Alibaba Cloud Enterprise Distributed Application Service

Microservice Governance

Issue: 20200623

Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

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

Style	Description	Example
{} or {a|b}	This format is used for a required value, where only one item can be selected.	switch {active|stand}

Contents

1 Overview of microservice governance

Microservice governance is an important function of Enterprise Distributed Application Service (EDAS). In EDAS, you can perform governance on Spring Cloud, Dubbo, and High-speed Service Framework (HSF) microservice-oriented applications, including service search, trace query, outlier instance removal, and service authentication.

2 Spring Cloud microservice governance

2.1 Query Spring Cloud services

You can query the list and details of services of Spring Cloud applications that are deployed in Enterprise Distributed Application Service (EDAS).

Context

You can switch between the old and new versions of the **Service Search** page.

- In the new version, the systems uses EDAS Agent to query services in the EDAS registry, MSE-hosted registry, and on-premises registries, including ZooKeeper, Nacos, Eureka, and Consul.
- In the old version, you can only query services in the EDAS registry.

Limits

- On the new Service Search page, you can query services of Spring Cloud Edgware and later versions and services in all registries.
- On the old Service Search page, you can query services of Spring Cloud Dalston and later versions that are registered in the EDAS registry.

View the service list

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces** to view the **Spring Cloud** services under the current account.

   The following information about a Spring Cloud service is provided: **Service Name**, **App Name**, and **Number of instances**.

   If many services exist, you can filter services by **Service Name**. Filter keywords are case-insensitive.

   Note:

   If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps:

   a. The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically

mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.

b. Check whether the microservice framework version is supported. For more information about the supported versions, see Limits.

View service details

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces**. Click a service name in the service list.
3. On the **Servive Detail** page, view the details of the target service.

   The **Servive Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.

   - **Basic Information**

   - **Service Call Relationship**

     The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address** and **port number**.
   - **Metadata**

     Information under **Metadata** includes metadata of the service and EDAS-provided metadata for implementing microservice capabilities.

2.2 Ensure Spring Cloud application availability through outlier instance removal

In a microservice framework, service calls are affected when consumers cannot perceive the abnormal application instances of a provider. This further affects consumers' serviceability and availability. The outlier instance removal function monitors the availability of application instances and dynamically adjusts them. This ensures successful service calls and improves service stability and quality of service (QoS).

Context

As shown in the following figure, the system includes Applications A, B, C, and D. Applicatio n A can call Applications B, C, and D. Some calls fail if Application A cannot perceive the

abnormal instances of Application B, C, or D. As shown in the following figure, Application B has one abnormal instance, and Applications C and D each have two abnormal instances. The performance and serviceability of Application A may be affected if Applications B, C, and D have many abnormal instances.

You can configure outlier application removal for Application A to ensure its serviceability and availability. This allows Application A to monitor the instance status of Applications B, C, and D and dynamically add or remove instances to ensure successful service calls.

The process of outlier instance removal is as follows.

1. Enterprise Distributed Application Service (EDAS) can detect any abnormal instance of Application B, C, or D and determines whether to remove the abnormal instance from the application based on the configured **Maximum Number of Removed Instances**.
2. Call requests of Application A are not allocated to the removed instance.
3. EDAS detects whether the abnormal instance is recovered at the configured **Recovery Detection Unit Time**.
4. The detection interval is proportional to the detection times and linearly increases with the **Recovery Detection Unit Time**, which is 0.5 minutes by default. When the configured **Maximum Number of Cumulative Rollbacks** is reached, EDAS detects whether the abnormal instance is recovered at the maximum interval.
5. When the instance is recovered, it is added to the instance list of the application to process call requests. The detection interval is reset to **Recovery Detection Unit Time**, for example, 0.5 minutes.

Note:

- When the provider has many abnormal instances (which exceed the configured maximum number), the number of actually removed instances is the same as the configured maximum number.
- When only one available instance is left among the provider's instances, this instance is not removed even if the error rate exceeds the configured threshold.

Create an outlier instance removal policy

1. In the left-side navigation pane, choose **Microservice Governance** > **Spring Cloud/Dubbo/HSF**, and then click **Outlier Ejection**.
2. On the **Outlier Instance Removal** page, click **Create Strategy**.

3. In the **Create Strategy** wizard, set the parameters on the **Basic Info** page and click **Next**.

   - **Namespace**: Select a **region** and a **namespace** from the drop-down lists.
   - **Strategy Name**: Enter a policy name. The name can be up to 64 characters in length.
   - **Framework Used by the Called Service**: Select **Dubbo** or **Spring Cloud** as needed.

4. On the **Select Effect App** page in the **Create Strategy** wizard, select the target application and click **>** to add the application to Selected application. Then, click **Next**.

   After the target application is selected, all abnormal application instances that are called by this application are removed. Call requests from the effective application are not sent to the removed instances.

5. In the **Create Strategy** wizard, set the parameters on the **Configuration Strategy** page and click **Next**.

   - **Exception Type**: Select **Network Anomaly** or **Network Anomaly + Business Anomaly (Dubbo Exception)** as needed.
   - **QPS Lower Limit**: Enter a queries per second (QPS) lower limit based on the statistical time window. The time window of applications in Dubbo 2.7 is 15s, and 10s for applications in Dubbo of other versions and Spring Cloud. When the QPS in a statistical time window, for example, 15s, reaches the specified lower limit, Enterprise Distributed Application Service (EDAS) starts collecting and analyzing error rate statistics.
   - **Lower Error Rate**: Set a call error rate threshold. If the error rate for an instance of the called application exceeds this value, the instance will be removed. Default value: 50%. For example, you set this parameter to 50%. An instance is removed if it is called 10 times in the statistical time window, but six calls fail (that is, the error rate is 60%).
   - **Maximum Number of Removed Instances**: Set the maximum number of abnormal instances to be removed. No more abnormal instances will be removed after the threshold is reached. For example, the total number of instances of an application is 6 and this parameter is set to 60%. The number of instances that can be removed is 3.6

(6 × 60%), which is rounded down to the nearest integer 3. If the calculated result is less than 1, no instance will be removed.

- **Recovery Detection Unit Time**: Set an interval for detecting whether abnormal instances are recovered, in milliseconds. After abnormal instances are removed, EDAS continuously accumulates the detection interval by the specified time unit. Default value: 30000 ms, that is, 0.5 minute.
- **Maximum Number of Cumulative Rollbacks**: Set the maximum number of cumulative rollbacks exceeding which the detection interval is no longer increased. For example, you set **Recovery Detection Unit Time** to 30000 ms and **Maximum Number of Cumulative Rollbacks** to 20. If the abnormal instance remains unrecovered after being detected 20 times, the instance is subsequently detected at an interval of 10 minutes (20 × 30000 ms). If the instance has been recovered before the specified threshold, the detection interval is reset to **Recovery Detection Unit Time**.

  

  Note:

  We recommend that you do not set **Maximum Number of Cumulative Rollbacks** to a large value. A large value will lead to a long detection interval. If the instance is recovered early in the detection interval, the recovery cannot be detected in a timely manner. This results in resource waste and postponed processing of service call requests.

