Alibaba Cloud Application Configuration Management

SDK Reference

Issue: 20191024

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, consequent**

ial, 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 SDK introduction

You can use ACM SDK to get and listen to configurations.

Available ACM SDKs include:

- ACM Java Native SDK: A Java native SDK for listening to and updating ACM configurations.
- Spring Cloud ACM: Java SDK that supports the Spring Cloud Config interface specification. This SDK won't be maintained any more. Therefore, we recommend that you use *spring-cloud-starter-alibaba-nacos-config* from Nacos instead.
- ACM Node.js SDK: A Node.js native SDK for listening to and updating ACM configurations.
- ACM C++ SDK: A C++ native SDK for listening to and updating ACM configurations .
- ACM Python (open-source): A Python native SDK for listening to and updating ACM configurations.
- ACM PHP (open-source): A PHP native SDK for listening to ACM configurations.
- Nacos Client: A Java native SDK for listening to and updating ACM configurations.

The functions and features of ACM SDKs are summarized in the following table:

Function / Language	Java Native	Java Spring Cloud	Python	Node.JS	C++	PHP	Nacos SDK
Get particular configurations	Supported	Supported	Supported	Supported	Supported	Supported	Supported
Listen to particular configurations	Supported	Supported	Supported	Supported	Supported	Unsupported	Supported
Write configurations	Supported	Unsupported	Unsupported	Unsupported	Unsupported	Supported	Supported

Function / Language	Java Native	Java Spring Cloud	Python	Node.JS	C++	PHP	Nacos SDK
Enumerate configurations under particular tenants	Supported	Unsupported	Unsupported	Unsupported	Unsupported	Unsupported	Unsupported
Support connecting server with Single IP solution	Supported	Unsupported	Supported	Unsupported	Unsupported	Unsupported	Supported
Support connecting server with multiple IP LB method*	Supported	Supported	Supported	Supported	Supported	Supported	Supported
Support user authentication with HmacSHA1 algorithm	Supported	Supported	Supported	Supported	Supported	Supported	Supported
Support local cache backup **	Supported	Supported	Supported	Supported	Supported	Unsupported	Supported
Support ECS instance RAM role authentication	Supported	Supported	Supported	Unsupported	Unsupported	Unsupported	Unsupported

Function / Language	Java Native	Java Spring Cloud	Python	Node.JS	C++	PHP	Nacos SDK
Open-source address	Planned	Planned	*acm-sdk-python*	Planned	Planned	*acm-sdk-php*	*Nacos*

- * Multiple IP LB method is a LoadBalance method based on multiple server IP addresses returned by ACM SDK from the address server, which boosts performance and achieves high availability.
- ** With local cache backup, ACM SDK can read from the local cache backup file saved when getting configurations last time, which avoids client downtime.
- *** `spring-cloud-starter-acm` won't be maintained any more. Therefore, we recommend that you use *spring-cloud-starter-alibaba-nacos-config* from Nacos instead.

2 ACM Java SDK

2.1 ACM Java Native SDK

2.1.1 Prerequisites

This topic explains how to use the ACM API with sample code.

Get the SDK

Enter the following POM configuration to establish API dependency on the SDK.

```
<dependency>
    <groupId>com.alibaba.edas.acm</groupId>
    <artifactId>acm-sdk</artifactId>
    <version>1.0.8</version>
</dependency>
<! -- Remove the following if logging implementation is available. -->
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.1.7</version>
</dependency>
```

