if an API is called immediately upon service start.
- Supports configuration of the maximum length of the first line in an HTTP message through servicecomb.rest.server.maxInitialLineLength.
- Supports the abstract object type for object deserialization by using servicecomb.rest.parameter.decodeAsObject.
- Uses servicecomb.rest.parameter.query.ignoreDefaultValue to forcibly disable the default value. This meets the special requirement in which the edge service does not generate a default value.
- Supports GlobalRestFailureHandler by SPI to process unknown exceptions when an edge service request is parsed.
- Reports non-intrusive tracing data through the event mechanism. Example: samples/apm-agent/pom.xml.
- Supports parameter name definition by `name` and `value` in SpringMVC mode.
- ExceptionToResponseConverter supports base classes. The default error conversion mechanism can be provided by implementing Throwable.
- Supports integration with Spring Boot 2 and provides two integration modes: Spring-boot2-starter-servlet and spring-boot2-starter-standalone.
- Supports ApiParam to define the description of a parameter. The generated contract contains the description.
- Provides configuration items servicecomb.http.filter.server.serverRestArgs.enabled and servicecomb.http.filter.client.default.enabled to disable ServerRestArgsFilter and DefaultHttpClientFilter respectively.
- Supports third-party REST service calling by CSE client API and manages the client.

### Modified features

- The default value of errorThresholdPercentage is changed from 2 to 0. That is, the error rate is not considered and only the number of consecutive errors is considered.
- For the primitive type, in the case of `required=false`, if the user does not enter the parameter, the system provides the default value. In the case of `required=true`, if the user does not enter the parameter, an exception is thrown. In SpringMVC mode, the default value is `required=true`. In JAX-RS mode, only `required=false` is supported, which complies with the standard protocol. In earlier versions, the default value is `false` in SpringMVC mode. Therefore, this is a modified feature and a fixed bug. This difference should be noticed in an upgrade.
- The load balancing module does not depend on the IRule and ServerListFilter components of Robin, and provides new APIs RuleExt and ServerListFilterExt for extension. For details, see https://bbs.huaweicloud.com/blogs/aa2b97ee9c511e89fc57ca23e93a89f. After an upgrade, you can use servicecomb.loadbalance.strategy.name to specify the load balancing policy. The servicecomb.loadbalance.policy configuration item is deleted.
- Timeout interval servicecomb.rest.server.timeout of the asynchronous Servlet is set to -1 by default, that is, no timeout is set. This solves the problem that the asynchronous Servlet timeout thread may occupy the service processing thread,
leading to request blocking. In most cases, you do not need to set this parameter and use the default value.

- ExceptionToResponseConverter supports base classes. The default error conversion mechanism can be provided by implementing Throwable.

- Upgrades guava from 19.0 to 21.0.
- Upgrades guice from 4.0 to 4.1.0.
- Supports protostuff-parser 2.2.25.
- Upgrades vert.x from 3.5.0 to 3.5.3.
- Removes the dependency on cse-handler-tcc from se-solution-service-engine and removes tcc-provider and tcc-consumer from Handler. If the service needs to use the TCC function provided by the module, the dependency of the module needs to be added explicitly in the code and the corresponding processing needs to be configured. The functions provided by this module will be replaced by other transaction frameworks. New functions of the service will not be constructed based on this function.

- Fixed bugs
  - Error code 400 was returned when the body parameter could not be parsed.
  - When Tomcat was in non-NIO, the available method of InputStream could not read parameters if value 0 was returned.
  - In version rollback (for example, the API was deleted incorrectly and a rollback was required), the API of an earlier version could not be accessed.
  - Files failed to be downloaded sometimes when multiple threads were processed.
  - Coding was error due to special characters contained in the Path parameter.
  - Error code 590 was returned when the JSON format was incorrect. However, error code 400 should be returned.
  - The servicecomb prefix used in the Bizkeeper module did not take effect.
  - The parameter verification failure message contained parameter names arg0 and arg1 instead of the actual name.
  - In the timeout scenario, the SDK printed a great volume of stack information.
  - The edge service failed to parse the URL and did not immediately return a response when the URL contained the special characters %%A. As a result, the response timed out.
  - The path parameter was properly encoded to resolve semantic errors of special characters.
  - Flow control refused the request for a long time when the time changed.