6. In the **Create Strategy** wizard, confirm the settings on the **Complete Creation** page and click **Submit**.

Verify the result

The outlier instance removal function is enabled after you configure an outlier instance removal policy. You can go to the details page of the application configured with outlier instance removal to view the application monitoring information. For example, a **topology** shows whether call requests are still forwarded to abnormal instances. You can check whether **Error Rate per Minute** for application calls is higher than the configured **Lower Error Rate**. Based on such information, you can determine whether the outlier instance removal policy takes effect.

2.3 Implement access control of Spring Cloud applications through service authentication

When you want to improve the security of a microservice-oriented application, you can configure authentication for other applications so that only the applications that match the authentication rule can call the microservice-oriented application.

Context

The following example shows how to use service authentication in Spring Cloud.

Consumers 1, 2, and 3 and a provider belong to the same namespace. Consumers 1, 2, and 3 can call all the paths (Paths 1, 2, and 3) of the provider by default.

You can set an authentication rule for all the paths of the provider. For example, set a blacklist for Consumer 1 to prevent it from calling the provider's all paths, and set a whitelist for Consumers 2 and 3 to allow them to call the provider's paths.

You can also configure authentication for specified paths. In all-interface authentication mode, Consumers 2 and 3 can access all paths of the provider. If Path 2 involves core business or data that cannot be called by Consumer 2, you can set a blacklist for Consumer 2 to prevent it from calling Path 2. Then, Consumer 2 can only call Paths 1 and 3.

The following figure shows the call process with authentication rules configured.

Create a service authentication rule

1. In the left-side navigation pane of **Spring Cloud**, click **Service Authentication**.
2. On the **Service Authentication** page, click **Create Rules**.
3. On the **Create Rules** page, set service authentication parameters, and click **OK**.

   Service authentication rule parameters:

Parameter	Description
Namespaces	The **region** and **Namespaces** where the service is located.
Rule Name	The name of the authentication rule. It can contain uppercase or lowercase letters, digits, underscores (_), and hyphens (-). It can be up to 64 characters in length.
Callee Framework	The framework used by the called application. Select **Spring Cloud**.

Parameter	Description
Called Party	The called application.
Add All Interface Rules **Notice:** **You can add a common rule for all interfaces only once.**	
Called Party	• **Spring Cloud**: The default value is **All Paths**, which cannot be changed. • **Dubbo**: The default value is **All Services /All Interfaces**, which cannot be changed.
Authentication Mode	The service authentication mode. Values: **Whitelist (Allow Call)** and **Blacklist (Deny Call)**. Select an option as needed.
Caller	The application that requires authentication for calling the service. Click **Add Caller** to select multiple applications.
Add Specified Interface Rule **Notice:** The rules added to the specified interface are not appended. Instead, they overwrite the common rules (if any) for all interfaces.	
Callee Interface	**Callee Framework** includes **Spring Cloud** and **Dubbo**. The setting of Callee Interface varies depending on the selected framework. • **Spring Cloud**: the path to the called application. • **Dubbo**: the service and interface of the called application.
Authentication Mode	The service authentication mode. Values: **Whitelist (Allow Call)** and **Blacklist (Deny Call)**. Select an option as needed.
Caller	The application that requires authentication for calling the service. Click **Add Caller** to select multiple applications.

Parameter	Description
State	Specifies whether to enable the rule. • On (default value): the rule is enabled once created. • Off: the rule is not enabled once created. To enable the rule, find it on the **Service Authentication** page and click **Enable** in the **Operating** column.

Verify the result

After the service authentication rule is configured and enabled, verify whether it takes effect as needed.

What's next

After creating a service authentication rule, you can **edit**, **close** (based on the rule status), or **enable** the rule. If service authentication is no longer required, **delete** the rule. The operations are simple and not described here.

2.4 Gracefully disconnect Spring Cloud applications

How do we make service updates or stop processes imperceptibly to consumers when using online applications? This can be done through graceful disconnection. Graceful disconnection can be implemented by modifying related container or framework configurations or deploying applications to Enterprise Distributed Application Service (EDAS) for lifecycle management. This topic describes how to implement graceful disconnection for Spring Cloud applications.

Background

How do we ensure the normal processing of consumer service requests during the period from when applications are stopped to when services are recovered? The most secure and reliable solution is to update your application when no service requests exist. However, in practice, service requests typically exist even when the application is disconnected. A traditional solution is to manually perform three steps: (1) Manually extract traffic. (2) Stop the application. (3) Update your application and then restart it. These manual operations ensure that the update process is imperceptible to clients.

An automatic mechanism can be provided at the container or framework level to manually extract traffic and process received requests. This makes the update process imperceptible

to services and improves the O&M efficiency. This mechanism is called graceful disconnection.

EDAS integrates graceful disconnection into the publishing process so that graceful disconnection is automatically implemented when you stop, deploy, roll back, scale in, and reset the applications in Elastic Compute Service (ECS) clusters of EDAS.

Graceful disconnection for open-source Spring Cloud applications

This section describes how to configure open-source Spring Cloud applications to support graceful disconnection.

1. Providers deregister services.

   ServiceRegistryEndpoint in Spring Cloud Commons provides the service deregistration function, which sends a request to control service disconnection through endpoints.

   ```
   curl -X "POST" "http://{ip}:{port}/actuator/service-registry?status=DOWN" -H "Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8"
   ```

   When service disconnection is controlled through endpoints, the class-related method setStatus(R registration, String status) that implements ServiceRegistry is called to deregister provided services from registries.

2. Consumers update the provider list.

   After services are deregistered from registries, consumers remove the deregistered providers from the subsequently requested provider list. For open-source Spring Cloud applications, load balancing is typically implemented by Ribbon. The following procedure shows how to modify the Ribbon update time.

   The default Ribbon update time is 30s, indicating that a deregistered provider is removed from the provider list 30s after deregistration. You can set the update time to a smaller value to make the provider list update faster. For example, you can set the update time to 2,000 ms as follows.

   ```
   ribbon.ServerListRefreshInterval=2000
   ```

3. Stop the provider application.

   You can stop an application without service impact when the provider no longer receives new requests and all requests received before service deregistration have been processed.

Graceful disconnection for Spring Cloud applications in EDAS

In the preceding solution of graceful disconnection for open-source Spring Cloud applications, ServiceRegistryEndpoint is called based on the Spring Boot Actuator dependency, which is accessible only after security configuration is added. Spring Boot 1 and Spring Boot 2 have different Actuator addresses.

To t1838890.dita#Task72618, change the configured dependency to Nacos Server. When you stop an application in EDAS, EDAS removes the related provider from the service registry. This requires neither additional configuration for the application nor the Spring Boot Actuator dependency. You only need to set the provider list update time to a value less than 3s. The following describes how to perform graceful disconnection in EDAS.