Code example

```
import java.util.Properties;
import com.alibaba.edas.acm.ConfigService;
import com.alibaba.edas.acm.exception.ConfigException;
import com.alibaba.edas.acm.listener.ConfigChangeListener;
import com.alibaba.edas.acm.listener.PropertiesListener;
// Sample code, for sample test only.
public class ACMTest {
    // Attribute/Switch
    private static String config = "DefaultValue";
    private static Properties acmProperties = new Properties();
    public static void main(String[] args) {
        try {
            // Copy the corresponding values from the namespaces page
in the console.
            Properties properties = new Properties();
            properties.put("endpoint", "$endpoint");
            properties.put("namespace", "$namespace");
            // Access ACM with instance RAM role
            // properties.put("ramRoleName", "$ramRoleName");
            properties.put("accessKey", "$accessKey");
            properties.put("secretKey", "$secretKey");
            // If it is an encrypted configuration, then add the
following two lines for automatic decryption.
            //properties.put("openKMSFilter", true);
            //properties.put("regionId", "$regionId");
            ConfigService.init(properties);
            // Actively get the configuration.
```

```
            String content = ConfigService.getConfig("${dataId}", "${
group}", 6000);
            System.out.println(content);
            // Add listeners to the configuration during initializa
tion, which calls back a notification when the configuration is
changed.
            ConfigService.addListener("${dataId}", "${group}", new
ConfigChangeListener() {
                public void receiveConfigInfo(String configInfo) {
                    // After the configuration is updated, the latest
value is returned to the user via this callback function.
                    // Remember not to make blocking operations in
callback function. Otherwise the notification thread will be blocked.
                    config = configInfo;
                    System.out.println(configInfo);
                }
            });
            /**
             * The following listener can be used if the content of
 the configuration value is in properties (key=value) format. This
allows you to manage multiple configuration items in one configuration
.
             */
            /**
            ConfigService.addListener("${dataId}", "${group}", new
PropertiesListener() {
                @Override
                public void innerReceive(Properties properties) {
                    // TODO Auto-generated method stub
                    acmProperties = properties;
                    System.out.println(properties);
                }
            });
            **/
        } catch (ConfigException e) {
            e.printStackTrace();
        }
        // Keep the main thread alive throughout the test, because the
 configuration subscription runs in a daemon thread, which exits once
 the main thread exits. The following code is not required in a real
environment.
        while (true) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    // Expose the configuration value via the get configuration API
for external use.
    public static String getConfig() {
        return config;
    }
    // Expose the configuration value via the get configuration API
for external use.
    public static Object getPorpertiesValue(String key) {
        if (acmProperties ! = null) {
            return acmProperties.get(key);
        }
        return null;
    }
```

```
}
```

Options of passing parameters

To help you get onboard, the sample code initializes the parameters with code . However, the real production process may involve different environments (namely different accounts, regions, or namespaces), and the parameters vary with environments, so you must use variables to pass the parameters. To facilitate the input parameter configuration and reduce the configuration cost, ACM provides multiple options of passing parameters.

Note:
EDAS automatically injects for you during the publishing, and therefore you don’t need to take any action.

Initialization parameters	How to pass parameters
endpoint	From highest priority to lowest priority: 1. JVM parameters: `-Daddress.server.domain=xxx` 2. Environment variables: `address_server_domain=xxx` 3. Code: see the preceding sample code
namespace	From highest priority to lowest priority: 1. JVM parameters: `-Dtenant.id=xxx` 2. Code: see the preceding sample code
ramRoleName	Note: The authorization priority is higher than the accesskey/ridgekey priority from high to low 1. JVM parameters: `-Dram.role.name=xxx` 2. Code: see the preceding sample code

Initialization parameters	How to pass parameters
accessKey/secretKey	From highest priority to lowest priority: 1. By files: store accessKey and secretKey in the format of Properties (which meets the requirement of the `public void java.io.Reader .Properties.load(Reader reader )` method) in a file specified with `-Dspas.identity`. If not specified, then the `/home/admin/.spas_key/<ApplicationName>` file is used by default (the application name is specified with `-Dproject.name`) 2. Environment variable: `spas_accesKey=xxx spas_secretKey=xxx` 3. Code: see the preceding sample code

Related documents

- *#unique_5*

2.1.2 Get configurations

Description

It obtains configurations from ACM when the service starts.

```
public static String getConfig(String dataId, String group, long
timeoutMs) throws ConfigException
```

Request parameters

Parameter	Parameter type	Description
dataId	**String**	**Configuration ID. Use a naming rule such as `package.class` (for example `com.taobao.tc.refund.log.level`) to ensure global uniqueness. We recommend that you indicate business meaning of the configuration in the "class" section. All characters must be in lower case. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 256 bytes.**

Parameter	Parameter type	Description
group	String	Configuration group. We recommend that you use `product name : module name` (for example `ACM: Test`) to ensure the uniqueness. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 128 bytes.
timeout	String	Length of configuration read time-out (in ms). Recommended value: 3000.

Return values

Parameter type	Description
String	Configuration value

Request example

```
try {
    // Initialize the configuration service. Retrieves the following
parameters in console with sample code.
    ConfigService.init("${endpoint}", "${namespace}", "${accessKey}",
 "${secretKey}");
    // Actively retrieves configuration
    String content = ConfigService.getConfig("${dataId}", "${group}",
3000);
    System.out.println(content);
} catch (ConfigException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
```

```
}
```

Exception

A `ConfigException` exception is thrown in case of a configuration read time-out or a network error.

2.1.3 Listen for configurations

Description

If you want ACM to push configuration changes, you can use the ACM dynamic listener configuration interface.

```
public static void addListener(String dataId, ConfigChangeListener
Adapter listener)
```

Request parameters

Parameter name	Parameter type	Description
dataId	string	Configuration ID. Use a naming rule like package .class (for example, com.taobao.tc.refund .log.level) to ensure global uniqueness. It is recommended to indicate business meaning of the configuration in the “class” section. All characters must be in lowercase. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 256 bytes .