2.3.35

- New features
  - Requests sent from the edge service better support the form format. Allows Key conversion from Json to form.
  - In Spring MVC encoding mode, a POJO object is queried, which reduces the number of API parameters.
  - Provides a mechanism for the O&M tool to diagnose, query, and update the instance list cached on the client.
  - RestTemplate supports weak typing, such as JsonObject or String, to access server APIs.
- In Servlet mode, the container path can be registered with the base path of Swagger so that Swagger can access the server using the full-path URL.
- The JAX-RS development mode supports BeanParam.
- Highway supports setting the maximum number of connections through `servicecomb.highway.server.connection-limit`.

**Modified features**
- Changes `RestObjectMapper.INSTANCE` to `RestObjectMapperFactory.getRestObjectMapper`.
- Configuration items support aliases, specifically, `cse.credentials.*` and `cse.monitor.*` support aliases `servicecomb.credentials.*` and `servicecomb.monitor.*`.
- Changes `servicecomb.monitor.client.sslEnalbed` into `servicecomb.monitor.client.sslEnabled`.
- Changes `servicecomb.monitor.client.enable` to `servicecomb.monitor.client.enabled`.
- Upgrades Jackson from 2.9.5 to 2.9.6.

**Fixed bugs**
- If `cse-solution-service-engine` contained the edge module, the thread pool for service code execution changed to the reactive mode and the dependency was removed.
- The address of the monitoring service was changed to support a port number in a URL and the default port number was used. For example, `http:80`, and `https:443`.
- `HttpServerFilter` threw a `NullPointerException` (NPE); therefore, the server did not correctly respond to the NPE. As a result, the request timed out.
- In the Linux OS, the main class path was incorrectly scanned. As a result, the Spring loading path could not be found and an error was reported when the combined Log4j file was generated.
- A large number of messages indicating repeated `cse.*` and `servicecomb.*` configuration items were printed.
- When fault injection was enabled, the client might fail to respond to services.
- In reactive mode, a large number of warning logs were generated during graceful shutdown.
- When an instance was removed in sticky session mode, the deleted instance was still connected within the session period.
- After an instance went offline, the client kept the instance online for a long time.
- In the scenario where the edge service called services using Tomcat as containers, if the Filter API returned error code 401, the browser obtained error code 490.

**2.3.30**

**New features**
- Supports default value definition of parameters by labels in JAX-RS and Spring MVC development modes.
- Supports settings on cross-domain request access.
- Sends events related to service governance (such as circuit breaker and instance isolation), which is supported in service development.
- Supports blacklist and whitelist based on the public key.
- Scans the package of the class where the main method locates by default during Spring start, simplifying user configuration.
- Opens the LocalContext API.
- Supports the form format for the request from the Edge service and then the form format is automatically converted to jason format through the Edge service.
- Provides the client ping mechanism to check whether the client-cached instances are available through ping extension. This function is enabled by default and is used together with the instance isolation function to isolate instances that fail to be detected.

### Modified features

- Adds aliases servicecomb.executor.groupThreadPool and servicecomb.executor.reactive to cse.executor.groupThreadPool and cse.executor.reactive.
- By default, BeanUtils.init scans the package where the main method locates, simplifying user configuration.
- By default, the path information parsed by the schema definition is not registered with the service center. Configures the servicecomb.service.registry.registerPath=true configuration item to enable registration of such path information.
- Replaces the cse.xxx configuration item with the servicecomb.xx configuration item. In this case, servicecomb.xx is preferentially used in programs and configuration files.
- Changes cse-config-order in the configuration file to servicecomb-config-order.
- Modifies cse.configurationSource.additionalUrls and cse.configurationSource.defaultFileName to servicecomb.configurationSource.defaultFileName and servicecomb.configurationSource.defaultFileName respectively.
- Prints schema content to logs when detecting that the schema of the service center is different from that of the local schema to reduce the locating cost.
- Switches functions such as instance isolation, attribute-based routing, and data center-based routing to DiscoveryFilter.
- Modifies implementation of ServerListFilterExt. Specifically, Ribbon ServerListFilter is not inherited, and the new method public List<Server> getFilteredListOfServers (List<Server> servers, Invocation invocation) is provided to support the filter based on the Invocation attribute. Compilation fails if ServerListFilterExt is used. If necessary, you can extend the new ServerListFilterExt or DiscoveryTree.
- Renames the CseServer property, that is, changes it to ServiceCombServer, and deletes the LoadBalancerStats property.
- Enables instance isolation by default. If cse-solution-service-engine is used, the retry function is enabled by default.