1. Log on to the EDAS console. In the upper-left corner, select a region.
2. In the left-side navigation pane, choose **Application Management** > **Applications**. On the Applications page, click the name of the Spring Cloud application in the ECS cluster.
3. On the Application Details page, perform the following operations to trigger graceful disconnection for Spring Cloud applications in EDAS.
   - In the upper-right corner, click **Stop Application**, **Deploy Application**, or **Roll Back Application**.
   - After the application is stopped, click **Process Instances in Batch** on the right side of the page. On the **Process Instances in Batch** page, select the application that you want to scale in and click **Batch Scale In**.
   - On the **Instance Information** tab, find the application instance and click **Reset** in the Actions column.
4. In the upper-left corner of the Application Details page, the message **A change process is being executed for this application. Status: Executing** appears. Click **View Details** on the right of the message.
5. On the **View Details** page, view the change record on graceful disconnection.
   a) Services are deregistered from the Nacos registry during the **smooth application disconnection** phase, which is displayed in the application change order.
   b) By default, the application is stopped 3s after services are deregistered.

3 Dubbo microservice governance

3.1 Query Dubbo services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of Dubbo applications that are deployed in EDAS.

Context

You can switch between the old and new versions of the **Service Search** page.

- In the new version, the systems uses EDAS Agent to query services in the EDAS registry, MSE-hosted registry, and on-premises registries, including ZooKeeper, Nacos, Eureka, and Consul.
- In the old version, you can only query services in the EDAS registry.

Limits

- On the new Service Search page, you can query services of all Dubbo versions and services in all registries.
- On the old Service Search page, you can only query services of Dubbo 2.7.x that are registered to the EDAS registry through Nacos.

View the service list

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces** to view the **Dubbo** services under the current account.

   The following information about Dubbo services is displayed: **Service Name**, **Version**, **Group**, **App Name**, and **Number of instances**.

   If many services exist, you can filter services by **Service Name**, **IP**, or **App Name**. Filter keywords are case-insensitive. When you search for services by **IP**, pay attention to the following:

   - Elastic Compute Service (ECS) cluster: Enter the IP address of the ECS instance.
   - Container Service Kubernetes cluster: Enter the IP address of a pod.

   Note:

If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps:

a. The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.

b. Check whether the microservice framework version is supported. For more information about the supported versions, see Limits.

View service details

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces**. Click a service name in the service list.
3. On the **Servive Detail** page, view the details of the target service.

   The **Servive Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.

   - **Basic Information**
   - **Service Call Relationship**

     The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address**, **port number**, **serialization method**, and **timeout period**.
   - **Metadata**

     The Metadata section provides **Metadata** and **Interface Metadata**.

     - Information under **Metadata** includes metadata of the service and EDAS-provided metadata for implementing microservice capabilities.
     - **Interface Metadata** includes **Method Name**, **Parameters**, and **Return Type**.

3.2 Ensure Dubbo application availability through outlier instance removal

In a microservice framework, service calls are affected when consumers cannot perceive the abnormal application instances of a provider. This further affects consumers' serviceability and availability. The outlier instance removal function monitors the

availability of application instances and dynamically adjusts them. This ensures successful service calls and improves service stability and quality of service (QoS).

Context

As shown in the following figure, the system includes Applications A, B, C, and D. Applicatio n A can call Applications B, C, and D. Some calls fail if Application A cannot perceive the abnormal instances of Application B, C, or D. As shown in the following figure, Application B has one abnormal instance, and Applications C and D each have two abnormal instances . The performance and serviceability of Application A may be affected if Applications B, C, and D have many abnormal instances.

You can configure outlier application removal for Application A to ensure its serviceability and availability. This allows Application A to monitor the instance status of Applications B, C, and D and dynamically add or remove instances to ensure successful service calls.

The process of outlier instance removal is as follows.

1. Enterprise Distributed Application Service (EDAS) can detect any abnormal instance of Application B, C, or D and determines whether to remove the abnormal instance from the application based on the configured **Maximum Number of Removed Instances**.
2. Call requests of Application A are not allocated to the removed instance.
3. EDAS detects whether the abnormal instance is recovered at the configured **Recovery Detection Unit Time**.
4. The detection interval is proportional to the detection times and linearly increases with the **Recovery Detection Unit Time**, which is 0.5 minutes by default. When the configured **Maximum Number of Cumulative Rollbacks** is reached, EDAS detects whether the abnormal instance is recovered at the maximum interval.
5. When the instance is recovered, it is added to the instance list of the application to process call requests. The detection interval is reset to **Recovery Detection Unit Time**, for example, 0.5 minutes.

 **Note:**

- When the provider has many abnormal instances (which exceed the configured maximum number), the number of actually removed instances is the same as the configured maximum number.
- When only one available instance is left among the provider's instances, this instance is not removed even if the error rate exceeds the configured threshold.

Create an outlier instance removal policy

1. In the left-side navigation pane, choose **Microservice Governance** > **Spring Cloud/ Dubbo/HSF**, and then click **Outlier Ejection**.
2. On the **Outlier Instance Removal** page, click **Create Strategy**.
3. In the **Create Strategy** wizard, set the parameters on the **Basic Info** page and click **Next**.

   - **Namespace**: Select a **region** and a **namespace** from the drop-down lists.
   - **Strategy Name**: Enter a policy name. The name can be up to 64 characters in length.
   - **Framework Used by the Called Service**: Select **Dubbo** or **Spring Cloud** as needed.
4. On the **Select Effect App** page in the **Create Strategy** wizard, select the target application and click **>** to add the application to Selected application. Then, click **Next**.

   After the target application is selected, all abnormal application instances that are called by this application are removed. Call requests from the effective application are not sent to the removed instances.
5. In the **Create Strategy** wizard, set the parameters on the **Configuration Strategy** page and click **Next**.

   - **Exception Type**: Select **Network Anomaly** or **Network Anomaly + Business Anomaly (Dubbo Exception)** as needed.
   - **QPS Lower Limit**: Enter a queries per second (QPS) lower limit based on the statistical time window. The time window of applications in Dubbo 2.7 is 15s, and 10s for applications in Dubbo of other versions and Spring Cloud. When the QPS in a statistical time window, for example, 15s, reaches the specified lower limit, Enterprise Distributed Application Service (EDAS) starts collecting and analyzing error rate statistics.
   - **Lower Error Rate**: Set a call error rate threshold. If the error rate for an instance of the called application exceeds this value, the instance will be removed. Default value: 50%. For example, you set this parameter to 50%. An instance is removed if it is called 10 times in the statistical time window, but six calls fail (that is, the error rate is 60%).
   - **Maximum Number of Removed Instances**: Set the maximum number of abnormal instances to be removed. No more abnormal instances will be removed after the

threshold is reached. For example, the total number of instances of an application is 6 and this parameter is set to 60%. The number of instances that can be removed is 3.6 (6 × 60%), which is rounded down to the nearest integer 3. If the calculated result is less than 1, no instance will be removed.