Parameter name	Parameter type	Description
group	string	Configuration group. We recommend that you use product name: module name (for example ACM: Test) to guarantee the uniqueness. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 128 bytes.
listener	ConfigChangeListener	Listener. Configuration changes go into the callback function of the listener.

Returned values

Parameter type	Description
string	Configuration value. This value is returned through the callback function during initialization or configuration modification.

Request example

```
// Initialize the configuration service, and the console automatically
 obtains the following parameters through the sample code.
Configservice. init ("$ {endpoint}", "$ {namespace }", "$ {accesskey
}", "$ {ridgekey }");
// Add listeners to the configuration during initialization, which
calls back a notification of configuration changes.
Configservice. addlistener ("$ {dataid}", "$ {group}", new fig () {
    public void receiveConfigInfo(String configInfo) {
        // After the configuration is updated, the latest value is
returned to the user by this callback function.
        System.out.println(configInfo);
    }
});
// Keep the main thread alive throughout the test, because the
configuration subscription runs in a daemon thread, which exits once
 the main thread exits. The following code is not required in a real
environment.
while (true) {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
```

```
}
```

2.1.4 Publish configurations

Description

It publishes ACM configurations automatically with program to reduce operation and maintenance costs with automation.

Note:
It uses the same publishing interface to create or modify a configuration. If the specified configuration doesn’t exist, then it creates a configuration. If the specified configuration exists, then it updates the configuration.

```
public static boolean publishConfig(String dataId, String group,
String content) throws ConfigException
```

Request parameters

Parameter	Parameter type	Description
dataId	String	Configuration ID. Use a naming rule such as `package.class` (for example `com.taobao.tc.refund.log.level`) to ensure global uniqueness. It is recommended to indicate business meaning of the configuration in the “class” section. All characters must be in lower case. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 256 bytes.
group	String	Configuration group. We recommend that you use `product name: module name` (for example `ACM: Test`) to ensure the uniqueness. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 128 bytes.
content	String	Configuration content. It must not exceed 100K bytes.

Returned values

Parameter type	Description
boolean	If the publishing is successful

Request example

```
try {
    // Initialize the configuration service. Retrieves the following
parameters in console with sample code.
    ConfigService.init("${endpoint}", "${namespace}", "${accessKey}",
 "${secretKey}");
    // Actively retrieves configuration
    boolean isPublishOk = ConfigService.publishConfig("${dataId}", "${
group}", "${content}");
    System.out.println(isPublishOk);
} catch (ConfigException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Exception

A `ConfigException` exception is thrown in case of a configuration read time-out or a network error.

2.1.5 Delete configurations

Description

It deletes ACM configurations automatically with program to reduce operation and maintenance costs with automation.

Note:
If the specified configuration exists, then it deletes the configuration. If the specified configuration doesn’t exist, then it returns a successful message.

```
public static boolean removeConfig(String dataId, String group) throws
 ConfigException
```

Request parameters

Parameter	Parameter Type	Description
dataId	String	Configuration ID
group	String	Configuration group

Returned values

Parameter type	Description
boolean	If the deletion is successful

Request example

```
try {
    // Initialize the configuration service. Retrieves the following
parameters in console with sample code.
    ConfigService.init("${endpoint}", "${namespace}", "${accessKey}",
 "${secretKey}");
    // Actively retrieves configuration
    boolean isRemoveOk = ConfigService.removeConfig("${dataId}", "${
group}");
    System.out.println(isRemoveOk);
} catch (ConfigException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Exception

A `ConfigException` exception is thrown in case of a configuration read time-out or a network error.

2.2 Spring Cloud ACM

2.2.1 Prerequisites

spring-cloud-starter-acm won't be maintained any more. Therefore, we recommend that you use *spring-cloud-starter-alibaba-nacos-config* from Nacos instead.

Steps for using the Spring Cloud ACM SDK are as follows.

1. **Add Maven dependency.**

   ```
   <dependency>
        <groupId>com.alibaba.cloud</groupId>
        <artifactId>spring-cloud-starter-acm</artifactId>
        <version>1.0.8</version>
    </dependency>
   ```

2. **Configure the application name and the application group.**

   Configure `application.properties` in Spring Boot, and configure `spring.application.group` and `spring.application.name`.