### Fixed bugs

- The difference between the value of the first period and that of the following period in rate limiting was 1.
- Deserialization failed when the return value was `void`.
- The NullPointerException (NPE) was thrown when the value of ContextClassLoader was null.
- In the RequestBody (required = false) scenario, an exception occurred when the empty body was parsed.
- If the server did not start, the client started and the API was called. After the server started, the client could not be called.
- A service might fail to be registered if the NIC information failed to be obtained during the microservice start.
- The host information was not correctly printed in the case of a short connection for the Access Log.
- When a service was deleted, the client continued to query the service version periodically.

2.3.27

- New features
  - The J2EE (Servlet) running environment supports file upload and download.
  - The request timeout duration can be dynamically set based on the API level.
  - The mapping rule is optimized. That is, mapping rules can be loaded from mapping files of different modules. One key can be mapped to multiple keys.
  - The public cloud supports the use of PAAS_CSE_ENDPOINT environment variables to specify the service center and configuration center addresses.
  - The Produces type can be extended through SPI.

- Modified features
  - Microservice contract registry is optimized. That is, a microservice fails to start when a microservice of the same version is detected but the environment is not development. The environment configuration item is changed to `service_description.environment`.

- Fixed bugs
  - When the Edge forwarded the REST service request from the Tomcat container and the Trunked encoding was used, the browser (or a third-party HTTP tool) failed to parse the response.
  - In the JAX RS development mode, a contract was automatically generated and an error was reported when a contract was manually written.
  - Memory leakage may have occurred on Edge Service MicroserviceVersion.
  - When the Edge Service was abnormal, error code 502 was returned instead of the provider error code.
  - Concurrent access to the DiscoveryTree was resolved.
  - When dark launch used uppercase and lowercase sensitive rules and the parameter value was empty, an exception occurred.
  - Response parsing failed when the Generics parameter type contained the Generics field.
  - The system could not exit gracefully in abnormal scenarios.
  - When the `cse.*` and `servicecomb.*` configuration items coexisted, parameter values were incorrect.

2.3.25

- New features
- The J2EE running environment (Tomcat) supports graceful exit. The event mechanism is added to the graceful exit of the Vert.x running environment, and the logic for graceful exit of these two environments is unified.
- The edge service provides two general routing management mechanisms: dark management based on service names and versions, and dark management based on URL mapping. This helps to simplify edge service usage.
- The access log can print context header formation, and the supported format can be expanded.

**Modified features**
- The edge mechanism is used to manage consumer contracts. By default, all versions (0+) are pulled (the latest version is pulled previously). In addition, the isolated ClassLoader is used to call APIs for different versions to resolve abnormal API call and recovery failure, as the customer is upgraded before the provider with API changes (the customer changes accordingly).
- The validation-api is upgraded from 1.1.0 to 2.0.0.
- The hibernate-validator is upgraded from 5.2.4.Final to 6.0.2.Final.
- The hibernate-validator-annotation-processor is upgraded from 5.2.2.Final to 6.0.2.Final.

**Fixed bugs**
- The configuration center printed null pointer exceptions (NPEs) cyclically in watch mode.

### 2.3.23

**New features**
- Interface definition supports the data type that is self-referenced.
- Graceful shutdown is supported. Specifically, when the service stops, the network thread is closed and waits for the call to finish.
- SDK supports HTTP 2.0 as the communication protocol.
- In contract definition, x-java-interface is optional.

**Modified features**
- The ServiceComb Java Chassis version is commit id 5fc99f5.
- Provides an instance protection mechanism when the service center network is temporarily unreachable or the service center restarts due to a fault. When an empty instance is queried from the service center, the current instance is pinged by default. If the instance is successfully pinged, the local instance is not removed.
- Upgrades the Spring version from 4.3.5.RELEASE to 4.3.16.RELEASE. Note: In the new version, when Spring's native RestTemplate API has unknown HTTP error codes, the API behavior changes. However, APIs provided by CSE are not affected.

**Fixed bugs**
- Configuration item query of the configuration center failed because the microservice name was not transcoded.
- An exception occurred when a large-size file was upgraded.
- The server encountered NPEs when an empty file was uploaded by the consumer.
- The microservice failed to register with the service center if the microservice started before the service center.
- Microservices that had information deleted during microservice registry could not be registered.