- **Recovery Detection Unit Time**: Set an interval for detecting whether abnormal instances are recovered, in milliseconds. After abnormal instances are removed, EDAS continuously accumulates the detection interval by the specified time unit. Default value: 30000 ms, that is, 0.5 minute.
- **Maximum Number of Cumulative Rollbacks**: Set the maximum number of cumulative rollbacks exceeding which the detection interval is no longer increased. For example, you set **Recovery Detection Unit Time** to 30000 ms and **Maximum Number of Cumulative Rollbacks** to 20. If the abnormal instance remains unrecovered after being detected 20 times, the instance is subsequently detected at an interval of 10 minutes (20 × 30000 ms). If the instance has been recovered before the specified threshold, the detection interval is reset to **Recovery Detection Unit Time**.

  Note:
  We recommend that you do not set **Maximum Number of Cumulative Rollbacks** to a large value. A large value will lead to a long detection interval. If the instance is recovered early in the detection interval, the recovery cannot be detected in a timely manner. This results in resource waste and postponed processing of service call requests.

6. In the **Create Strategy** wizard, confirm the settings on the **Complete Creation** page and click **Submit**.

Verify the result

The outlier instance removal function is enabled after you configure an outlier instance removal policy. You can go to the details page of the application configured with outlier instance removal to view the application monitoring information. For example, a **topology** shows whether call requests are still forwarded to abnormal instances. You can check whether **Error Rate per Minute** for application calls is higher than the configured **Lower Error Rate**. Based on such information, you can determine whether the outlier instance removal policy takes effect.

3.3 Implement access control of Dubbo applications through service authentication

When you want to improve the security of a microservice-oriented application, you can configure authentication for other applications so that only the applications that match the authentication rule can call the microservice-oriented application.

Context

The following example shows how to use service authentication in Dubbo.

Consumers 1, 2, and 3 and a provider belong to the same namespace. Consumers 1, 2, and 3 can call all the services and interfaces of the provider.

You can set an authentication method for all the services and interfaces of the provider. For example, set a blacklist for Consumer 1 to prevent it from calling the provider's services and interfaces, and set a whitelist for Consumers 2 and 3 to allow them to call the provider's services and interfaces.

You can also set an authentication method for specified services and interfaces of the provider. For example, set a blacklist for Consumer 2 to prevent it from calling the services and Interface 2 of the provider because they involve core business or data. Then, Consumer 2 can only call Interfaces 1 and 3 of the provider.

The following figure shows the call process with authentication rules configured.

Create a service authentication rule

1. In the left-side navigation pane of **Spring Cloud**, click **Service Authentication**.
2. On the **Service Authentication** page, click **Create Rules**.
3. On the **Create Rules** page, set service authentication parameters, and click **OK**.

   Service authentication rule parameters:

Parameter	Description
	The **region** and where the service is located.
Rule Name	The name of the authentication rule. It can contain uppercase or lowercase letters, digits, underscores (_), and hyphens (-). It can be up to 64 characters in length.

Parameter	Description
Callee Framework	The framework used by the called application. Select **Spring Cloud**.
Called Party	The called application.
Add All Interface Rules **Notice:** **You can add a common rule for all interfaces only once.**	
Called Party	• **Spring Cloud**: The default value is **All Paths**, which cannot be changed. • **Dubbo**: The default value is **All Services /All Interfaces**, which cannot be changed.
Authentication Mode	The service authentication mode. Values: **Whitelist (Allow Call)** and **Blacklist (Deny Call)**. Select an option as needed.
Caller	The application that requires authentication for calling the service. Click **Add Caller** to select multiple applications.
Add Specified Interface Rule **Notice:** The rules added to the specified interface are not appended. Instead, they overwrite the common rules (if any) for all interfaces.	
Callee Interface	**Callee Framework** includes **Spring Cloud** and **Dubbo**. The setting of Callee Interface varies depending on the selected framework. • **Spring Cloud**: the path to the called application. • **Dubbo**: the service and interface of the called application.
Authentication Mode	The service authentication mode. Values: **Whitelist (Allow Call)** and **Blacklist (Deny Call)**. Select an option as needed.
Caller	The application that requires authentication for calling the service. Click **Add Caller** to select multiple applications.

Parameter	Description
State	Specifies whether to enable the rule. • On (default value): the rule is enabled once created. • Off: the rule is not enabled once created. To enable the rule, find it on the **Service Authentication** page and click **Enable** in the **Operating** column.

Verify the result

After the service authentication rule is configured and enabled, verify whether it takes effect as needed.

What's next

After creating a service authentication rule, you can **edit**, **close** (based on the rule status), or **enable** the rule. If service authentication is no longer required, **delete** the rule. The operations are simple and not described here.

3.4 Gracefully disconnect Dubbo applications

Online applications must be developed in a way to ensure normal service requests even during the period from when applications are stopped for service upgrade and deployment to when services are restarted and recovered. That is, the entire process must be imperceptible to clients. You need to configure graceful disconnection to properly shut down applications for deployment, stop, rollback, scale-in, and reset.

Background

Graceful disconnection is a series of operations to ensure that an application is properly shut down before you deploy, stop, roll back, scale in, or reset the application, to avoid data errors, loss, application call exceptions, and other problems caused by unexpected shutdown.

How do we ensure the normal processing of consumer service requests during the period from when applications are stopped to when services are recovered? The most secure and reliable solution is to update your application when no service requests exist. However, in practice, service requests typically exist even when the application is disconnected. A traditional solution is to manually perform three steps: (1) Manually extract traffic. (2) Stop

the application. (3) Update your application and then restart it. These manual operations ensure that the update process is imperceptible to clients.

An automatic mechanism can be provided at the container or framework level to manually extract traffic and process received requests. This makes the update process imperceptible to services and improves the O&M efficiency. This mechanism is called graceful disconnect ion.

Enterprise Distributed Application Service (EDAS) integrates graceful disconnection into the publishing process so that graceful disconnection is automatically implemented when you stop, deploy, roll back, scale in, and reset the applications in Elastic Compute Service (ECS) clusters of EDAS.

Configure graceful disconnection for Dubbo applications

- Graceful disconnection for services

  By default, graceful disconnection is enabled for Dubbo applications. The default wait time before application shutdown is 10,000 ms. You can modify the wait time by setting `dubbo.service.shutdown.wait`. For example, set the wait time to 20s by adding the following configuration:

  ```
  dubbo.service.shutdown.wait=20000
  ```

  Graceful disconnection does not ensure that all sent or received requests have been processed before the application is shut down. To ensure that all requests are properly processed before application shutdown, you can set the preceding parameter based on the processing time of your services.
- Graceful disconnection for containers

  You can enable graceful disconnection by setting `dubbo.shutdown.hook` to `true` when you use Dubbo through `org.apache.dubbo.container.Main`.
- QoS-based graceful disconnection

  `ShutdownHook` does not ensure that all shutdown processes are completed. Dubbo supports multi-step shutdown for full service protection.

  Multi-step shutdown is the stopping of an application in multiple steps. You can run auto O&M scripts or perform manual operations to ensure that all scripts are executed.

  Before an application is shut down, you can disconnect all services by using the QoS command `offline`. No more requests are sent to the application after services are disconnected from the registry. A wait time is applied to ensure that all existing requests

are processed. Then, shut down the application through SIGTERM or SIGINT. This ensures that no services are affected.

You can use the QoS feature through telnet or HTTP. For more information, see Dubbo-QoS instructions.