   ```
   spring.application.group=com.alibaba.cloud.acm
   ```

```
spring.application.name=sample-app
```

3. **Configure the ACM environment and authentication information.**

   Edit the `application.properties` **file in Spring Boot, and configure** `alibaba.acm.endpoint`, `alibaba.acm.namespace`, `alibaba.acm.accessKey`, **and** `alibaba.acm.secretKey`:

```
spring.application.group=com.alibaba.cloud.acm
spring.application.name=sample-app
alibaba.acm.endpoint=xxx

# Namespace ID
alibaba.acm.namespace=xxx

# Access ACM with instance RAM role
# alibaba.acm.ramRoleName=xxx

alibaba.acm.accessKey=xxx
alibaba.acm.secretKey=xxx

# If it is an encrypted configuration, then add the following two
lines for automatic decryption.
# alibaba.acm.openKMSFilter=true
# Regionid can be obtained by the Zone ID in the namespace details
# alibaba.acm.regionId=xxx

# If group is not DEFAULT_GROUP, then set alibaba.acm.group manually
.
# alibaba.acm.group=xxx

# Options include properties, yaml, and yml, where properties is the
 default (only version 1.0.8 and above)
#alibaba.acm.file-extension=properties
```

4. **Add application configuration in the ACM console.**

   Log on to the ACM console and create a new configuration under the correspond ing namespace.

   - **Write the Data ID in the following format:**

     `${spring.application.group}:${spring.application.name}.{alibaba.acm.file-extension}`

     For example: `com.alibaba.cloud.acm:sample-app.properties`

   - **Select** `Properties` **for the configuration format, and put the specific key-value pairs in the configuration body:**

```
user.id = 001
user.name = juven2
```

```
user.age = 88
```

Notes

- `spring-cloud-starter-acm 1.0.7` **and higher version now supports** `Spring Boot 2.x`.
- `spring-cloud-starter-acm 1.0.8` **and higher version now supports** `YAML`.
- **We recommend that you use** `2.0.1. RELEASE` **and higher version of** `Spring Boot 2.x`. `2.0.0. RELEASE` **has** *a bug that prevents it from reading old data*.
- **To download the complete sample code, click:** *spring-cloud-acm-sample.zip*.

Related documents

- *#unique_5*

2.2.2 Inject configuration

Use the Spring MVC annotation to inject the configurations and reduce configuration management costs.

The `@Value` **can be used directly to inject the configurations:**

```
@Component
class SampleRunner implements ApplicationRunner {

    @Value("${user.id}")
    String userId;

    @Value("${user.name}")
    String userName;

    @Value("${user.age}")
    int userAge;

    @Override
    public void run(ApplicationArguments args){
        System.out.println(userId);
        System.out.println(userName);
        System.out.println(userAge);
    }

}
```

Note:

If the same key is configured in `application.properties` **of the Spring Boot application and** `${spring.application.group}:${spring.application.name}.properties` **of ACM at the same time, the value in ACM overrides the default value of the application.**

2.2.3 Spring Boot Integration

Health check

Spring Cloud ACM incorporates the *Health Check* of Spring Boot. Access the health endpoint to see if the Spring Boot application is properly connected to the ACM server:

```
{
  "status": "UP",
  "acm": {
    "status": "UP",
    "dataIds": [
      "com.alibaba.cloud.acm:sample-app.properties"
    ]
  },
  "diskSpace": {
    "status": "UP",
    "total": 1000240963584,
    "free": 858827423744,
    "threshold": 10485760
  },
  "refreshScope": {
    "status": "UP"
  }
}
```

@RefreshScope

Spring Cloud ACM also supports the *Refresh Scope* feature of Spring Cloud.

The Bean marked with the annotation `@RefreshScope` automatically listens for the changes of the ACM server and updates the configuration at run time. Sample code:

```
@RestController
@RequestMapping("/sample")
@RefreshScope
class SampleController {

    @Value("${user.name}")
    String userName;

    @RequestMapping("/acm")
    public String simple() {
        return "Hello Spring Cloud ACM!" + " Hello " + userName +
 "!" ;
    }

}
```

The Bean marked with the annotation `@ConfigurationProperties` is subject to Refresh Scope by default.

ACM Endpoint

Spring Cloud ACM has a built-in endpoint named `acm`, which can be accessed by accessing /acm under the management.port (8080 by default) of Spring Boot applications, for example: *http://localhost:8080/acm*

```
{
  "runtime": {
    "sources": [
      {
        "dataId": "com.alibaba.cloud.acm:sample-app.properties",
        "lastSynced": "2017-10-10 10:46:27"
      }
    ],
    "refreshHistory": [
      {
        "timestamp": "2017-10-10 10:46:24",
        "dataId": "com.alibaba.cloud.acm:sample-app.properties",
        "md5": "8692ae986ec7bc345b3f0f4de602ff13"
      }
    ]
  },
  "config": {
    "group": "DEFAULT_GROUP",
    "timeOut": 3000,
    "endpoint": "xxx",
    "namespace": "xxx",
    "accessKey": "xxx",
    "secretKey": "xxx"
  }
}
```

Considerations

- **When using Spring Boot 2.x, the ACM Endpoint path is `/actuator/acm`.**
- **When using Spring Boot 2.x, you must add configuration `management.endpoints.web.exposure.include=*` before you can access ACM Endpoint.**

2.3 Nacos SDK

2.3.1 Nacos Client

This topic explains how to use Nacos Client SDK.