### 2.3.20

- **Modified features**
  - During the initialization of the configuration center, the pre-configuration function is supported after one configuration query is performed.
  - The port number does not need to be configured in the service center address. Instead, the default port number can be used according to the protocol.
  - Spring Boot's `application.yml` configuration file has a higher priority than the `microservice.yaml` file.
  - A service instance fails to be queried when the `environment` parameter is configured. This problem involves changing the API for querying a service center instance. The `environment` parameter needs to be carried in the API.

- **Fixed bugs**
  - A service instance failed to be queried when the `environment` parameter was configured.
  - Modified the `traceId` generation and printing functions in the EdgeService scenario.

### 2.3.18

- **New features**
  - Adds the environment to which the service metrics belong to report the service metrics to the dashboard.
  - Prints the data collected by metrics in logs.
  - Adds a simplified method for setting configuration items.
  - Supports the JSR Validation API.
  - Supports file downloading.
  - Supports integration with the configuration file of Spring Boot or Spring Cloud.
  - Provides `cse-solution-spring-cloud` to simplify Spring Cloud application access to CSE.

- **Modified features**
  - Caches the SPI service instance to avoid creating new instances each time.
  - Optimizes the metrics to align with the open-source ecosystem, and discards the perf-stats processing chain. Enables the metrics module after it is referenced through `servicecomb.metrics.publisher.defaultLog.enabled=true`.
  - Adjusts default values such as the metric release period and determines whether to print logs. Adjusts the error detection conditions for instance isolation and retries to reduce log printing and unnecessary retries.
  - Upgrades Guava from 16.0.1 to 19.0.
  - Upgrades Jackson from 2.9.4 to 2.9.5.
  - Enhances the process for obtaining the IP address of the data reporting server.
  - Removes the dependency on `slf4j-log4j12` from foundation-auth.

- **Fixed bugs**
  - The client rate limiting did not support dots (.) in the microservice or schema name.
- Changed DarklaunchRule to allow for one call at a time to avoid concurrency 
problems.
- Metrics could not be obtained in some scenarios.
- A concurrent access problem occurred when both WeightedResponseTimeRule and a 
dark launch policy were triggered.

### 2.3.12

- **New features**
  - Supports communication using OpenSSL.
  - Isolates service instances on the environment.
  - Supports API-level rate limiting for the service provider.
  - Supports fault injection for the service consumer.
- **Modified features**
  - Uses HttpClientFilter and HttpServerFilter to parse the REST parameter to support 
user-defined encoding extension.
  - Enables the system to automatically switch to an available server when multiple 
configuration centers are configured.
  - Enables the pull mode of the service center by default.
  - Changes the default size of the HTTP header from 8 KB to 32 KB.
- **Fixed bugs**
  - Null pointer exception (NPE) occurred in the log when the Edge Service could not 
find the service.
  - Enabled SessionSticklessRule to support service-level parameter configuration, 
solving the problem of parameter settings on the governance page not taking effect.
  - The total number of monitoring data requests and the request failure rate were both 
calculated incorrectly.

### 2.3.9

- **New features**
  Reports the version of the framework component to the service center based on O&M 
requirements.
- **Modified features**
  - Upgrades Jackson from 2.8.10 to 2.9.4.
  - Changes the minimum supported version of the service center to 2.2.45.
- **Fixed bugs**
  Memory leakage may have occurred when the Edge service forwarded services that did 
not exist.

### 2.3.8

- **New features**
  - Provides generic support for REST.
  - Provides user-defined configuration items for Netty DNS.
  - Integrates the view function from Spring Web MVC.
- Adds the TraceId in the access log.

**Modified features**
- Changes the ServiceComb code package name to `org.apache.servicecomb`.
- Adds request success/failure records to the Invocation plug-in.
- Migrates the `com.huawei.paas.cse:foundation-config-cc` module to the `org.apache.servicecomb:config-cc` module.
- Removes the third-party dependency that conflicts with the Apache license.
- Upgrades Netty to 4.1.17 and Jackson to 2.9.2.