Graceful disconnection for EDAS Dubbo applications

The Dubbo-provided graceful disconnection capability works during system runtime only when an O&M system is used with Dubbo. After Dubbo applications are deployed in EDAS, you can manage the lifecycle of Dubbo applications in EDAS. When you stop Dubbo applications in EDAS, you can view the graceful disconnection records in application change orders.

1. Log on to the EDAS console. In the upper-left corner, select a region.
2. In the left-side navigation pane, choose **Application Management** > **Applications**. On the Applications page, click the name of the Dubbo application in the ECS cluster.
3. On the Application Details page, perform the following operations to trigger graceful disconnection for Dubbo applications in EDAS.
   - In the upper-right corner, click **Stop Application**, **Deploy Application**, or **Roll Back Application**.
   - After the application is stopped, click **Process Instances in Batch** on the right side of the page. On the **Process Instances in Batch** page, select the application that you want to scale in and click **Batch Scale In**.
   - On the **Instance Information** tab, find the application instance and click **Reset** in the Actions column.
4. In the upper-left corner of the Application Details page, the message **A change process is being executed for this application. Status: Executing** appears. Click **View Details** on the right of the message.
5. On the **View Details** page, view the change record on graceful disconnection.
   a) Services are deregistered from the Nacos registry during the **smooth application disconnection** phase, which is displayed in the application change order.
   b) The system checks whether the Dubbo application enables the QoS port when **the application is being stopped**. If the QoS port is enabled, the system calls the offline

command to disconnect services. The application stop process disconnects services after receiving the command.

```
2019-08-19 15:01:46.629  INFO 31887 --- [ qos-worker-3-1] o.apache.dubbo.qos.
command.impl.Offline  :  [DUBBO] receive offline command, dubbo version: 2.7.3,
current host: 30.5.121.2
2019-08-19 15:01:46.630  INFO 31887 --- [ qos-worker-3-1] o.a.dubbo.registry
.nacos.NacosRegistry   :  [DUBBO] Unregister: dubbo://30.5.121.2:20883/com.
alibaba.middleware.sp.dubbo.samples.DemoApi? anyhost=true&application=
dubbo-nacos-provider-from-d&bean.name=demo1&deprecated=false&dubbo=2.0.
2&dynamic=true&generic=false&group=DUBBO&interface=com.alibaba.middleware
.sp.dubbo.samples.DemoApi&methods=hi,echo,test1,plus&pid=31887&register=
true&release=2.7.3&revision=1.0
```

Notice:

- kill -9 pid is only used to shut down an application but does not support graceful disconnection. We recommend that you use the application stop function provided by the EDAS console.
- The wait time set by timeout during graceful disconnection indicates the maximum time for running each destroy rather than the total wait time for all steps. For example, when the wait time is set to 5s, the system waits for 5s when shutting down the server and client, respectively.
- To enable EDAS to access port 22222 and call the qos command, you must enable the qos command and expose the default port 22222.

3.5 Use Dubbo Admin to manage Dubbo services

3.5.1 Overview of using Dubbo Admin to manage Dubbo services

A good service governance solution is essential for building a distributed system, helping you greatly reduce the cost of collaborative development and improve the version iteration efficiency.

Limits

- Dubbo version: the Dubbo service governance console provides functions only in Dubbo 2.7.3 and later.
- Registry: the service search and service list features are only available for Dubbo applications that use Nacos for service registration and discovery.

- Metadata center: the service method list is displayed only for Dubbo applications that use Nacos as the metadata center.
- Configuration center: service governance is only available for Dubbo applications that use Nacos as the configuration center.

We recommend that you use Nacos as the registry, configuration center, and metadata center and use the latest Dubbo 2.7.x version.

Scenarios

Before service governance is applied, Remote Procedure Call (RPC) calls are manually configured in point-to-point mode. This results in high O&M costs and is prone to errors. It is difficult to evaluate system stability at runtime. Service governance is introduced to centrally manage RPC calls. Service governance is designed to improve the performance of distributed systems, reduce repetitive work, avoid problems that arise from manual configuration of physical files, and reduce the technology costs of developers.

Service governance solutions and capabilities vary depending on different product frameworks. Service governance is essential for microservice implementation. In addition to RPC calling, service governance also includes service registration, subscription, and change delivery. Service governance supports auxiliary actions such as statistics reporting, authentication, routing rule implementation, and load balancing. When selecting a microservice framework, pay special attention to the service governance capabilities that it provides.

The following are some typical service governance cases:

- Version-based service management, used for canary release
- Request replication and playback, used to simulate traffic for stress testing
- Request tagging, used for real-time online stress testing
- More flexible load balancing and routing policies
- A built-in circuit breaking mechanism that avoids the avalanche effect of the entire distributed system

Dubbo service governance features

- t1839111.dita#topic517
- t1839112.dita#task336
- t1839113.dita#task289
- t1839114.dita#task384

- t1839115.dita#task_2329472
- t1839116.dita#topic545
- t1839117.dita#task947
- t1839118.dita#topic855

3.5.2 Service search

Service search is the most basic function of Dubbo OPS. It currently supports query by service, application, and IP address.

Query Dubbo services

In the drop-down list on the right of **Search Dubbo Services or Applications**, select a query dimension, enter a keyword in the text box, and click **Search**.

You can enter a service name, IP address, or application name. Enter a service name in the format of `group/service:version`. Fuzzy search is supported. You can enter `*` or leave it blank to search all services. Fuzzy search is not supported for IP addresses and applications.

Query service details

1. In the **Query Result** section, find the target service, and click **Details** in the **Actions** column.
2. On the **Dubbo Service Details** page, view the basic information, details, and metadata about the service.
   - **Basic Information**: includes the service, application, group, and version information.
   - **Service Information**: includes the Provider and Consumer tabs, which display information such as the IP address, port, and application name.
   - **Metadata**: is only displayed for Dubbo 2.7 and later for the moment.

3.5.3 Conditional routing

A routing rule determines the target server for a Dubbo service call.

Procedure

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.

3. In the **Create Conditional Routing Rule** dialog box, set the rule parameters and click **Create**.
   - **Service Name**: in the format of `group/service:version`. **group** and **version** can be left blank.
   - **Application Name**: the name of an application.
   - **Rule Content**: for more information, see Routing rules.

3.5.4 Label-based routing

Label-based routing divides the providers of one or more services into the same group to limit traffic within this group. This implements traffic isolation and provides basic capabilities for blue-green release and canary release.

Procedure

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Label Routing Rule** dialog box, set **Application Name** and **Rule Content**, and then click **Create**.

   For more information about **Rule Content**, see Routing rules.

3.5.5 Blacklist and whitelist

Blacklists and whitelists are part of conditional routing. You can specify blacklists and whitelists at the service and application levels.

Procedure

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Blacklist and Whitelist Rules** dialog box, set the rule parameters and click **Save**.
   - **Service Unique ID**: the name of a service, in the format of `group/service:version`.
   - **Application Name**: the name of an application.
   - **Whitelist**: a list of IP addresses that are allowed to access the service. Separate multiple IP addresses with commas (,).
   - **Blacklist**: a list of IP addresses that are prevented from accessing the service. Separate multiple IP addresses with commas (,).