Get the SDK

To! -- Remove the following if logging implementation is available. -- get Nacos Client SDK, add the following configuration to the pom.xml file in the Maven project.

```
<dependency>
```

```
    <groupId>com.alibaba.nacos</groupId>
    <artifactId>nacos-client</artifactId>
    <version>${latest.version}</version>
</dependency>
<>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

Example

Run the following code in your project for configuration management.

```
import com.alibaba.nacos.api.NacosFactory;
import com.alibaba.nacos.api.PropertyKeyConst;
import com.alibaba.nacos.api.config.ConfigService;
import com.alibaba.nacos.api.exception.NacosException;
import com.alibaba.nacos.client.config.listener.impl.Properties
Listener;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Properties;
/**
 * Sample code, for sample test only.
 */
public class NacosExample {
    private static final Logger LOGGER = LoggerFactory.getLogger(
NacosExample.class);
    public static void main(String[] args) throws NacosException {
        Properties properties = new Properties();
        String dataId = "com.alibaba.nacos.example";
        String group = "DEFAULT_GROUP";
        String content = "connectTimeoutInMills=5000";
        // Copy endpoint and namespace from the namespace details on
the namespaces page of the console
        properties.put(PropertyKeyConst.ENDPOINT, "${endpoint}");
        properties.put(PropertyKeyConst.NAMESPACE, "${namespace}");
        // We recommend that you use the accessKey and secretKey of
the RAM account.
        properties.put(PropertyKeyConst.ACCESS_KEY, "${accessKey}");
        properties.put(PropertyKeyConst.SECRET_KEY, "${secretKey}");
        ConfigService configService = NacosFactory.createConfigService
(properties);
        // Publish configuration
        boolean publishConfig = configService.publishConfig(dataId,
group, content);
        LOGGER.info("publishConfig: {}", publishConfig);
        wait2Sync();
        // Query configuration
        String config = configService.getConfig(dataId, group, 5000);
        LOGGER.info("getConfig: {}", config);
        // Listen for configuration changes
        configService.addListener(dataId, group, new Properties
Listener() {
            @Override
            public void innerReceive(Properties properties) {
                LOGGER.info("innerReceive: {}", properties);
            }
        });
        // Update configuration
```

```
        boolean updateConfig = configService.publishConfig(dataId,
group, "connectTimeoutInMills=3000");
        LOGGER.info("updateConfig: {}", updateConfig);
        wait2Sync();
        // Delete configuration
        boolean removeConfig = configService.removeConfig(dataId,
group);
        LOGGER.info("removeConfig: {}", removeConfig);
        wait2Sync();
        config = configService.getConfig(dataId, group, 5000);
        LOGGER.info("getConfig: {}", config);
    }
    private static void wait2Sync() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            // ignore
        }
    }
}
```

Options of passing parameters

To help you get onboard, the sample code initializes the parameters with code. However, the real production process may involve different environments (namely different accounts, regions, or namespaces), and the parameters vary with environments, so you must use variables to pass the parameters. To facilitate the input parameter configuration and reduce the configuration cost, ACM provides multiple options of passing parameters.

Note:
EDAS automatically injects for you during the publishing, and therefore you don’t need to take any action.

Initialization parameters	How to pass parameters
endpoint	**From highest priority to lowest priority:** **1. By JVM parameters:** `-Daddress.server.domain=xxx` **2. By environmental variables:** `address_server_domain=xxx` **3. By code: see the preceding sample code**
namespace	**From highest priority to lowest priority:** **1. By JVM parameters:** `-Dtenant.id=xxx` **2. By code: see the preceding sample code**

Initialization parameters	How to pass parameters
accessKey/secretKey	From highest priority to lowest priority: 1. By files: store accessKey and secretKey in the format of Properties (which meets the requirement of the `public void java.io.Reader .Properties.load(Reader reader )` method) in a file specified with `-Dspas.identity`. If not specified, then the `/home/admin/.spas_key/<ApplicationName>` file is used by default (the application name is specified with `-Dproject.name`) 2. Environment variable: `spas_accesKey=xxx spas_secretKey=xxx` 3. By code: see the preceding sample code

2.3.2 Nacos Spring

This topic explains how to use Nacos Spring SDK.

Get the SDK

To get Nacos Spring SDK, add the following configuration to the pom.xml file in the Maven project.