**Fixed bugs**
- An exception occurred when a consumer called the highway API because the request parameter name contained a hyphen (-).
- Only the first part of `service_description.description` was registered with the service center because `service_description.description` in the `microservice.yaml` file was separated by commas (,).
- APIGateway could not be connected due to the domain name resolution.

### 2.2.39

**New features**
- Interconnects Spring Cloud applications with CSE.
- Supports the embedded traceID of invocation tracing.
- Uses the `@Api` field to generate tags.
- Supports the CompletableFuture asynchronous invocation feature.

**Modified features**
- Optimizes the instance query mechanism of the service center to improve reliability.
- Optimizes reactive invocation performance.
- Adds service and instance information to the dashboard.

**Fixed bugs**
- Earlier versions of Java SDK could not be accessed because redundant attribute information was delivered during dark launch.
- Configured the loading priority of the cse-solution-service-engine configuration to a minimum, and allowed developers to customize the configuration.
- A request signing failure occurred when multiple service center addresses were configured.

### 2.2.31

**Modified features**
Upgrades the Jackson version to match that of ServiceComb. Deletes duplicate dependency items.

**Fixed bugs**
- The dark launch rules were case sensitive.
- The circuit breaker status was incorrect and the number of circuit breakers was not reported.
- Enabled the Java SDK to provide a standard alarm mechanism to enable the configuration center to report connection failure events.
### 2.2.28

- **New features**
  - Adds a starter to Spring Cloud to interconnect with the service center.
  - Enables the dark launch module to support data types, such as long and double.

- **Fixed bugs**
  - Lots of logs were printed when a large number of services degraded at the same time.
  - Upgraded Jackson to resolve security vulnerabilities.
  - The return value of the strong consistency transaction was lost.
  - The circuit breaker status of the monitoring data was incorrect.

### 2.2.21

- **New features**
  Adds password authentication when TCC transaction logs are stored in the Redis database.

- **Fixed bugs**
  - A .jar package conflict occurred when the metrics module was integrated into the OSGI environment.
  - In scenarios in which AK/SK authentication was used, the order in which the configuration center loaded the decryption tools caused the decryption to fail when first connecting to the service center.
  - SignerUtils incorrectly modified the time zone.

### 2.2.13

- **Fixed bugs**
  The microservice startup fails after using the `System.setProperty` code to configure SSL configuration parameters.

### 2.2.11

- **New features**
  Uses private and public key pairs to authenticate identities between microservices.

- **Fixed bugs**
  Modified the code format and spelling. This modification does not affect Java SDK functions.

### 6.3 Go SDK

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.2</td>
<td>2018.10.24</td>
<td>Disabled the default timeout mechanism.</td>
</tr>
<tr>
<td>1.3.1</td>
<td>2018.10.10</td>
<td>Rectified the fault of cross-app calling.</td>
</tr>
<tr>
<td>Version</td>
<td>Release Date</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1.3</td>
<td>2018.09.17</td>
<td>Supported Go mod and unified sc instance management.</td>
</tr>
<tr>
<td>1.2.7</td>
<td>2018.07.10</td>
<td>Fixed the circuit breaker and service governance issues.</td>
</tr>
<tr>
<td>1.2.5</td>
<td>2018.06.12</td>
<td>Corrected dashboard data.</td>
</tr>
<tr>
<td>1.2.3</td>
<td>2018.06.01</td>
<td>Corrected dashboard data.</td>
</tr>
<tr>
<td>1.2.1</td>
<td>2018.05.11</td>
<td>Fixed the issue of some governance policies not taking effect.</td>
</tr>
<tr>
<td>1.2.0</td>
<td>2018.04.26</td>
<td>Modified the Registry API.</td>
</tr>
<tr>
<td>1.1.3</td>
<td>2018.04.13</td>
<td>Modified the connection to the configuration center and CSE monitoring, and supported dark launch based on user-defined rules (matching the header).</td>
</tr>
<tr>
<td>1.1.0</td>
<td>2018.03.06</td>
<td>Supported multiple projects and dark launch.</td>
</tr>
<tr>
<td>1.0</td>
<td>2018.01.20</td>
<td>This is the first release. Supported rapid development for microservices using the REST/highway protocol and fast access to CSE.</td>
</tr>
</tbody>
</table>

### 1.3.2
- Fixed bugs
  - After configurations are delivered from the configuration center, the service does not take effect after being restarted.