3.5.6 Dynamic configuration

In Dubbo, you can configure dynamic configuration rules to dynamically adjust Remote Procedure Call (RPC) calls without application restart. In Dubbo 2.7.0 and later, you can modify dynamic configuration at the service and application levels.

Set a dynamic configuration

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Dynamic Configuration** dialog box, set a dynamic configuration rule and click **Create**.

Notice:

In Dubbo service governance of Enterprise Distributed Application Service (EDAS), do not configure weights and load balancing in dynamic configuration. EDAS provides Weight configuration and Load balancing.

- Service Name: in the format of `group/service:version`. `group` and `version` can be left blank.
- Example: `DUBBO/com.alibaba.edas.HelloService:1.0.0`. Application Name: the name of an application created in EDAS.
- Rule Content: Enter the parameters for dynamic configuration. Example:

```
{
 "configVersion": "v2.7",
 "enabled": true,
 "configs": [
  {
   "addresses": [
    "0.0.0.0"
   ],
   "providerAddresses": [
    "1.1.1.1:20880",
    "2.2.2.2:20881"
   ],
   "side": "consumer",
   "parameters": {
    "timeout": "6000",
    "cluster": "failfast"
   }
  },
  {
   "addresses": [
    "0.0.0.0"
   ],
   "side": "consumer",
   "applications/services": [],
   "parameters": {
    "threadpool": "fixed",
```

```
      "threads": 200,
      "iothreads": 4,
      "dispatcher": "all"
    }
   }
  ]
}
```

In the preceding configuration example:

- configVersion: the version of Dubbo.
- enabled: specifies whether the overwrite rule takes effect. If you leave this parameter unspecified, the overwrite rule takes effect by default.
- configs: the content of a rule. You can specify n (n >= 1) rule bodies, such as side, applications, services, parameters, addresses, and providerAddresses.

Clarify the following configurations:

a. Whether the current configuration is applied to consumers or providers:

- Consumers: `"side": "consumer"`. When configuring consumers, you can use providerAddress and applications to select provider examples or applications.
- Providers: `"side": "provider"`.

b. Scope of the current configuration on application instances:

- All instances: `"addresses": ["0.0.0.0"]` or `"addresses": ["0.0.0.0:*"]`, depending on the side value.
- Specified instances: `"addresses": [Instance address list]`.

c. Properties to be modified.

Examples of dynamic configuration rules

- Disable provider applications: this feature is used to temporarily remove a provider instance. To disable access by consumers, apply a routing rule.

```
{
  "configVersion": "v2.7",
  "enabled": true,
  "configs": [
   {
    "addresses": ["10.20.153.10:20880"],
    "side": "provider",
    "parameters": {
      "disabled": true
    }
   }
  ]
```

```
}
```

- Graceful service degradation: This feature is used to temporarily block an erroneous non-critical service.

```
{
   "configVersion": "v2.7",
   "enabled": true,
   "configs": [
    {
     "addresses": ["10.20.153.10:20880"],
     "side": "provider",
     "parameters": {
       "force": "return null"
     }
    }
   ]
}
```

3.5.7 Weight configuration

Weight configuration is applicable to load balancing. You can configure the weights of load balancing-involved Dubbo applications in Enterprise Distributed Application Service (EDAS) that use Nacos for service registration and discovery. You can increase or reduce the weights of some addresses through weight configuration.

Create a weight rule

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Weight Rule** dialog box, set the weight and click **Create**.

> **Note:**
> Set either **Service Name** or **Application Name**.

- **Service Name**: in the format of `group/service:version`. `group` and `version` can be left blank. Example: `DUBBO/com.alibaba.edas.HelloService:1.0.0`.
- **Application Name**: the name of an application created in EDAS.
- **Weight**: Enter an integer greater than or equal to 0.
- **Address List**: Enter the addresses to which you want to apply weight configuration.

Result

After creating a weight rule, you can search for, view, modify, and delete the rule on the Weight Configuration page.

3.5.8 Load balancing

Dubbo is a distributed service framework that avoids single point of failure (SPOF) and supports service scale-out. Dubbo supports load balancing. Dubbo distributes call requests from consumers to prevent a small number of providers from being overloaded. Some requests time out if providers are overloaded. Therefore, it is necessary to keep load balancing among providers.

Procedure

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Dynamic Configuration** dialog box, set a load balancing rule and click **Create**.
   - **Service Name**: in the format of `group/service:version`. `group` and `version` can be left blank. Example: `DUBBO/com.alibaba.edas.HelloService:1.0.0`.
   - **Application Name**: the name of an application created in Enterprise Distributed Application Service (EDAS).
   - **Method**: the method to which load balancing applies. You can configure a single method or enter `*` to indicate all methods.
   - **Policy**: Load balancing supports Random, Round Robin, and Least Active.
     - **Random**:

       Set a random probability based on a weight. The probability of collision on a section is high. The evenness of call request distribution is proportional to the number of call requests. Call requests are evenly distributed based on a weighted probability. This helps dynamically adjust provider weights.
     - **RoundRobin**:

       Set a round robin ratio based on an agreed weight. The round robin algorithm distributes call requests more evenly than the random probability algorithm. However, slow providers accumulate call requests. For example, call requests that are distributed to a slow but functioning instance accumulate and cannot be processed in time.
     - **LeastActive**:

       Call requests are randomly distributed among instances with the same difference in the number of active requests before and after a call. Slow providers receive

fewer call requests because they have a greater difference in the number of active requests before and after a call.

Result

After creating a load balancing rule, you can search for, view, modify, and delete the rule on the Load Balancing page.

3.5.9 Global configuration

The global configuration feature of Dubbo allows you to set global configurations that apply to all instances of an application or all applications, implementing central management of configurations.

Set a Dubbo global configuration

1. At the top of the page, click the tab.
2. In the **Query Result** section of the tab, click **Create** on the right.
3. In the **Create Dubbo Configuration** dialog box, set a rule. For more information, see Global configuration rules. After the configuration is complete, click **Create**.

Global configuration rules

The global configuration feature allows you to centrally manage configurations. Global configuration is the same as local configuration in terms of content and format. Global configuration can be regarded as external storage of `dubbo.properties`. Then, the configuration center is used to extract the public configurations for central management.

```
dubbo.protocol.name=dubbo
dubbo.protocol.port=20880

dubbo.application.qos.port=22222
```

Global configuration priorities

By default, global configuration takes precedence over local configuration. Therefore, the following configuration overwrites local configuration. The configuration priorities are as follows: 1. JVM System Properties, -D parameter; 2. Externalized Configuration, global configuration (which is described in this topic); 3. Configurations collected by ServiceConfig and ReferenceConfig; 4. Local configuration file dubbo.properties

You can set the following parameter to adjust priority of configurations in the configuration center:

```
-Ddubbo.config-center.highest-priority=false
```

Scope of global configuration

The global configuration feature supports configuration at the global level and application level. Configurations at the global level are shared by all Dubbo applications, whereas configurations at the application level are maintained by every application and are visible only to this application.