```
<dependency>
    <groupId>com.alibaba.nacos</groupId>
    <artifactId>nacos-spring-context</artifactId>
    <version>${latest.version}</version>
</dependency>
```

Example

1. **Add `@EnableNacosConfig` annotation to enable the configuration management service of Nacos Spring. Use `@NacosPropertySource` to load the configuration source with the `dataId` of `com.alibaba.nacos.example`, and turn on automatic update.**

```
@Configuration
// The endpoint and namespace can be found in the namespace details
. We recommend that you use the accessKey and secretKey of the RAM
account.
@EnableNacosConfig(globalProperties = @NacosProperties(endpoint =
 "${endpoint}", namespace = "${namespace}",
```

```
accessKey = "${accessKey}", secretKey = "${secretKey}"))
@NacosPropertySource(dataId = "com.alibaba.nacos.example",
autoRefreshed = true)
public class NacosConfiguration {
}
```

2. **Set the value of the properties with the `@NacosValue` annotation of Nacos.**

```
Controller
@RequestMapping("config")
public class ConfigController {

    @NacosValue("${connectTimeoutInMills:5000}", autoRefreshed =
true)
    private int connectTimeoutInMills;

    @RequestMapping(value = "/get", method = GET)
    @ResponseBody
    public int get() {
        return connectTimeoutInMills;
    }
}
```

3. **Open the ACM console and create a new configuration under the corresponding namespace.**

   - **Data ID: `com.alibaba.nacos.example`**
   - **Select `Properties` for the configuration format, and put the specific key-value pairs in the configuration body:**

     ```
     connectTimeoutInMills=3000
     ```

   For the complete sample code, see *nacos-spring-config-example*.

Related documents

- *Nacos spring Project*

2.3.3 Nacos Spring Boot

This topic explains how to use nacos-config-spring-boot-starter.

Get the Starter

To get the nacos-config-spring-boot-starter, add the following configuration to the pom.xml file in the Maven project.

```
<dependency>
    <groupId>com.alibaba.boot</groupId>
    <artifactId>nacos-config-spring-boot-starter</artifactId>
    <version>${latest.version}</version>
```

```
</dependency>
```

Example

1. **Configure the connection in application.properties.**

```
# The value of endpoint and namespace can be found in namespace
details.
nacos.config.endpoint=${endpoint}
nacos.config.namespace=${namespace}
# We recommend that you use the accessKey and secretKey of the RAM
account
nacos.config.access-key=${accessKey}
nacos.config.secret-key=${secretKey}
```

2. **Use `@NacosPropertySource` to load the configuration source with the `dataId` of `com.alibaba.nacos.example`, and turn on automatic update.**

```
@SpringBootApplication
@NacosPropertySource(dataId = "com.alibaba.nacos.example",
autoRefreshed = true)
public class NacosConfigApplication {
    public static void main(String[] args) {
        SpringApplication.run(NacosConfigApplication.class, args);
    }
}
```

3. **Set the value of the properties with the @NacosValue annotation of Nacos.**

```
@Controller
@RequestMapping("config")
public class ConfigController {

    @NacosValue("${connectTimeoutInMills:5000}", autoRefreshed =
true)
    private int connectTimeoutInMills;

    @RequestMapping(value = "/get", method = GET)
    @ResponseBody
    public int get() {
        return connectTimeoutInMills;
    }
}
```

4. **Open the ACM console and create a new configuration under the corresponding namespace.**

   - **Data ID: `com.alibaba.nacos.example`**
   - **Select `Properties` for the configuration format, and put the specific key-value pairs in the configuration body:**

     ```
     connectTimeoutInMills=3000
     ```

   For the complete sample code, see *nacos-spring-boot-config-example*.

Related documents

- *Nacos Config Spring Boot*

2.3.4 Nacos Spring Cloud

This topic explains how to integrate spring-cloud-starter-alibaba-nacos-config.

Get the Starter

Add the following configuration to the pom.xml file of the Maven project to obtain spring-cloud-starter-alibaba-nacos-config.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
    <version>${latest.version}</version>
</dependency>
```

Example

1. **Configure the connection and configuration source in bootstrap.properties. The `dataId` of the configuration source is `com.alibaba.nacos.example.properties`.**

```
# The value of endpoint and namespace can be found in namespace
details.
spring.cloud.nacos.config.endpoint=${endpoint}
spring.cloud.nacos.config.namespace=${namespace}
# We recommend that you use the accessKey and secretKey of the RAM
account
spring.cloud.nacos.config.access-key=${accessKey}
spring.cloud.nacos.config.secret-key=${secretKey}
spring.application.name=com.alibaba.nacos.example
# Specify the configuration extension name, including properties,
yaml, and yml, where properties is the default value.
spring.cloud.nacos.config.file-extension=properties
```