### 1.3.1
- Fixed bugs
  - After the provider was deleted from the service center, an error occurred in updating the consumer cache.
  - After the session holding policy was delivered, a null pointer exception was reported.
  - Cross-app calling failed.

### 1.3
- New features
  - The import path is changed to [github.com/go-chassis/go-chassis](https://github.com/go-chassis/go-chassis).
  - Manages instance cache of the registry center in a unified manner.
  - Supports pluggable configuration center.
  - Provides the runtime package.
  - Provides more route labels.
- Provides the benchmark tool.
- Uses opentracing as the global dotting tool.
- Supports Go 1.11 modules.

**Modified features**
- bootstrap starts plug-ins in order.
- Decouples the authentication module.
- Disables circuit breaker on the provider side.
- Uses Invocation.Ctx to save the header information.
- Maintains dependency information at local.

**Fixed bugs**
- The cache could not be updated after service information was deleted.
- IndexSchema memory leaked.
- Duplicated threads were used for heartbeat detections.
- Tracing higher than level 3 could not be generated.
- Clients were managed by endpoints.

**1.2.7**

**Fixed bugs**
- The circuit breaker and service governance policy did not take effect for the first request.
- When all requests were in circuit breaker, the number of circuit breakers was calculated to the error rate.

**1.2.5**

**Fixed bugs**
- The error rate of the dashboard changed to -1 in certain cases.
- The circuit breaker request was calculated to the number of failures.
- The log recording call failures did not contain server information.
- The service-level load balancing governance policy did not take effect.
- The circuit breaker state on the dashboard was closed after circuit breaker took effect.

**1.2.3**

**Modified features**
- The number of requests in the dashboard is the total number of requests in the last 10s.

**Fixed bugs**
- The dashboard could not distinguish between different instances of the same host.
- The error rate in the dashboard was incorrect.
- The null pointer could not be determined when the provider data was obtained.
- Instance information could not be synchronized after static microservice information was deleted.
- The polling policy did not take effect on multiple versions and multiple instances.
- The local routing rule could not be used after a dark launch rule was deleted.

1.2.1

- Fixed bugs
  - The manual circuit breaker did not take effect.
  - Retry could be performed but the correct result could not be obtained.
  - Forcible service degradation did not take effect.

1.2.0

- Modified features
  - Splits the Registry API into the Registrator, ServiceDiscovery, and ContractDiscovery APIs.
  - Adds the Router API.
  - Supports HTTP 2.0.
- Fixed bugs
  - Data could not be reported to the CSE dashboard after a microservice was deleted.
  - The forceClosed configuration item in the circuit breaker did not take effect.
  - No return code was displayed after the access to a server timed out when the REST protocol is used.
  - The function for setting routing weight rules was unstable.

1.1.3

- Modified features
  - Modifies the access to the configuration center and CSE monitoring.
  - Supports dark launch based on user-defined rules (matching the header).
- Fixed bugs
  - No retry mechanism was provided for service registry during startup.
  - When the routing rule weights were less than 100%, the remaining traffic was not imported to the latest microservice version.
  - The REST protocol did not take effect during an application-cross call.
  - The return code and information for the isolation timeout were incorrect.
  - The weight policy did not take effect.

1.1.0

- New features
  - Supports multiple projects.
  - Supports dark launch (weight).
- Fixed bugs
  - The client did not update the cache when a server instance was added or deleted.
  - The processing results were different when the service access using different protocols timed out.
- The configured timeout duration did not take effect.

1.0

- New features
  - Supports the development of microservices using the REST/highway protocol.
  - Supports functions, such as registry and discovery, load balancing, circuit breaker, service degradation, fault tolerance, fault injection, and routing policy (REST).
  - Supports dynamic governance functions, such as load balancing, circuit breaker, service degradation, fault tolerance, and fault injection.
  - Supports the dashboard monitoring function.