Result

After setting a global configuration, you can search for, view, modify, and delete global configuration rules on the Global Configuration page.

4 HSF microservice governance

4.1 Query HSF services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of High-speed Service Framework (HSF) applications that are deployed in EDAS.

View the service list

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces** to view the **HSF** services under the current account.

   The following information of HSF services is displayed: **Service Name**, **Version**, **Group**, **App Name**, and **Number of Instances**.

   If many services exist, you can filter services by **Service Name**, **IP**, or **App Name**. Filter keywords are case-insensitive. When you search for services by **IP**, pay attention to the following:

   - Elastic Compute Service (ECS) cluster: Enter the IP address of the ECS instance.
   - Container Service Kubernetes cluster: Enter the IP address of a pod.

   Note:
   If you can query services of your applications on the old Service Search page but not on the new Service Search page, troubleshoot the problem by following these steps:

   The new Service Search page is released at 00:00:00 of January 20, 2020. You must restart your applications after this time point so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before querying services on the new Service Search page.

View service details

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Query**.
2. On the **Service Query** page, select a **region** and a **Namespaces**. Click the **Spring Cloud**, **Dubbo**, or **HSF** tab. Click a specific service name in the service list.

3. On the **Servive Detail** page, view the details of the target service.

   The **Servive Detail** page provides the following information: **Basic Information**, **Service Call Relationship**, and **Metadata**.

   - **Basic Information**

   - **Service Call Relationship**

     The Service Call Relationship section provides the **Service Providers** and **Service Consumers** tabs, which list information such as the **IP address**, **port number**, **serialization method**, and **timeout period**.

   - **Metadata**

     Interface Metadata includes **Method Name**, **Parameters**, and **Return Type**.

4.2 Ensure HSF application availability through outlier instance removal

In a microservice framework, service calls are affected when consumers cannot perceive the abnormal application instances of a provider. This further affects consumers' serviceability and availability. The outlier instance removal function monitors the availability of High-speed Service Framework (HSF) applications and service instances and dynamically adjusts them. This ensures successful service calls and improves service stability and quality of service (QoS).

Context

As shown in the following figure, the system includes Applications A, B, C, and D. Applicatio n A can call Applications B, C, and D. Some calls fail if Application A cannot perceive the abnormal instances of Application B, C, or D. As shown in the following figure, Application B has one abnormal instance, and Applications C and D each have two abnormal instances . The performance and serviceability of Application A may be affected if Applications B, C, and D have many abnormal instances.

You can configure outlier application removal for Application A to ensure its serviceability and availability. This allows Application A to monitor the instance status of Applications B, C, and D and dynamically add or remove instances to ensure successful service calls.

The process of outlier instance removal is as follows.

1. Enterprise Distributed Application Service (EDAS) can detect any abnormal instance of Application B, C, or D and determines whether to remove the abnormal instance from the application based on the configured **Maximum Number of Removed Instances**.
2. Call requests of Application A are not allocated to the removed instance.
3. EDAS detects whether the abnormal instance is recovered at the configured **Recovery Detection Unit Time**.
4. The detection interval is proportional to the detection times and linearly increases with the **Recovery Detection Unit Time**, which is 0.5 minutes by default. When the configured **Maximum Number of Cumulative Rollbacks** is reached, EDAS detects whether the abnormal instance is recovered at the maximum interval.
5. When the instance is recovered, it is added to the instance list of the application to process call requests. The detection interval is reset to **Recovery Detection Unit Time**, for example, 0.5 minutes.

Note:

- When the provider has many abnormal instances (which exceed the configured maximum number), the number of actually removed instances is the same as the configured maximum number.
- When only one available instance is left among the provider's instances, this instance is not removed even if the error rate exceeds the configured threshold.

Create an outlier instance removal policy

For HSF applications, you can create application- and service-level outlier instance removal policies.

1. In the left-side navigation pane, choose **Application Management** > **Configuration Management**.
2. On the **Configuration Management** page, select a **region** and a **Namespaces**. On the right, click **Create +**.

3. On the **Create Configuration** page, complete the settings and click **Publish** at the bottom of the page.

   Parameters for configuring outlier instance removal:

   - **Data ID**: the ID of your configuration, in the format of <App ID>. QOSCONFIG. You can obtain the app ID on the Application Details page.
   - **Group**: is set to HSF and cannot be modified.
   - **Target Region**: is set to the **region** that you selected before configuration, and cannot be modified.
   - **Configuration Body**: Enter the policy for removing outlier instances.

     Configure an outlier removal policy for HSF applications based on properties and their values. You can configure an outlier removal policy at the application or service level . The following provides configuration examples at these two levels.

     

     Note:

     The service-level configuration takes precedence over the application-level configuration.

     - Example of configuring an application-level outlier instance removal policy

       ```
       {
       "DEFAULT": {
       "errorRateThreshold":0.5,
       "isolationTime":60000,
       "maxIsolationRate":0.2,
       "maxIsolationTimeMultiple":15,
       "qosEnabled":true,
       "requestThreshold":20,
       "timeWindowInSeconds":10,
       "ipDimension":true
       }
       }
       ```

     - Example of configuring a service-level outlier instance removal policy

       ```
       {
       "DEFAULT": {
       "errorRateThreshold":0.5,
       "isolationTime":60000,
       "maxIsolationRate":0.2,
       "maxIsolationTimeMultiple":15,
       "qosEnabled":true,
       "requestThreshold":20,
       "timeWindowInSeconds":10
       },
       "service:version": {
       "errorRateThreshold":0.5,
       "isolationTime":60000,
       "maxIsolationRate":0.2,
       ```

```
"maxIsolationTimeMultiple":15,
"qosEnabled":true,
"requestThreshold":20,
"timeWindowInSeconds":10
}
}
```

If you have other requirements, see Parameters for configuring outlier instance removal.

Parameters for configuring outlier instance removal

You can configure an outlier instance removal policy through properties on the Configuration Management page, or by using the -D JVM parameter. The configuration completed on the Configuration Management page takes precedence over the configuration through the -D parameter. We recommend that you complete configuration on the Configuration Management page.