2. **Sets the property value with Spring's @Value annotation. Enable automatic update of configurations with Spring Cloud native annotation @RefreshScope.**

```
@RestController
@RequestMapping("/config")
@RefreshScope
public class ConfigController {

    @Value("${connectTimeoutInMills:5000}")
    private int connectTimeoutInMills;

    public void setConnectTimeoutInMills(int connectTimeoutInMills)
 {
        this.connectTimeoutInMills = connectTimeoutInMills;
    }

    @RequestMapping(value = "/get", method = GET)
    @ResponseBody
    public int get() {
```

```
        return connectTimeoutInMills;
    }
}
```

3. **Open the ACM console and create a new configuration under the corresponding namespace.**
   - **Data ID:** `com.alibaba.nacos.example.properties`
   - **Select** `Properties` **for the configuration format, and put the specific key-value pairs in the configuration body:**

     ```
     connectTimeoutInMills=3000
     ```

   For the complete sample code, see *nacos-spring-cloud-config-example*.

Related documents

- *Spring Cloud Alibaba Nacos Config*

3 ACM Node.js SDK

Install the ACM Client for Node.js

ACM Client for Node.js makes it possible to help front-end developers release the front and back ends independently, and separate development from operations, which improves the development efficiency dramatically.

Enter the following command to install the client:

```
npm i acm-client --save
```

Sample code

Run the following code in your project for configuration management.

```
const ACMClient = require('acm-client');
const co = require('co');
const acm = new ACMClient({
  endpoint: 'acm.aliyun.com', // Can be found in the ACM console
  namespace: '***************', // Can be found in the ACM console
  accessKey: '***************', // Can be found in the ACM console
  secretKey: '***************', // Can be found in the ACM console
  requestTimeout: 6000, // Length of request time-out, 6s by default.
});
// Actively pull the configuration.
co(function*() {
  const content= yield acm.getConfig('test', 'DEFAULT_GROUP');
  console.log('getConfig = ',content);
});
// Listening for data updating
acm.subscribe({
  dataId: 'test',
  group: 'DEFAULT_GROUP',
}, content => {
  console.log(content);
});
```

Error events handling

```
acm.on('error', function (err) {
  // Logs can be recorded here centrally
  // If error events are not listened for, all exceptions get printed
to stderr.
```

```
});
```

API description

- **API for getting configurations**

  Used to get the configuration from ACM when the service starts.

  ```
  function* getConfig(dataId, group)
  ```

- **Request parameters**

Parameter name	Parameter type	Description
dataId	string	Configuration ID. Use a naming rule like package .class (for example, com.taobao.tc.refund .log.level) to ensure global uniqueness. It is recommended to indicate business meaning of the configuration in the “class” section. All characters must be in lowercase. Only English characters and four special characters (".", ":", "-", and "_") are allowed . It must not exceed 256 bytes.
group	string	Configuration group. We recommend that you use product name: module name (for example ACM: Test) to guarantee the uniqueness. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 128 bytes.

- **Return values**

Parameter type	Description
string	Configuration value

- **API for configuration listening**

  If you want ACM to push configuration changes, you can use the ACM dynamic configuration listening interface.

```
function subscribe(info, listener)
```

- **Request parameters**

Parameter name	Parameter type	Description
info	Object	Info.dataId: Configuration ID. info.group: Configuration group
listener	Function	Listener. Configuration changes go into the callback function of the listener.

- **API for canceling configuration listening**

  Cancels configuration listening when the service starts.

```
function unSubscribe(info, [listener])
```

- **Request parameters**

Parameter name	Parameter type	Description
info	Object	info.dataId: Configuration ID. Info.group: Configuration groupp|
listener	Function	Callback function. (Optional, all listener functions are removed if this parameter is not passed in.)

Node.js project link

https://www.npmjs.com/package/acm-client

Feedback

- *@hustxiaoc*

4 ACM C++ SDK

This topic explains how to use ACM C++ SDK. ACM C++ SDK only supports Linux platform.