6.4 Service Mesh

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.2</td>
<td>2018.10.10</td>
<td>Rectified memory leakage.</td>
</tr>
<tr>
<td>1.4.12</td>
<td>2018.07.03</td>
<td>Fixed the circuit breaker and service governance issues.</td>
</tr>
<tr>
<td>1.4.10</td>
<td>2018.06.15</td>
<td>Corrected dashboard data.</td>
</tr>
<tr>
<td>1.4.8</td>
<td>2018.06.01</td>
<td>Corrected dashboard data.</td>
</tr>
<tr>
<td>1.4.6</td>
<td>2018.05.11</td>
<td>Fixed the issue of some governance policies not taking effect.</td>
</tr>
<tr>
<td>1.4.5</td>
<td>2018.04.26</td>
<td>Supported HTTP 2.0.</td>
</tr>
<tr>
<td>1.4.4</td>
<td>2018.04.13</td>
<td>Supported multiple projects.</td>
</tr>
<tr>
<td>1.3.7</td>
<td>2018.02.23</td>
<td>Added the default monitoring address.</td>
</tr>
<tr>
<td>1.3.3</td>
<td>2018.01.20</td>
<td>Supported Dubbo applications access.</td>
</tr>
<tr>
<td>1.2</td>
<td>2017.12.28</td>
<td>Simplified configurations and improved usability.</td>
</tr>
<tr>
<td>1.1.2</td>
<td>2017.12.20</td>
<td>Simplified configurations and improved usability.</td>
</tr>
<tr>
<td>1.1</td>
<td>2017.12.08</td>
<td>Supported the dedicated infrastructure layer to help service applications quickly become microservices.</td>
</tr>
<tr>
<td>1.0.5</td>
<td>2017.12.01</td>
<td>Supported the dedicated infrastructure layer to help service applications quickly become microservices.</td>
</tr>
</tbody>
</table>

1.5.2

- New features
  - Supports specifying a provider port during URL request permission by a consumer.
● Modified features
  - The provider can use the environment variable SERVICE_PORTS to set the provider port.
  - Disables circuit breaker on the provider side.
  - Cancels the URI configuration of the CSE monitor in factory defaults.

● Fixed bugs
  - The cache could not be updated after service information was deleted.
  - IndexSchema memory leaked.
  - Duplicated threads were used for heartbeat detections.
  - Tracing higher than level 3 could not be generated.
  - Clients were managed by endpoints.

1.4.12

● Fixed bugs
  - The circuit breaker and service governance policy did not take effect for the first request.
  - When all requests were in circuit breaker, the number of circuit breakers was calculated to the error rate.

1.4.10

● Fixed bugs
  - The error rate of the dashboard changed to -1 in certain cases.
  - The circuit breaker request was calculated to the number of failures.
  - The log recording call failures did not contain server information.
  - When the backend service returned error 500, the client could not read the error information.
  - The service-level load balancing governance policy did not take effect.
  - The circuit breaker state on the dashboard was closed after circuit breaker took effect.

1.4.8

● Modified features
  - The number of requests in the dashboard is the total number of requests in the last 10s.
  - The mesher transparently returns the 5XX information sent from the server.

● Fixed bugs
  - The dashboard could not distinguish between different instances of the same host.
  - The error rate in the dashboard was incorrect.
  - The null pointer could not be determined when the provider data was obtained.
  - Instance information could not be synchronized after static microservice information was deleted.
  - The polling policy did not take effect on multiple versions and multiple instances.
- The local routing rule could not be used after a dark launch rule was deleted.

1.4.6

- Fixed bugs
  - The manual circuit breaker did not take effect.
  - Retry could be performed but the correct result could not be obtained.
  - Forcible service degradation did not take effect.

1.4.5

- New features
  - Supports HTTP 2.0.
- Fixed bugs
  - Data could not be reported to the CSE dashboard after a microservice was deleted.
  - The `forceClosed` configuration item in the circuit breaker did not take effect.
  - The function for setting routing weight rules was unstable.

1.4.4

- New features
  - Supports multiple projects.
  - Supports dark launch governance.
  - Supports ServiceStage deployment without AK/SK configuration.
- Fixed bugs
  - The client did not update the cache when a server instance was added or deleted.
  - The processing results were different when the service access using different protocols timed out.
  - The configured timeout duration did not take effect.
  - By default, the Dubbo protocol could not be used to call a service with a version other than 0.0.1.
  - The configured Dubbo protocol fault injection policy did not take effect.
  - Retry was not provided for service registry during startup.
  - When the routing rule weights were less than 100%, the remaining traffic was not imported to the latest microservice version.
  - The isolation timeout status code 200 was reported.
  - The admin API was unavailable.
  - The Dubbo provider may have exited abnormally.
  - A call failed if the length of the header field exceeded 6441 characters.
  - Mesher QPS data changed slowly.