Parameter	Property	-D parameter	Description	Default value
Maximum Number of Calls	requestThreshold	-Dhsf.qos.request.threshold	The outlier instance is removed only when the number of calls in the most recent statistics window exceeds the threshold.	10
Lower Error Rate	errorRateThreshold	-Dhsf.qos.error.rate.threshold	When the error rate of an instance in the called application or service exceeds the threshold, the instance is removed.	0.5

Parameter	Property	-D parameter	Description	Default value
Maximum Number of Removed Instances	maxIsolationRate	-Dhsf.qos.max.isolation.rate	The maximum number of abnormal instances to be removed. If the threshold is reached, no more abnormal instances are removed. For example, the total number of instances of an application is 6 and this parameter is set to 60%. The number of instances that can be removed is 3.6 (6 × 60%), which is rounded down to the nearest integer 3. If the calculation result is less than 1, one instance is removed.	0.2

Parameter	Property	-D parameter	Description	Default value
Recovery Detection Unit Time	isolationTime	-Dhsf.qos.isolation.time	After abnormal instances are removed, Enterprise Distributed Application Service (EDAS) continuously detect whether abnormal instances are recovered at an interval that accumulates by the specified time unit. The unit is milliseconds (ms).	60 × 1,000 ms (1 minute)

Parameter	Property	-D parameter	Description	Default value
Maximum Number of Cumulative Rollbacks	maxIsolationTimeMultiple	-Dhsf.qos.max.isolation.time.multiple	Set the maximum number of cumulative rollbacks exceeding which the detection interval is no longer increased. For example, Recovery Detection Unit Time is set to 60,000 ms and Maximum Number of Cumulative Rollbacks is set to 60. If the abnormal instance remains unrecovered after being detected 60 times, the instance is subsequently detected at an interval of 60 minutes (60 × 60,000 ms). If the instance has been recovered before the specified threshold, the detection interval is reset to Recovery Detection Unit Time.	60

Parameter	Property	-D parameter	Description	Default value
Enable Outlier Instance Removal	qosEnabled	-Dhsf.qos.enable	Specifies whether to enable outlier instance removal for the application or service.	false
Time Window for Statistics	timeWindowInSeconds	-Dhsf.qos.time.window.in.seconds	The time window for statistics on THE Maximum number of calls, that is, the statistical period.	10s

Parameter	Property	-D parameter	Description	Default value
Exception Type	bizExceptionPredicateClassName	-Dhsf.qos.biz.exception.class.name	The exception type of instances of the application or service. All service exceptions are considered as exceptions by default. You can also define specific service exceptions through custom interfaces. For example: • Define all business exceptions as exceptions: com.taobao.hsf.exception.CountBizExceptionPredicate. • Ignore all service exceptions: com.taobao.hsf.exception.IgnoreBizExceptionPredicate. • Set the instance used to implement com.taobao.hsf.Predicate in bizExceptionPredicate.	com.taobao.hsf.exception.CountBizExceptionPredicate, which defines all business exceptions as exceptions

Verify the result

The outlier instance removal function is enabled after you configure an outlier instance removal policy. You can go to the details page of the application configured with outlier instance removal to view the application monitoring information. For example, a **topology** shows whether call requests are still forwarded to abnormal instances. You can check whether **Error Rate per Minute** for application calls is higher than the configured **Lower Error Rate**. Based on such information, you can determine whether the outlier instance removal policy takes effect.

4.3 Graceful HSF application release

This topic describes how to gracefully release HSF applications in Enterprise Distributed Application Service (EDAS).

Prerequisites

- Make sure that your EDAS Container version is 3.5.7 or later. If your EDAS Container version is earlier than 3.5.7, upgrade it. For more information, see #unique_28.
- Make sure that your application is configured with a health check URL.

  To release an HSF application gracefully, configure a health check URL for the applicatio n, so that a script that is automatically executed after the application is started is mounted, which informs EDAS when the application is started.

  By default, the health check URL is not configured in EDAS. You must create and configure the corresponding Controller in the application code.

```
@RestController
public class HealthCheckController {

   @RequestMapping("/health")
   public String healthCheck(){
      return "success";
   }

}
```

  Compared with the port-based health check, URL-based health check reflects the health status of the application more accurately.

  - Before a health check URL is configured:

  - After a health check URL is configured:

Context

During application startup, the service is registered with the registry, and the consumer that subscribes to this service initiates a call to the service provider after receiving the notification. Application startup is a continuous process, during which a service may have been released but the dependent components, such as Redis or database resources, are not initialized. The call fails if any inbound traffic exists at this time. To avoid call failure, you can use the graceful HSF application release function.

All ProviderBeans of HSF are not registered with the registry during initialization. Instead, they are registered only after all beans in the Spring container are initialized and RefreshEvent is sent. In addition, Pandora sets the status to True after all services are registered. O&M is also required. After the app server (Tomcat) is started but before the web server is started, use **curl localhost:12201/hsf/status** to check whether the service has been initialized. If yes, start the web server (Apache or NGINX).

Configure delayed release for HSF applications

1. In the left-side navigation pane, choose**Application Management** > **Applications**
2. On the **Applications** page, click the application you want to release gracefully.
3. On the **Basic Information** page, click **Settings** in the **Application Settings** section, and then click **JVM** in the list that appears.
4. In the **Application Settings** dialog box, click **Custom**. In the **Custom Parameters** field, enter -Dhsf.publish.delayed=true, and then click **Configure JVM Parameters**.

   After the delayed release is configured, the HSF application is not released immediately. Instead, it is released only after the release script notification is received.
5. Log on to the ECS instance where the HSF application is deployed for verification.
   a) Run **telnet localhost 12201** to log on to the Elastic Compute Service (ECS) instance.
   b) Run **cd hsf** to access the HSF directory.
   c) Run **ls** to check the service release status.

Mount the auto release script

1. In the left-side navigation pane, choose**Application Management** > **Applications**
2. On the **Applications** page, click the application you want to release gracefully.
3. On the **Basic Information** page, click **Settings** in the **Application Settings** section, and then click **Mount Script** in the list that appears.

4. In the **Mount Script** dialog box, click **Post-launch Script**, and run the following command to mount the script.

```
grep "PANDORA QOS PORT" /home/admin/edas-assist/edas-assist.pid | sed 's/\x0D
$//' | awk -F":" '{ print "curl localhost:"$2"/hsf/online? k=hsf"}'| sh
```

The following describes the mounted script:

- Content of the edas-assist.pid file

```
PID:19426
HSF PORT:12200
PANDORA QOS PORT:12203
MONITOR PORT:8006
CSP PORT:8719
```

- /home/admin/edas-assist/edas-assist.pid is the file that records the port number of Pandora Boot. The port number of Pandora Boot is randomly generated after EDAS Container is started, which is generally 12201. When the port has been occupied, the next port is used.
- **curl localhost:"$2"/hsf/online? k=hsf** is used to release the HSF application and notify the container that the HSF has been released. You can manually run this command.

Verify the result

You can verify whether the HSF application is gracefully released by using the Quality of Service (QoS) method or the log method.

- QoS

  After the script is configured, you can gracefully release the HSF application when deploying or resetting the application. Log on to the ECS instance in which the applicatio n is deployed and check the service release status.

- Log

  Check whether the /home/admin/logs/hsf/hsf.log file contains the following logs. If yes, the HSF application has received the release command.

```
01 2019-11-26 16:23:03.456 INFO [qos-worker-3-1:t.hsf] [38ef6d01-10a8-405d-8725-
bd7bf121e2e9] [] [] Receive online command.Do HSF online.
```

4.4 View HSF service reports

Service statistics show the runtime status of all the services of all the applications under the current tenant from the past 24 hours, including the service call volume, call time

consumption, and number of call errors. These statistics allow you to easily compare all services in the system.

Procedure

1. In the left-side navigation pane, choose **Microservice Governance** > **Service Statistics**.
2. On the **Service Statistics**page, view the runtime data of services.