Install ACM C++ SDK

1. **Download SDK dependency package:** *ACM C++ SDK*
2. **Extract the downloaded package to create the following directory structure:**
   - **example/**
   - **include/**
   - **lib/**

   The preceding directories and files serve the following purposes:
   - **example: acm.cpp demonstrates how to use SDK. Makefile compiles and manages the example directory.**
   - **include: the header files to be included in your own program.**
   - **lib: The directory contains the 64-bit static library and dynamic library.**
3. **The environment variable LD_LIBRARY_PATH specifies the ACM dynamic library search path, which is also the path for installation.**

   ```
   export LD_LIBRARY_PATH=/usr/lib64:/usr/lib:/lib:/lib64:../lib
   ```
4. **Go to the example directory and modify the start parameter of the acm.cpp file and the dataId and group parameters. Run the make command for compiling. Run the following sample code:**

   ```
   cd example //Enter the example directory
   vim acm.cpp //Modify the acm.cpp file
   make //Compile the sample code
   ./acm //Run the sample code
   ```

Sample code

Run the following code in your project for configuration management.

```
#include "ACM.h"
using namespace std;
using namespace acm;
//Define the listener.
class MyListener : public ManagerListener
{
public:
```

```
    MyListener(const std::string&amp; data_id, const std::string&amp;
group):data_id_(data_id),group_(group){}
    virtual ~MyListener()
    {}
    virtual void getExecutor()
    {
        printf("data_id:%s group:%s getExecutor\n",data_id_.c_str(),
group_.c_str());
    }
    //Callback during configuration modification
    virtual void receiveConfigInfo( std::string &amp;configInfo)
    {
        printf("data_id:%s group:%s configInfo:\n%s\n", data_id_.c_str
(), group_.c_str(), configInfo.c_str());
        config_ = configInfo;
    }
private:
    std::string data_id_;
    std::string group_;
    std::string config_;
};
int main() {
    ACM::init("acm.aliyun.com", // endpoint: Can be found in the ACM
console.
              "***************", // namespace: Can be found in the ACM
 console.
              "***************", // accessKey: Can be found in the ACM
 console.
              "***************"); // secretKey: Can be found in the
ACM console.
    //The dataId and group of the configuration that is listened for
    std::string dataId = "${dataId}";
    std::string group = "${group}";
    std::string content;
    //Actively pull the configuration
    ACM::getConfig(dataId, group, 5000, content);
    printf("get ok config %s\n", content.c_str());
    MyListener* listener = new MyListener(dataId, group);
    //Listen for configuration changes
    ACM::addListener(dataId, group, listener);
    printf("add listener ok %s %s\n", dataId.c_str(), group.c_str());
    do {
        printf("input q to quit\n");
    } while (getchar() ! = 'q');
    return 0;
}
```

Interface description

- **Interface for initialization**

  Set endpoint, nameSpace, accessKey, and secretKey.

```
static void init(const char* endpoint,
                 const char* nameSpace,
                 char* accessKey,
```

```
                    char* secretKey);
```

Parameters are described as follows:

Parameter name	Parameter type	Description
endpoint	const char*	ACM domain name, which is found in the console
nameSpace	const char*	Namespace, which is found in the console
accessKey	char*	accessKey, which is found in the console
secretKey	char*	secretKey, which is found in the console

- **Interface for getting configurations**

  Obtains configurations from ACM when the service starts.

```
static bool getConfig(const std::string &amp;dataId,
                      const std::string &amp;group,
                      int timeoutMs,
                      std::string &amp;content);
```

Parameters are described as follows:

Parameter name	Parameter type	Description
dataId	const string	Configuration ID. Use a naming rule like package .class (for example, com.taobao.tc.refund .log.level) to ensure global uniqueness. It is recommended to indicate business meaning of the configuration in the “class” section. All characters must be in lower case. Only English characters and four special characters (".", ":", "-", and "_") are allowed . It must not exceed 256 bytes.

Parameter name	Parameter type	Description
group	const string	Configuration group. We recommend that you use product name: module name (for example ACM: Test) to guarantee the uniqueness. Only English characters and four special characters (".", ":", "-", and "_") are allowed. It must not exceed 128 bytes.
timeoutMs	int	Time-out period
content	string	Returned configuration content

- Listening to configuration interface

  If you want ACM to push configuration changes, you can use the ACM dynamic configuration listening interface.

```
static void addListener(const std::string &amp;dataId,
                        const std::string &amp;group,
                        ManagerListener* listener);
```

  Parameters are described as follows:

Parameter name	Parameter type	Description
dataId	const string	dataId
group	const string	group
listener	ManagerListener	Listener. The listener callback function will be run when configuration is modified.

- Cancel configuration listening interface

  Cancels configuration listening when the service starts.

```
static void removeListener(const std::string &amp;dataId,
                           const std::string &amp;group,
```

```
                                ManagerListener* listener);
```

Parameters are described as follows:

Parameter name	Parameter type	Description
dataId	const string	dataId
group	const string	group
listener	Manager listener	Listener to be canceled

5 ACM PHP SDK

ACM provides PHP SDK for managing application configurations with ACM for PHP programs.

ACM PHP SDK is now open-sourced. For instructions, see *Github*.

6 ACM Python SDK

ACM provides Python SDK for managing application configurations with ACM for Python program.

ACM Python SDK is now open-sourced. For instructions, see *Github*.