1.3.7

- Fixed bugs
  - The default listening address was not configured.
  - The time weight policy did not take effect on the Dubbo protocol.
1.3.3

- New features
  Supports Dubbo applications access.

- Modified features
  Combines some configuration files.

- Fixed bugs
  - Mesher could not be started after an exit due to an initialization error.
  - Service degradation failed when the service degradation policy was set to `returnnull`.
  - The exception code was not generated when Service Mesh failed to obtain the chain.
  - A 404 error was reported when Service Mesh deleted instances upon exit.

1.2

- Modified features
  Hides the configuration of the processing chain from users. This configuration can be customized by users, but remains unchanged by default.

- Fixed bugs
  The CSE could not be started if the network interface card (NIC) was not eth0.

1.1.2

- Modified features
  - Uses the `--mode` command parameter to determine the running mode. The `sidecar` or `per-host` parameter can be added to `--mode`. By default, `sidecar` is added to `--mode`.
  - Registers the frame name, version number, and registry information to the service center.
  - Does not require inputting the environment variables when running the `start.sh` command.
  - Hides the SSL configuration of internal components to simplify the configuration file.
  - Does not require specifying `CSE_REGISTRY_ADDR` and `SERVICE_NAME` when deploying in ServiceStage.

1.1

- New features
  - Adds a new load balancing policy whereby instances are selected based on latency.
  - Supports the fault injection function.

1.0.5

- Modified features
  Sets `appId` of the routing information to match that of the consumer by default, which avoids having to configure the routing table.
Fixed bugs
The mesh process broke down because of the tracing handler configuration.

### 6.5 Key Generation Tool

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.4</td>
<td>2017.12.28</td>
<td>This is the first release. Provide the key encryption tool.</td>
</tr>
</tbody>
</table>

**1.0.4**

- New features
  - Generate the encrypted key file and the AK/SK file at a time.
  - Provide the encryption API.

### 6.6 Local CSE

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.1</td>
<td>2018.07.03</td>
<td>Updated README.md.</td>
</tr>
<tr>
<td>1.0.0</td>
<td>2018.06.11</td>
<td>This is the first release. Local CSE is a GUI-based tool, used for the lightweight service and configuration centers that are developed locally.</td>
</tr>
</tbody>
</table>

**1.0.1**

- Modified features
  - Updates README.md.
  - Updates the public cloud logo.

**1.0.0**

- New features
  - Integrates the local lightweight service center.
  - Integrates the local lightweight configuration center.
  - Provides a lightweight console.
# Edition Description

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Management scale</td>
<td>Nodes</td>
<td>20 (Max.)</td>
<td>&gt; 20</td>
</tr>
<tr>
<td>Microservice</td>
<td>JAVA/Go</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td></td>
<td>ServiceComb,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SpringCloud/Boot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commercial Service</td>
<td>Mesh</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Registry center</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Configuration center</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Real-time dashboard</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Load balancing</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Rate limiting</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Service degradation</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Circuit breaker</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Fault injection</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Blacklist and</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>whitelist</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dark launch</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Distributed</td>
<td>transaction</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Pipeline</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>Container management</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote debugging</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>VM cluster</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Physical machine cluster</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Windows cluster</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Container orchestration</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Task management</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td><strong>Basic O&amp;M</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lifecycle management</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>(deployment, uninstall, start/stop, upgrade, and rollback)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dashboard</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>(application metrics: throughput, latency, and success rate; node metrics: CPU, memory, and disk usage)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event analysis</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Log analysis</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Metric management</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Threshold rules</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Auto scaling</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td><strong>Software Repository</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private image repository</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Docker images</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Private software repository</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td><strong>Application orchestration</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack management</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Template management</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Designer</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Advanced O&amp;M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Transaction analysis</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>SQL performance analysis</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Trace analysis</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Method Tracing</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service manager</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Remote technical support engineers</td>
<td></td>
<td></td>
<td>7*24 hours service response, remote technical support provided by the R&amp;D team</td>
</tr>
<tr>
<td>Onsite support for direct troubleshooting</td>
<td></td>
<td></td>
<td>Time for arriving at the customer site: Beijing, Shenzhen, Hangzhou, and Xi'an: 2 hours Other cities: Next day Workload: For the platinum edition, a maximum of one person day each quarter.</td>
</tr>
</tbody>
</table>